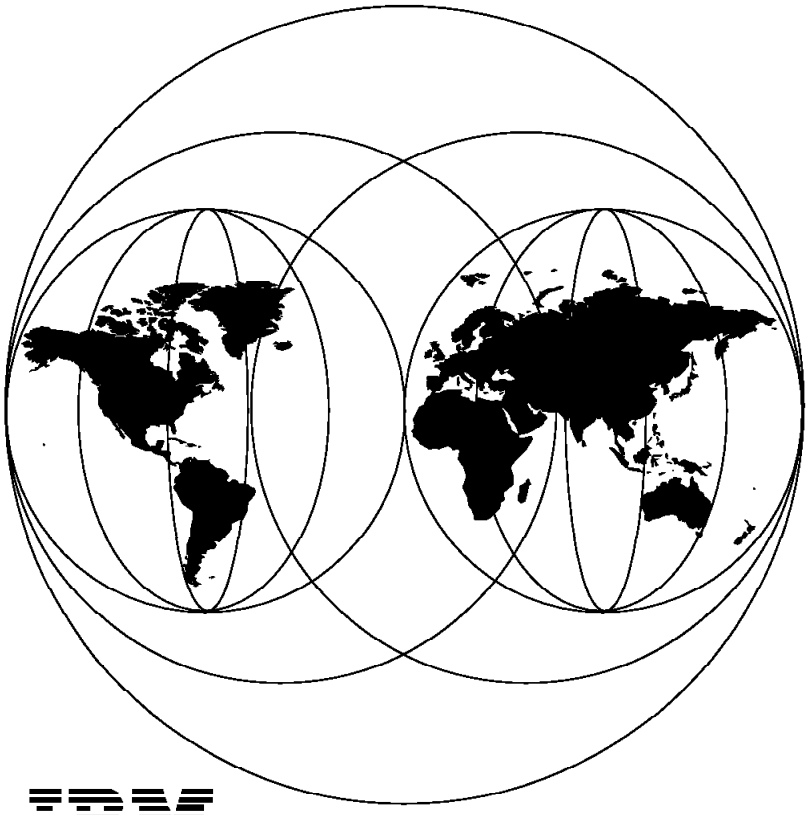
SG24-4658-00

International Technical Support Organization

Migrating and Managing Data on RS/6000 SP with DB2 Parallel Edition

April 1996



IBM

International Technical Support Organization
Poughkeepsie Center

International Technical Support Organization

SG24-4658-00

Migrating and Managing Data on RS/6000 SP with DB2 Parallel Edition

April 1996

Take Note!

Before using this information and the product it supports, be sure to read the general information under “Special Notices” on page xv.

First Edition (April 1996)

This edition applies to Version 1 Release 1 of DB2 Parallel Edition for use with RISC/6000 SP, PSSP Version 2 Release 1.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address given below.

An ITSO Technical Bulletin Evaluation Form for reader's feedback appears facing Chapter 1. If the form has been removed, comments may be addressed to:

IBM Corporation, International Technical Support Organization
Dept. 541 Mail Station P099
522 South Road
Poughkeepsie, New York 12601-5400

When you send information to IBM, you grant IBM a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Abstract

This redbook provides a detailed coverage of data migration inherent in a large commercial environment. It focuses on the different methods of migrating DB2 for MVS data to RISC/6000 SP with DB2 Parallel Edition. It presents a detailed description of the connectivity definitions to link the mainframe system to the RISC/6000 SP over ESCON channels. Information about space considerations, data extraction, propagation are also provided. It also details specific information about the installation of DB2 Parallel Edition on RISC/6000 SP, the pros and cons of the various methods of data migration and the technique for downloading large data greater than 2 GB using PIOFS.

This document was written for IBM systems engineers and customers who will be faced with the challenge of implementing data migration from a variety of sources to the RISC/6000 SP with DB2 Parallel Edition. Some knowledge of RISC/6000 SP, DB2 MVS, DB2 Parallel Edition, data propagation and ESCON is assumed.

(263 pages)

Contents

Figures

Tables

Special Notices

This publication is intended to help IBM system engineers and IBM customer who will be faced with the challenges of migrating existing data from a mainframe system to RISC/6000 SP with DB2 Parallel Edition. The information in this publication is not intended as the specification of any programming interfaces that are provided by the PSSP Package or DB2 Parallel Edition Package. See the PUBLICATIONS section of the IBM Programming Announcement for PSSP Package and DB2 Parallel Edition Package for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, NY 10594 USA.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The information about non-IBM (VENDOR) products in this manual has been supplied by the vendor and IBM assumes no responsibility for its accuracy or completeness. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Any performance data contained in this document was determined in a controlled environment, and therefore, the results that may be obtained in other operating environments may vary significantly. Users of this document should verify the applicable data for their specific environment.

The following document contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples contain the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

Reference to PTF numbers that have not been released through the normal distribution process does not imply general availability. The purpose of including these reference numbers is to alert IBM customers to specific information relative to

the implementation of the PTF when it becomes available to each customer according to the normal IBM PTF distribution process.

You can reproduce a page in this document as a transparency, if that page has the copyright notice on it. The copyright notice must appear on each page being reproduced.

The following terms are trademarks of the International Business Machines Corporation in the United States and/or other countries:

ADSTAR	AIX
APPN	C Set ++
DATABASE 2	DataHub
DataJoiner	DataPropagator
DB2	DB2 Paralle Edition
Distributed Database Connection Services/2	Distributed Relational Database Architecture
DProp	DRDA
Enterprise Systems Architecture/390	Enterprise Systems Connection Architecture
ES/3090	ESCON XDF
ESCON	Hardware Configuration Definition
IBM	Micro Channel
MVS/ESA	MVS/XA
S/390	Scalable POWERparallel Systems
SP	System/390
VTAM	9076 RISC/6000 SP

The following terms are trademarks of other companies:

C-bus is a trademark of Corollary, Inc.

PC Direct is a trademark of Ziff Communications Company and is used by IBM Corporation under license.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

Windows is a trademark of Microsoft Corporation.

XDF.*	Ametron, Inc.
X/Open.*	X/Open Company Limited
SCSI.*	Security Control Systems, Inc.
HYPERchannel.*	Network Systems Corporation
SunOS, SPARCstation, Network File System, NFS.*	Sun Microsystems, Inc.

Other trademarks are trademarks of their respective companies.

Preface

This redbook is intended for IBM systems engineers and customers who plan to implement data migration and management from the host system to RISC/6000 SP with DB2 Parallel Edition. It contains detailed scenarios involved in implementing data extraction, migration and loading, data propagation and space consideration requirements needed in this environment.

How This Document Is Organized

The document is organized as follows:

- Chapter 1, “RISC/6000 SP and DB2 Parallel Edition Overview”

  This chapter provides a rudimentary overview of the DB2 Parallel Edition with RISC/6000 SP. It introduces the fundamental functions, terminologies and some definitions associated with the DB2 Parallel Edition.

- Chapter 2, “Planning Your Parallel Database on the RISC/6000 SP”

  This chapter provides insight into some of the factors that need to be considered when planning to implement your parallel database. It focuses on hardware configuration, space management consideration, the segment manager tool and the various types of external disk devices.

- Chapter 3, “Installation of Hardware and Software for MVS Connectivity”

  This chapter discusses the installation of hardware and software required to establish connectivity between the MVS host system and the RISC/6000 SP. It describes the terminologies commonly used by MVS experts and the one used by the AIX experts and the relationship between them. The ESCON channel and ESCON director, used in the implementation of the connectivity, are also discussed in this chapter.

- Chapter 4, “Implementing DB2 Parallel Edition on the RISC/6000 SP”

  This chapter details the procedures required to install and configure DB2 Parallel Edition in the RISC/6000 SP environment. It exploits the use of the functions that are specific to the RISC/6000 SP. These functions include the automounter daemon (amd) and the supper which is used for file propagation. The amd and the supper are public domain software.

- Chapter 5, “Data Extractions”

  This chapter examines the different methods of data extractions from the host mainframe to RISC/6000 SP with DB2 Parallel Edition. It focuses on the use of the DSNTIAUL tool, which is a program that processes dynamic SQL.

- Chapter 6, “Data Migration and Loading”

  This chapter describes the various methods of migrating and loading data from the host mainframe to RISC/6000 SP with DB2 Parallel Edition. It illustrates with pictorial diagrams the different scenarios of migrating data in this platform. It focuses on the use of CLIO/S, TCP/IP and batchpipes/MVS for data migration and loading. It also discusses the pros and cons of each scenario and the associated business and technical implications.

- Chapter 7, “Implementing Data Propagation from DB2 for MVS to DB2 Parallel Edition”

  This chapter provides the procedures required to implement data propagation between DB2 for MVS and DB2 Parallel Edition using the DataPropagator Relational tool. This tool is used to provide data replication from DB2 for MVS to DB2 Parallel Edition on the RISC/6000 SP. It is focused on the basic steps of installing and configuring the data replicator solution.

- Chapter 8, “DB2 Parallel Edition AutoLoader Tool”

  This chapter discusses the use of a tool called autoloader for extracting and loading data to DB2 Parallel Edition. This tool provides a simplified method of extracting and loading data in one step.

- Chapter 9, “Using the Parallel I/O File System for Data Loading”

  This chapter gives a brief overview of the IBM AIX Parallel I/O File System (PIOFS) and how to exploit its capabilities in a large database environment. It details how to use the PIOFS to download large files that are greater than 2 GB.

Related Publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this document.

- *DB2 Parallel Edition for AIX Administration Guide and Reference*, SC09-1982
- *IBM Client Input Output/Sockets User's Guide and Reference*, CLIO-user-02 (available with the product)
- *Distributed Database Connection Services for AIX*, S20H-4794
- *DATABASE 2 Installing and Using OS/2 Clients Version 2*, S20H-4782
- *Using DataHub for OS/2 Installation and Configuration*, SC26-8366
- *An Introduction to DataPropagator Relational*, GC26-3398-03
- *DataPropagator Relational Guide Release 2,* SC26-3399-03.
- *IBM AIX Parallel I/O File System Installation, Administration, and Use*, SH34-6065
- *IBM Systems Journal Vol. 34, No. 2, 1995 Parallel File Systems for the IBM SP computers*, G321-0120
- *AIX Version 4.1 Enterprise System Connection Adapter: User's Guide and Service Information,* SC31-8197
- *AIX Version 4.1 Block Multiplexor Channel Adapter: User's Guide and Service Information,* SC31-8196

International Technical Support Organization Publications

- *RISC/6000 370 Channel Support ESCON and Block Multiplexer,* SG24-4589
- *DB2 Parallel Edition for AIX/6000 Concepts and Facilities,* SG24-2514
- *Using ADSM to Back Up Databases*, SG24-4335-01

A complete list of International Technical Support Organization publications, known as redbooks, with a brief description of each, may be found in:

International Technical Support Organization Bibliography of Redbooks, GG24-3070.

How Customers Can Get Redbooks and Other ITSO Deliverables

Customers may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **IBMLINK**

  Registered customers have access to PUBORDER to order hardcopy, to REDBOOKS disk to obtain BookManager BOOKs

- **IBM Bookshop** — send orders to:

  usib6fpl@ibmmail.com (United States)
  bookshop@dk.ibm.com (Outside United States)

- **Telephone orders**

  1-800-879-2755 (United States)
  354-9408 (Australia)
  359-2-731076 (Bulgaria)
  42-2-67106-250 (Czech Republic)
  593-2-5651-00 (Ecuador)
  03-69-78901 (Israel)
  905-627-1163 (Mexico)
  064-4-57659-36 (New Zealand)
  027-011-320-9299 (South Africa)
  0256-478166 (United Kingdom)
  32-2-225-3738 (Belgium)
  1-800-IBM-CALL (Canada)
  45-934545 (Denmark)
  01805-5090 (Germany)
  0462-73-6669 (Japan)
  31-20513-5100 (The Netherlands)
  507-639977 (Panama)

- **Mail Orders** — send orders to:

  IBM Publications
  P.O. Box 9046
  Boulder, CO 80301-9191
  USA

  IBM Direct Services
  Sortemosevej 21,
  3450 Allerod
  Denmark

- **Fax** — send orders to:

  1-800-445-9269 (United States)
  32-2-225-3478 (Belgium)
  905-316-7210 (Canada)
  593-2-5651-45 (Ecuador)
  03-69-59985 (Israel)
  31-20513-3296 (The Netherlands)
  507-693604 (Panama)
  0256-843173 (United Kingdom)
  359-2-730235 (Bulgaria)
  42-2-67106-402 (Czech Republic)
  07032-15-3300 (Germany)
  0462-73-7313 (Japan)
  064-4-57659-16 (New Zealand)
  027-011-320-9113 (South Africa)

- **1-800-IBM-4FAX (United States only)** — ask for:

  Index # 4421 Abstracts of new redbooks
  Index # 4422 IBM redbooks
  Index # 4420 Redbooks for last six months

- **Direct Services**

  Send note to `softwareshop@vnet.ibm.com`

- **Redbooks Home Page on the World Wide Web**

  http://www.redbooks.ibm.com/redbooks

- **E-mail (Internet)**

  Send note to redbook@vnet.ibm.com

- **Internet Listserver**

  With an Internet E-mail address, anyone can subscribe to an IBM Announcement Listserver. To initiate the service, send an E-mail note to announce@webster.ibmlink.ibm.com with the keyword subscribe in the body of the note (leave the subject line blank). A category form and detailed instructions will be sent to you.

How IBM Employees Can Get Redbooks and ITSO Deliverables

Employees may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **PUBORDER** — to order hardcopies in United States
- **GOPHER link to the Internet**

  Type GOPHER
  Select IBM GOPHER SERVERS
  Select ITSO GOPHER SERVER for Redbooks

- **Tools disks**

  To get LIST3820s of redbooks, type one of the following commands:

  ```
  TOOLS SENDTO EHONE4 TOOLS2 REDPRINT GET GG24xxxx PACKAGE
  TOOLS SENDTO CANVM2 TOOLS REDPRINT GET GG24xxxx PACKAGE (Canadian users only)
  ```

  To get lists of redbooks:

  ```
  TOOLS SENDTO WTSCPOK TOOLS REDBOOKS GET REDBOOKS CATALOG
  TOOLS SENDTO USDIST MKTTOOLS MKTTOOLS GET ITSOCAT TXT
  TOOLS SENDTO USDIST MKTTOOLS MKTTOOLS GET LISTSERV PACKAGE
  ```

  To register for information on workshops, residencies, and redbooks:

  ```
  TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ITSOREGI 1996
  ```

  For a list of product area specialists in the ITSO:

  ```
  TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ORGCARD PACKAGE
  ```

- **Redbooks Home Page on the World Wide Web**

  http://w3.itso.ibm.com/redbooks/redbooks.html

- **ITSO4USA category on INEWS**
- **IBM Bookshop** — send orders to:

  USIB6FPL at IBMMAIL or DKIBMBSH at IBMMAIL

- **Internet Listserver**

  With an Internet E-mail address, anyone can subscribe to an IBM Announcement Listserver. To initiate the service, send an E-mail note to announce@webster.ibmlink.ibm.com with the keyword subscribe in the body of

the note (leave the subject line blank). A category form and detailed instructions will be sent to you.

Acknowledgments

This project was designed and managed by:

Endy Chiakpo
International Technical Support Organization, Poughkeepsie Center

The authors of this document are:

Peter Abrahams
IBM United Kingdom

Alain Benhaim
IBM France

Endy Chiakpo
ITSO Poughkeepsie Center

Otto Goerlich
IBM Germany

Adriana Gonzalez Garcia
IBM Mexico

Erick Llaguno Alanis
IBM Chile

Dr. Rodolphe Michel
IBM UK (ISC Hursley)

Rolf Stadler
IBM Switzerland

This publication is the result of a residency conducted at the International Technical Support Organization, Poughkeepsie Center.

Thanks to the following people for the invaluable advice and guidance provided in the production of this document:

Michel Perraud
International Technical Support Organization, Poughkeepsie Center

Ken Trowell
IBM Australia

IBM POWERparallel Systems Laboratory:
Joe Casey
Iwao Hatanaka
Brian O'Leary

IBM Toronto Laboratory:
Scott Bailey

Kenneth Chen
Roger Zheng

IBM Dallas System Center:
Dave Hewett
Jack Woodson

IBM TJ Watson Research Center:
Hui-I Hsiao

IBM EMEA PSSC Montpellier France:
Eric Le Bail

Chapter 1. RISC/6000 SP and DB2 Parallel Edition Overview

The IBM RISC/6000 SP provides a robust and reliable platform for customers in both commercial and scientific environments to process their applications. The robustness of the RISC/6000 SP and the high industrial strength that it provides through a high degree of scalability and parallelism makes it very suitable for parallel databases. For more information on RISC/6000 SP refer to the redbook captioned *Intro to Parallel Processing & Scalable POWERparallel,* GG24-4344.

1.1 Parallel Databases

The concept of processing data in a parallel environment is fast becoming the way some businesses and commercial customers are turning to as the solution for today and future data processing challenges. This is due to the rate of increase in database size and also because the response time requirements have outpaced advancements in processor and mass storage technology. One method of fulfilling the increasing demand for high processing power and input/output bandwidth in complex database applications is to have a number of processors, loosely or tightly coupled, serving parallel database requests simultaneously.

Large scale parallel processing technology has made giant strides in the past decade, and there is no doubt that it has established a place for itself. However, the current generation of massively parallel processor systems, in particular, IBM's RISC/6000 Scalable Parallel class of systems, are much more robust and easy to use.

One of the main enablers for commercial applications is Database Management Systems (DBMS). Thus, a parallel DBMS is a natural step. Several businesses and industries are investing in Decision Support applications in order to understand various sales and purchase trends. These applications pose complex questions (queries) against large sets of data in order to gain an insight into the trends. Single system (or Serial) DBMSs cannot handle the capacity and the complexity requirements of these applications. Besides decision support, there are other new application classes such as Data Mining, Electronic Libraries, and Multimedia that require either large capacity or the ability to handle complexity. All these applications require parallel DBMS software.

1.2 DB2 Parallel Edition

DB2 Parallel Edition is one of the first IBM answers to the market place, to address the high demand by the customer for a much faster and reliable method of processing data in the parallel environment.

DB2 Parallel Edition is a parallel database software solution that can execute on AIX 3.2.5 or later releases. Its Shared-Nothing (SN) architecture model and Function Shipping execution model provide very important assets, namely scalability and capacity. This system can easily accommodate databases with hundreds of gigabytes of data. Likewise, the system model enables databases to be easily scaled with the addition of more system CPU and disk resources. DB2 Parallel Edition has been architected and implemented to provide the best query processing performance. The query optimization technology considers a variety of parallel execution strategies for different operations and queries and uses the

lowest cost in order to choose the best possible execution strategy. The execution time environment is optimized to reduce process overhead, synchronization overhead, and data transfer overhead.

1.2.1 DB2 Parallel Edition Utilities

DB2 Parallel Edition provides a variety of utilities to manage the parallel database system. Some of the important utilities are described as follows: The Load utility allows bulk loading of database tables from flat files. To support applications requiring very large database sizes (100's of GB and higher), DB2 Parallel Edition provides efficient ways of loading large volumes of data into the database. Data can be loaded in parallel into a single table by invoking the load utility at each of the nodes that contains a table partition for the given table. Typically, the input data is stored in a single flat file. Application programming interfaces (APIs) provided with the database system can be used to partition an input file into multiple files, one per table partition. The partitioned files can then be loaded in parallel. In addition, at each node, the load utility reads the input data file and directly creates data pages in the internal format used by the database engine.

1.2.1.1 Creating Databases

Normally, issuing the Create Database command ensures that the database is defined across all the nodes that are currently in the system. However, there are situations in which one may wish to create and drop the database only at a single node. For example, the Add Node command (described above) implicitly performs a create database at node operation for each existing database.

1.2.1.2 Table Reorganization

As a result of the insert, delete, and update activity, the physical layout of the database tables may change. Insertions may result in the creation of overflow data blocks and, as a result, the disk pages containing data belonging to the table may no longer be stored contiguously. On the other hand, deletions may create gaps in disk pages, thereby resulting in an inefficient utilization of disk space. If a table is partitioned across a set of nodes, insert/delete activity may also result in table partitions at some nodes having more data than those at other nodes, thus creating a skew in data distribution. Also, in many decision support applications, the database size increases with time. Thus, it may be necessary to increase the degree of declustering of a table in order to accommodate the additional data. Finally, even if the size of the database remains the same, the workload may change, thereby requiring a change in data placement. In all of the above situations, data reorganization utilities can be used to manage the physical storage of the table.

The reorg command can be used to recluster table data according to the ordering sequence defined by an index associated with the table. This can enhance performance by a significant factor when the table is accessed through the index that was used for the reorg versus going over the same index without reorg.

The Reorg operation executes in parallel across all the nodes that contain a table partition for a given table. The file in which the database table is stored is reorganized by creating a new file without any page gaps and overflow blocks. If the operation completes successfully on some nodes but not on others, then the table partitions remain successfully reorganized at the nodes where reorg succeeded.

The reorg table execution in a node is done independently with respect to the other nodes. However, the reorg operation may be time consuming and undoing it may be even more expensive. In addition, consider the case when reorg succeeds on, say, 60 nodes but fails on 1. It is more beneficial not to undo the operation. In this case, the operation returns an error message but is not undone since there is no penalty if some partitions are reorganized while others are not. On the other hand, the nodes where the partitions were reorganized would benefit from the resulting file compaction.

1.2.1.3 Data Redistribution

The partitioning strategy used to partition tables may, in some situations, cause a skew in the distribution of data across nodes. This can be due to a variety of factors including the distribution of attribute values in a relation and the nature of the partitioning strategy itself. At the initial placement time, it is possible to analyze the distribution of input attribute values and obtain a data placement that minimizes skew. However, data skew may be reintroduced over the database lifetime due to insertion and deletion of data. DB2 Parallel Edition provides the redistribute Nodegroup operation to redistribute the data across the nodegroup to control the skew.

For a given nodegroup, the redistribute operation considers the 4K partitions in the partitioning map and determines how the partitions should be moved in order to obtain an even distribution of data across the nodes of the nodegroup. The default assumption is that the nodes are equally represented across the 4K partitions, thus, if the partitions are evenly distributed among the set of nodes, then the data is also assumed to be evenly distributed across the nodes. The user may override this default assumption by providing a distribution file which assigns a weight to each of the 4K partitions. In this case, the redistribute operation will attempt to redistribute data among nodes such that the sum of the weights at each node is approximately the same.

If a nodegroup contains several tables, redistributing only one table and not the others could result in a loss of collocation among the tables. To avoid this, the redistribute command operates at the nodegroup level and each table is in turn redistributed. If a redistribute operation does not complete successfully, it is likely that some tables in the nodegroup have been redistributed while others have not. In this case, the operation can be completed by issuing the redistribute command with the continue option. It is also possible to issue the redistribute command with a rollback option, in order to undo the effects of the failed redistribute. The redistribute Nodegroup command is an on-line operation which locks only the table that is currently being redistributed. All other tables in the nodegroup are normally accessible. The current data distribution across partitions and nodes can be determined using two new SQL scalar functions, PARTITION and NODENUMBER. These functions return the partition number and node number to which a given row in a table is mapped.

1.3 Scalability

DB2 Parallel Edition supports scalability by allowing incremental addition of nodes to the shared-nothing parallel system. Thus, a user can start with a system configuration that is sufficient to handle current storage and performance requirements and add new nodes as the size of the database grows. New nodes can be added to increase storage capacity as well as performance. The command,

db2start add node, allows users to add nodes to the parallel database system configuration. The add node routine creates a database partition for each database in the database system on every node that is added. Since nodegroup allow for partial declustering of tables, the set of all tables for a given database may reside only on a subset of the nodes in the system. However, an application can connect to any database from any node in the system, regardless of whether that node contains data pertaining to that database.

The Drop Node command can be used to remove an existing node from the database configuration. However, if the node to be dropped contains any data belonging to any of the currently defined databases, then the node is not dropped. The redistribute nodegroup command must be used to remove data from this node before it can be dropped.

1.4 Performance Monitoring and Configuration Management

Database monitoring tools allow users to identify performance bottlenecks and take appropriate action to relieve the bottlenecks. DB2 Parallel Edition provides a database monitoring facility that allows system administrators users to gather data on different types of resource consumption. This data is collected at each node and can be used to identify bottlenecks at individual nodes. To obtain a global picture of the performance of the entire system, it is necessary to combine performance monitoring data across all nodes.

The database manager provides several configuration parameters at the database manager and individual database levels, that can be used to tune the performance of each database and the database manager as a whole. For example, users can control the size of buffers, maximum number of processes, size of log files, etc. These parameters can be set independently at each node thereby allowing users to tune the performance of each individual node.

1.4.1 Backup and Restore Considerations

The degree of parallelism achieved during backup and restore of a database is determined by the number of backup devices available. The DB2 Parallel Edition backup and restore design allows each node in the system to be backed up independently. Data from several nodes can be backed up simultaneously, if multiple backup devices are available. The backup utility creates a backup image of the entire database partition resident at a given node.

At restore time, it is necessary to ensure that the database partition that is being restored is in a consistent state with respect to the rest of the nodes in the system. This can be achieved using any of the three methods below:

1. By restoring all nodes in the system using backup images that are known to be consistent.
2. By restoring all nodes and rolling forward logs to a point in time where the database state is consistent across all nodes.
3. By restoring a subset of nodes and rollforward to end of log.

For more details, on DB2 Parallel Edition backup and restore considerations, you can refer to the *Backup, Recovery and Availability with DB2 Parallel Edition on RISC/6000 SP,* SG24-4695.

Chapter 2. Planning Your Parallel Database on the RISC/6000 SP

When you plan your parallel database, the following are a number of elements that you need to consider:

- The configuration you want. See 2.1, “RISC/6000 SP Hardware Configuration.”
- The nodegroups that you want and the placement of data within the nodegroups. See 2.9, “Nodegroups” on page 31.
- The amount of disk space. See 2.4, “Calculating Disk Space” on page 16.
- The balance of space consumption across segments, and the number of segment subdirectories. See 2.5.4, “Working with Multiple Segments” on page 22.
- The amount of real memory. See Memory Usage in *DB2 Parallel Edition for AIX, Administration Guide and Reference*, SC09-1982-00.
- Performance expectations. See *DB2 Parallel Edition for AIX, Administration Guide and Reference*, SC09-1982-00.
- The different methods of how to populate your database with data. See Chapter 6, “Data Migration and Loading” on page 101.
- Backup/Recovery considerations. Refer to the redbook captioned *Backup, Recovery and Availability with DB2 Parallel Edition on RISC/6000 SP*, SG24-4695-00.

2.1 RISC/6000 SP Hardware Configuration

The following are several things to be considered when you plan your hardware configuration.

- The number of RISC/6000 SP nodes to be used for your database
- The type of RISC/6000 SP nodes, either thin or wide nodes
- The type of disks and the amount of disk space to store your database.
- Whether or not resources, like disks, should be shared for high availability
- Network interfaces within the RISC/6000 SP itself and to externalized networks.

2.1.1 RISC/6000 SP Configuration

The number of nodes in a standard RISC/6000 SP configuration ranges from four (maybe two) to 512, which may be contained in multiple frames. A frame has eight drawers and may contain up to 16 thin nodes. One drawer may hold either one wide node or two thin nodes. In addition, it provides a High Performance Switch (HPS) to connect the nodes. DB2 Parallel Edition uses the HPS for node-to-node communications. With the high bandwidth of the HPS, you can scale your database to a much larger number of nodes than is possible in a LAN configuration. DB2 Parallel Edition supports as many nodes in a RISC/6000 SP configuration as the RISC/6000 SP itself.

For more information about RISC/6000 SP, refer to *Into Parallel Processing & Scalable POWERparallel*, GG24-4344-00.

The decision of how many thin nodes and wide nodes you need depends on the size of your database, the response time requirements, the amount of real memory that is required per node, and the number of micro channel slots needed per node. As a general rule, the larger the configuration you are using, the more homogenous the nodes in the RISC/6000 SP configuration should be.

Thin Nodes: The RISC/6000 SP thin node is typically configured as a *database worker* node. External disks are attached to it to hold the database data. It can hold up to 512 MB of memory, up to 9 GB of internal disk space, and has four available MCA slots.

Wide Nodes: The RISC/6000 SP wide node is more often used as a server node. Some possible functions to run on a wide node are as follows:

DB2 Parallel Edition coordinator node
ADSM server
network server
tape server
file server

It provides up to 2 GB of memory, up to 18 GB of internal disk space and has seven available MCA slots.

Figure 1 on page 7 illustrates the kind of flexibility possible with attaching external devices and using the nodes of the RISC/6000 SP for purposes other then just as database nodes.

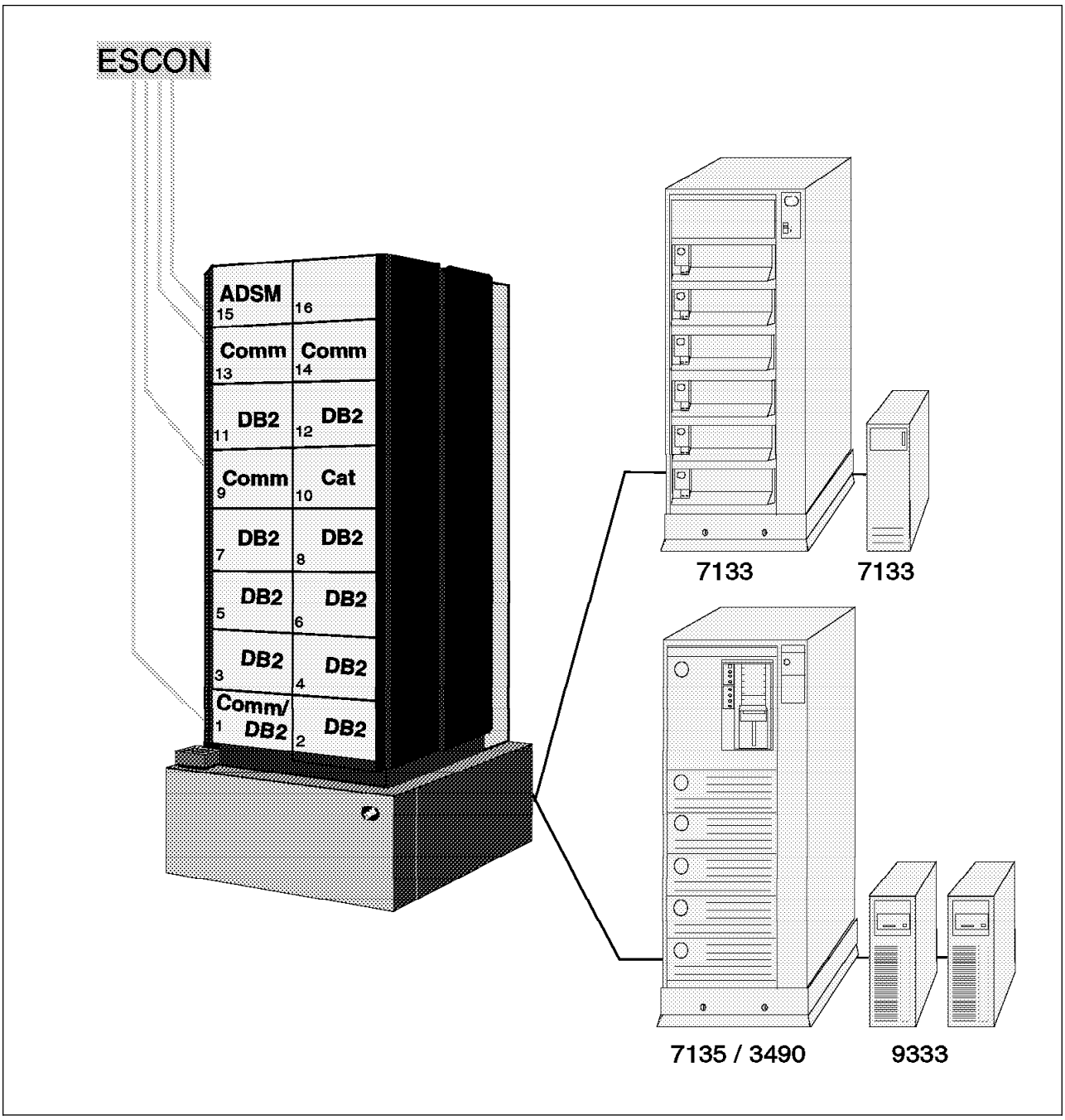


Figure 1. RISC/6000 SP with Thin Nodes, External Attached Disks and Tape Units

2.2 Disk and Disk Space Considerations

Each database node within your RISC/6000 SP configuration requires disk space for the database data on that node. DB2 Parallel Edition data is stored in AIX JFS file systems defined within the disks. On the RISC/6000 SP, the choice of disks can be first split into two parts, internal disks and external disks. For performance reasons, it recommended to keep the data on each node within 20 GB or lower.

2.2.1 Internal Disks

The current maximum size of an internal SCSI disk is 4.5 GB. A thin node may have two internal disks and therefore 9 GB internal disk space. A wide node may have up to four disks which gives 18 GB. Space will be required on these disks to hold the AIX operating system itself, paging space, and other user data. In addition, you may choose to mirror logical volumes to improve availability, but this will reduce your disk space available for DB2 Parallel Edition data even more. So, internal disks are very useful to store temporary data, like the files to be loaded into the DB2 Parallel Edition database. It is not recommended that you have a production DB2 Parallel Edition database on internal disks.

2.2.2 External Disks

External disks may be chosen to improve data transfer time to or from disks, to increase the amount of disk space attached to a RISC/6000 SP node and to increase the availability.

Speed advantages may be gained in the following different ways:

- Serial disks may be used. Data transmission to and from serial disks can be faster than SCSI for some specific types of applications such as those requiring simultaneous write operations.
- A larger number of smaller disks may be used. This will reduce the disk search time for an item of data. More than one disk may return different parts of the same answer sets at the same time, which introduces I/O parallelism.
- More than one adapter may be used. Two SCSI adapters with disks attached will provide two paths to data. This will allow up to twice as much data per second to be transferred.

Three types of external disks can be attached to a RISC/6000 SP node:

1. RAIDiant arrays
2. 7133 SSA disks
3. SCSI disks

2.2.2.1 IBM RAIDiant Array

You can use a RAIDiant array (RAID) subsystem to enhance availability. IBM products of this type are the IBM 7135, IBM 7137 and IBM 3514. These subsystems use the Redundant Array of Independent Disks (RAID) technology to provide high availability and support RAID levels 0, 1, 3, and 5. A typical RAID subsystem has two disk controllers and is connected to two RISC/6000 SP nodes. If a RISC/6000 SP node or a disk controller fails, the disks can be accessed from the other RISC/6000 SP node using the other controller.

Using RAID is a less expensive solution than mirroring, because with mirroring you need twice as much disk space than is required by the database. If performance is the most important point, mirroring is the better solution.

IBM 7135: The IBM 7135 RAIDiant Array is a disk storage subsystem ideally suited for the RISC/6000 SP and DB2 Parallel Edition, providing a large storage capacity and high data availability. The 7135 RAIDiant Array is offered in a range of models and features with storage capacities ranging from 2.6 GB to 135 GB. It attaches to a RISC/6000 SP node via a differential SCSI link.

7135 RAIDiant Array Model 110 This is a high availability model that offers 2.6 GB to 135 GB of disk storage. The 7135 Model 110 can be configured to implement up to three different high availability RAID levels (RAID-1, RAID-3 and RAID-5) as well as disk striping (often called RAID-0) through software running under AIX on the RISC/6000 SP nodes.

7135 RAIDiant Array Model 210 This is a high availability model that offers 2.6 GB to 135 GB of disk storage. The 7135 Model 210 can be configured to implement up to two different high availability RAID levels (RAID-1 and RAID-5) as well as disk striping (often called RAID-0) through software running under AIX 4.1.3 on the RISC/6000 SP nodes. Each Model 210 RAIDiant Array controller includes a 2 MB cache as standard.

- The redundant design and RAID support in Models 110 and 210 provide highly available disk storage for the RISC/6000 SP:
  - Multiple concurrent RAID modes are supported.

    This offers the interesting possibility of putting the DB2 Parallel Edition data on disks configured in RAID-5 mode (to get data protection) and the DB2 Parallel Edition log files on disks configured in RAID-1 mode (to get data protection and better performance).
  - RAID modes are selected and configured via the software support running under AIX.
  - Concurrent maintenance is possible for all active elements, including customer replacement of failed disk drives.
- Fault tolerant configurations are possible for the Model 110 and 210 with the dual-array controller feature. This type of configuration is the one most commonly used by RISC/6000 SP and DB2 Parallel Edition customers because it provides the highest availability characteristics.

Figure 2 on page 10 shows an example how a 7135 with dual controller attaches to two RISC/6000 SP nodes.

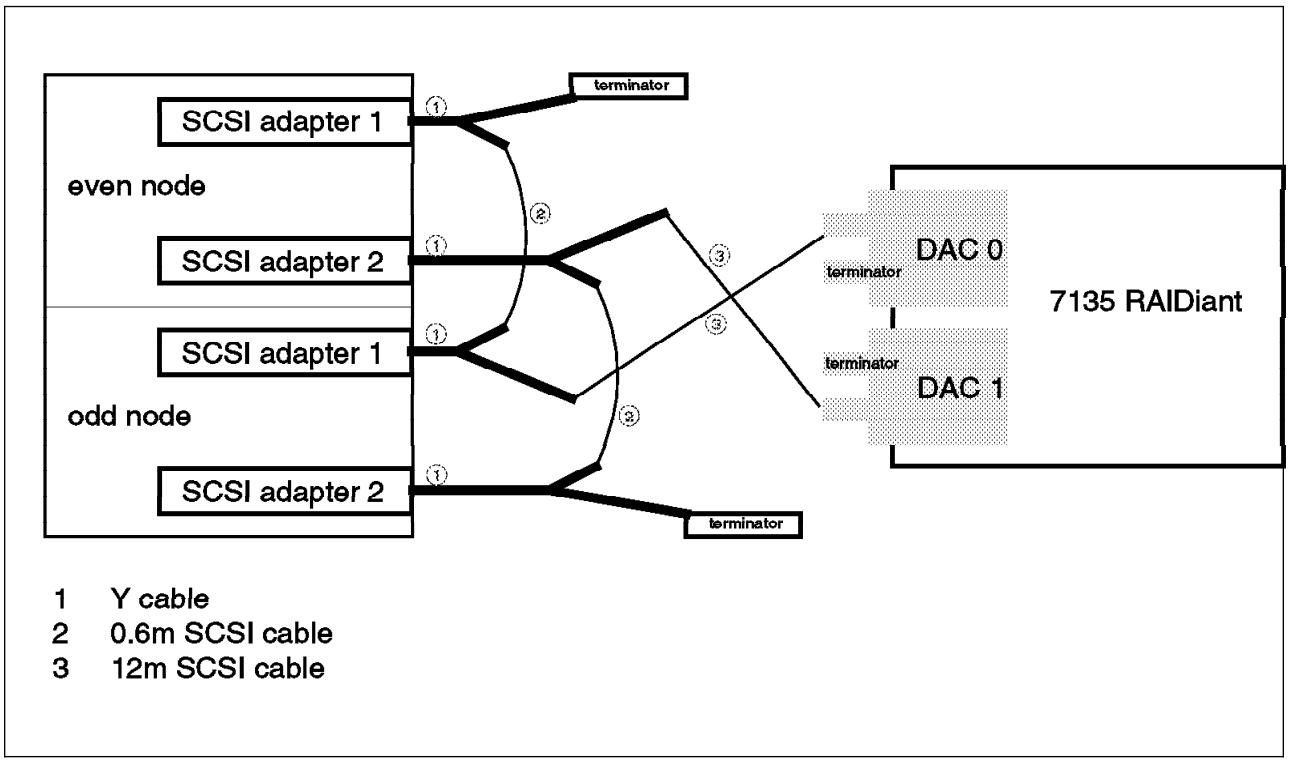


Figure 2. Two RISC/6000 SP Nodes Attached to a IBM 7135 Disk Subsystem

Figure 2 illustrates how highly available configurations can be built using the dual controller option and attaching the subsystem to two RISC/6000 SP nodes. This is akin to the configuration possible with the RISC/6000 SP PowerQuery solution.

IBM 7137: The IBM 7137 is a RAID subsystem and is attached to the RISC/6000 SP nodes through a fast and wide (16 bit) SCSI-2 differential interface. The 7137 incorporates disk drive modules with a capacity of 1.0 GB, 2.2 GB, or 4.5 GB each. The base configuration contains three disk drives and can be expanded to a maximum of eight drives. The 7137 Models 412, 413 and 414 are deskside units and the models 512, 513 and 514 are rack mountable.

The 7137 Disk Array Subsystem offers two RAID levels, RAID-0 and RAID-5. With RAID-5, the 7137 has a capacity range from 1.97 GB to 29.36 GB.

The 1 MB, or optional 4 MB, write cache enhances the internal performance of the 7137 by combining and optimizing write operations.

2.2.2.2 Extended Disk Controllers

Some disk subsystems (such as the 7135 Model 210 and the 7137) have memory cache, which can improve the performance of read and write operations. Because DB2 Parallel Edition workloads are usually CPU intensive, their performance may not benefit from the cache. In some environments, however, a write cache can enhance the performance of update activities.

2.2.2.3 IBM 7135 Model 210 Fast Load Feature

In order to provide customers the highest possible performance during database load operations, the IBM 7135 Model 210 controller provides a fast load feature that offers volatile write caching performance during database load time in which user data can be reloaded in the event of a non-recoverable failure, such as power loss or controller failure.

This option should only be enabled during write updates that are easily reclaimed in the event of any failure. Therefore, this option should only be used when loading data from a medium that allows a complete retry of the load process.

Who Should Use Fast Load

Given that fast load is volatile and does not provide full redundancy, it should not be used in high availability situations. However, there are many other arenas where you can take advantage of performance of fast load.

Database Loads

Database loads and restoration typically involve reading data from tape or across a network onto the IBM 7135. If a hardware problem occurs and data is lost, the database load process might be restarted after the hardware problem is corrected. Once the load operation is complete, the user can disable fast load and resume redundant operations.

How to Enable the Fast Load Feature

The fast load feature is configurable at the LUN (Logical Unit Number) level. A LUN is equivalent to an AIX hard disk. This makes it possible to have LUNs within your RAID-5 configuration running with write cache on and other LUNs with write caching off. For example, some database operations read from a critical database in order to generate new tables in another database. For this type of operation you can disable fast load on the LUNs which house the read database and enable fast load on the LUNs that contain the new tables. If a failure were to occur on the new tables that result in cache loss, you can restart the operation once the problem has been corrected.

To enable fast load, execute the following command:

```
chdev -l hdisknn -a write_cache=yes
```

To disable fast load, execute the following command:

```
chdev -l hdisknn -a write_cache=no
```

This is settable during run-time. In addition, disabling the fast load feature will force the controller to flush its cache to the disk drives.

Note: At the time of this publication, there is a known problem with the use of Chdev command sequence with fast load feature. To bypass this problem, you must follow these steps:

- Stop the applications and database
- Unmount the 7135 file systems for the target LUN
- Varyoff the 7135 Volume Groups for the target LUN
- Issue the Chdev Command

- Varyon the 7135 Volume Groups for the target LUN
- Mount the 7135 file systems for the target LUN
- Start the database and applications

For more details on this problem, refer to PMR 5717x,035,000.

2.2.2.4 IBM 7133 SSA Subsystem

The IBM 7133 Serial Storage Architecture (SSA) Disk Subsystem Models 010 and 500 provide high-performance (SSA provides better performance than SCSI disks), high-capacity storage. The rack-mountable IBM 7133 SSA Disk Subsystem Model 010 is designed for integration into a standard 19" rack which can be attached to the RISC/6000 SP. The 7133 is also available in a free standing deskside tower unit (the Model 500).

The 7133 Models 010 and 500 are functionally equivalent and can be populated with three different disk drive modules - 1.1 GB, 2.2 GB, and 4.5 GB Ultrastar drives. Each disk drive is mounted in an auto-docking carrier for easy replacement. Features are provided to support from 4 to 16 disk drive modules, yielding a maximum capacity of up to 72 GB.

Both models are equipped with redundant power/cooling units to improve high availability.

Figure 3 on page 13 shows the deskside model of the 7133 as well as the rack-mounted model.

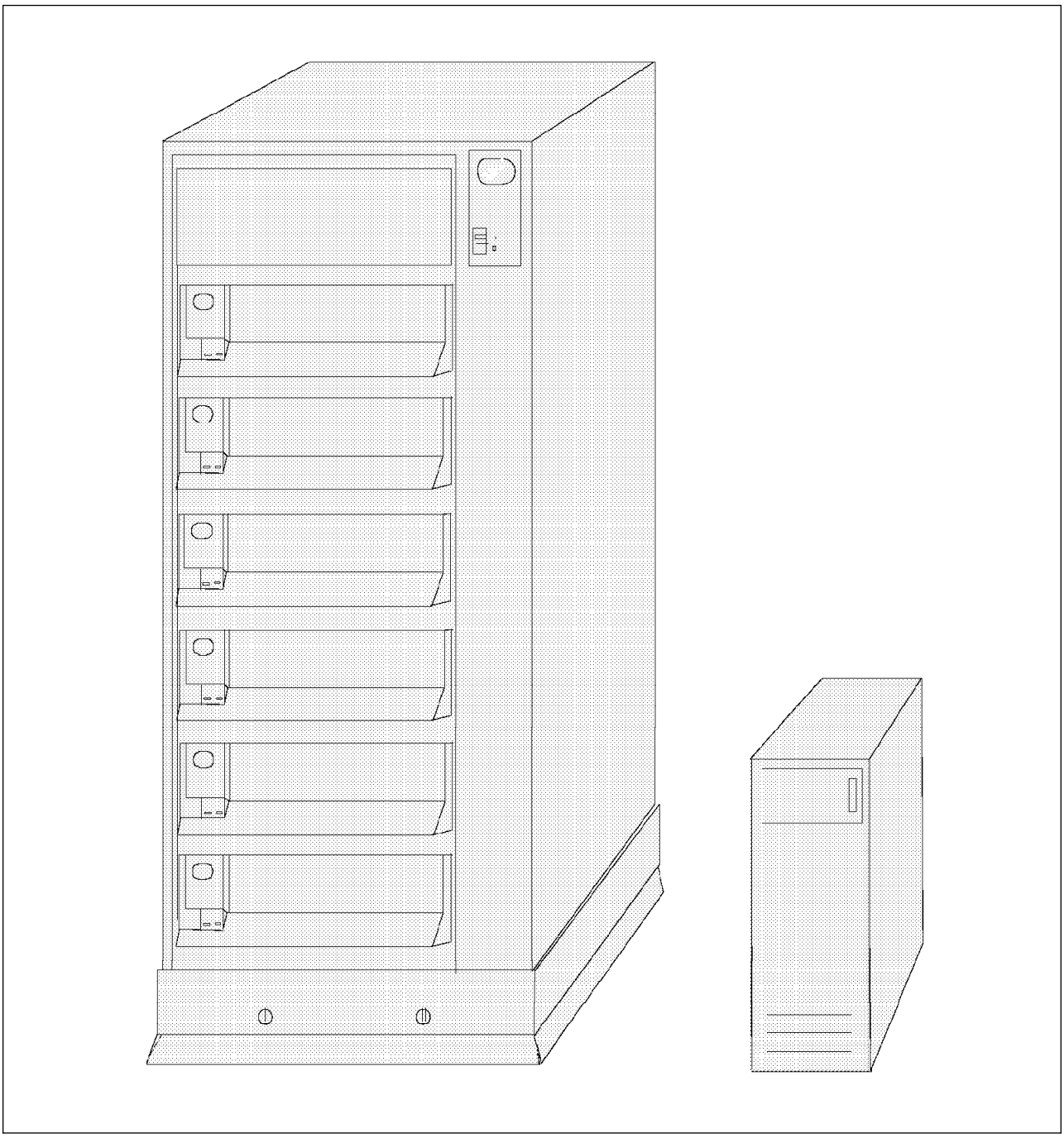


Figure 3. IBM 7133 SSA Disk Subsystem

The IBM 7133 SSA Disk Subsystem is attached to the RISC/6000 SP nodes via the IBM SSA 4-port Adapter. Unlike SCSI bus configurations, SSA devices are configured in loops with daisy-chained connections between devices and do not require bus arbitration. This enables multiple concurrent operations to occur in separate sections of the loop, resulting in higher overall throughput. This is an exclusive capability of SSA and is not possible on a SCSI bus, since the entire bus carries a single operation. A 7133 drawer or tower can support up to four loops of one to four disks.

The 7133 provides superior performance of up to 3000 I/O operations per second per adapter and a very large bandwidth. This is accomplished via a full duplex,

frame multiplexed, serial link, which allows for simultaneous read and write operations at 20 MB/sec in each direction on the loop. Figure 4 on page 14 shows the attachment of a 16 drive 7133 SSA unit to two RISC/6000 SP nodes.

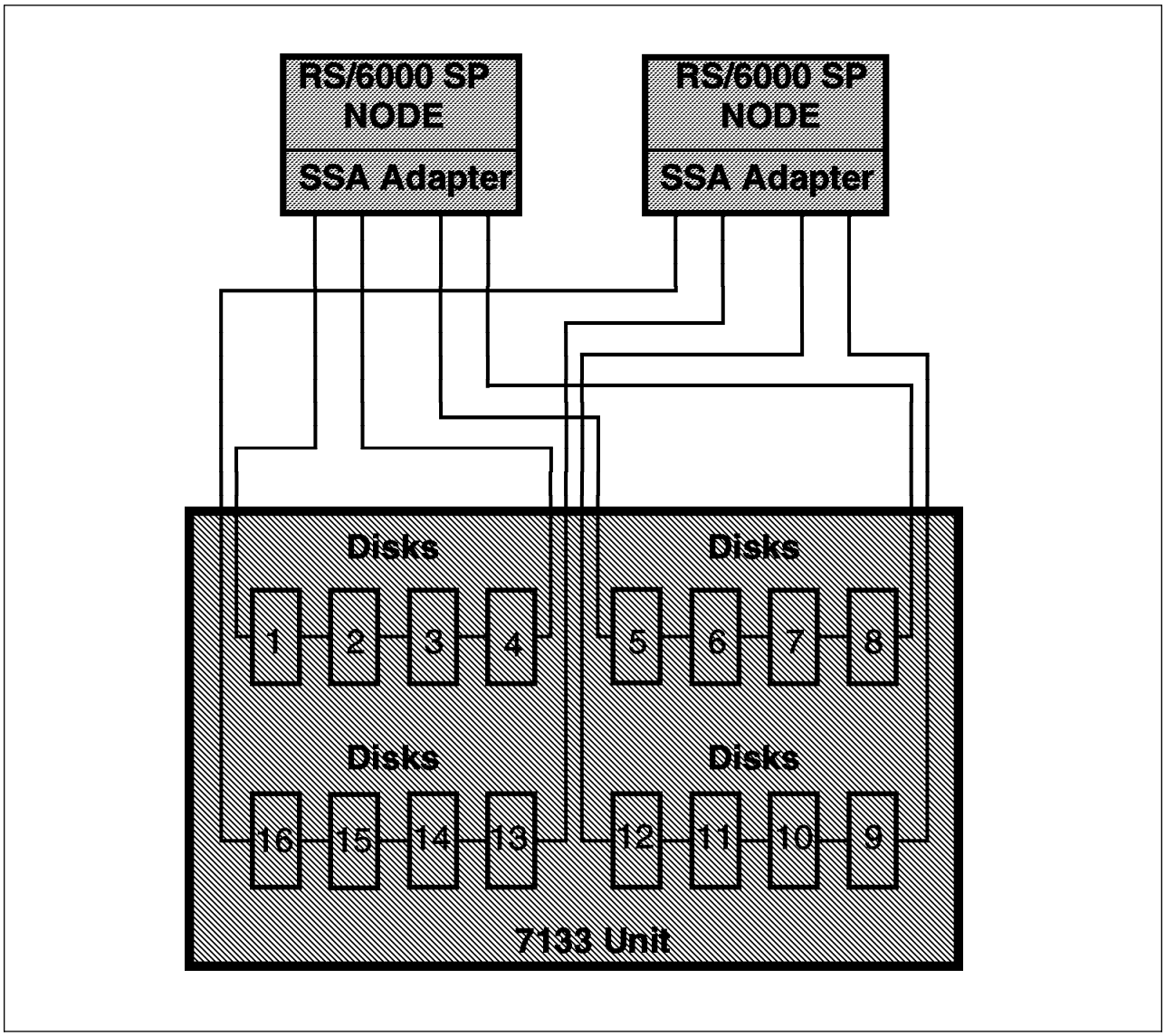


Figure 4. One 7133 SSA Disk Subsystem with 16 Disk Drive Modules

Note

If you use the 7133, you should consider mirroring disks for availability reasons.

2.2.2.5 SCSI - SCSI-2 Attached Disk Subsystems

SCSI attached disks include a number of disk subsystems, like the IBM 7131 Model 105 and the IBM 7134 Model 010.

IBM 7134 Model 010 and IBM 7135 Model 010: The IBM 7134 Model 010 is a rack mountable disk subsystem and attaches to the RISC/6000 SP nodes using the SCSI-2 Differential Fast/Wide interface. The 7134 may be configured with up to 16 disk drives of either 2.0 GB, 2.2 GB or 4.5 GB which gives a total capacity of 72

GB per subsystem. High availability requirements can be satisfied with AIX system mirroring.

The 7135 Model 010 is a non-RAID disk subsystem. It may hold up to twelve disk drives. The maximum capacity, with the 4.5 GB disk drives, is 54 GB. Every six disk drives require a SCSI-2 differential or SCSI-2 differential fast/wide adapter in the attaching RISC/6000 SP node.

If you intend to use this unit, disk mirroring should be considered for high availability reasons.

2.2.2.6 Optical Storage Devices

DB2 Parallel Edition does not have any specific restrictions regarding storage devices, except that the device be read/write, and not read-only. For example, DB2 Parallel Edition can use data that is attached to an optical storage device, such as the IBM 3995. The device can be accessed through a mechanism such as the AIX Journaled File System (JFS). The IBM 3995 optical storage device should be considered as a device where you have high capacity storage requirements but lower performance expectations.

2.3 Tape Libraries and Tape Devices

The usage of a tape library or one or more tape units may be considered for database backup and restore purposes. Database backup and restore functions can be managed by using the ADSTAR Distributed Storage Manager (ADSM). ADSM is a client/server product that provides save and restore functions on the RISC/6000 SP.

2.3.1 IBM 3590 Tape Subsystem with the Magstar Tape Drive

The 3590 tape subsystem attaches to the RISC/6000 SP nodes using the SCSI-2 Differential Interface or the SCSI-2 Differential Fast/Wide Interface.

The IBM 3590 tape subsystem is available in two models:

- The Model B11 is rack-mounted and incorporates a 10-cartridge Automatic Cartridge Facility (ACF) for high-capacity unattended operation.
- The Model B1A has no ACF and is designed to be incorporated into the IBM 3494 tape library.

Magstar's 9 MB/sec maximum uncompacted data rate can reduce backup and recovery times significantly. The 5 meters/sec high-speed search provides rapid access to stored data. The cartridges have a capacity of 10 GB and up to 30 GB can be stored on one cartridge using the new LZ1 compression technique. Larger cartridge capacity means fewer cartridges are required and automation equipment requirements can be reduced.

The Magstar drive offers random access to over 300 GB (compacted) utilizing the 10-cartridge ACF function. This allows the usage of the 3590 tape subsystem as a low-cost mini tape library. ADSM for AIX takes advantage of the ACF in random mode and Magstar's high-speed search facility to position to the data fast. In smaller database environments, this configuration substitutes for a tape library to enable truly scalable automation.

2.3.2 IBM 3494 Tape Library Dataserver

The IBM 3494 Tape Library Dataserver is a high-performance automated tape library for IBM 3590 Magstar and 3490E tape drives, providing a cost-effective, high reliability, and space-efficient tape automation solution for large DB2 Parallel Edition database environments.

The 3494 is a modular system that contains a cartridge accessor, a library manager, Magstar or 3490E tape drives, and space for tape cartridges. The 3494 tape library is expandable from one to 30 tape transports and from 210 to 3,040 cartridges which gives a total storage capacity of 91 TB (using the Magstar tape drive with compaction). From one up to 44 tape drives are supported by the 3494 tape library and the accessor can mount/demount up to 300 cartridges per hour.

The AIX operating system sends data to the Magstar or 3490E tape drives through the SCSI interface. System software, like ADSM communicates library commands to the IBM 3494 Library Manager, such as volume/drive mount commands. This communication is done via a LAN or RS-232 attachment.

2.4 Calculating Disk Space

When determining the amount of disk space required, you should have a thorough understanding of the data, its use, and your organizations growth plans. You must ensure that there is enough disk space for the following:

- To store all the data in the database
- To allow for database overhead, indexes, and temporary files (which are necessary for sorting, index creation, and other operations)
- To store log files

Also, when doing your calculation, allow for other factors, such as availability, speed, and the growth rate of the data.

As a general guideline, your disk space should be about three times the size of the raw data. Below is an example of sizing a disk space:

```
Raw data size               = 1.0 GB
Overhead when loaded        = 0.1 GB

Index                       = 0.3 GB
Temporary space             = 1.0 GB
Log space                   = 0.5 GB
Others                      = 0.1 GB
                            --------
Total                       = 3.0 GB
```

Notes:

1. The space required for the index file depends on how the indexes are defined and how many of them are going to created. It is possible for an index file to require more space than the data for the table itself.
2. AIX system files are not included in the preceding calculation.

Consider the following when estimating the size of a database:

2.4.1 System Catalog Tables

When a DB2 Parallel Edition database is initially created, about 1 MB of system tables are created. These system tables will grow as user tables, views, indexes, authorizations, and packages are added to the DB2 Parallel Edition database. If you intend to use a DB2 Parallel Edition configuration with a separate catalog node this is the space needed on the catalog node.

2.4.1.1 User Data Tables

For each user table in the DB2 Parallel Edition database, the space needed is:

```
(average row size + 8) * number of rows * 1.5
```

The average row size is the sum of the average column size, where each column size is described in Table 1.

Table 1. Column Overhead for Data Types

Data Type	Column Byte Count	Comments
SMALLINT	2	
INTEGER	4	
DECIMAL(m,n)	(m+2)/2	
FLOAT	8	
VARCHAR(n)	Current length of data item + 4	To estimate average column size, use the average data size, not the maximum declared size.
CHAR(n)	n	“n” is the length of each data item.
GRAPHIC(n)	n * 2	“n” is the length of each data item and each DBCS character requires two bytes.
LONG VARCHAR LONG VARGRAPHIC	24	The table row contains a descriptor pointing to the LONG VARCHAR or LONG VARGRAPHIC data in the separate .LF file. The descriptor is 24 bytes in length.
DATE	4	
TIME	3	
TIMESTAMP	10	

Note: For every column that allows nulls, add one extra byte for the null indicator. The factor of 1.5 is for overhead such as for page overhead and free space.

2.4.1.2 Long Field Data Files

If a table has LONG VARCHAR or LONG VARGRAPHIC data, in addition to the byte count of 24 for the descriptor (in the table row), the data itself must be stored. Long field data is stored in a separate file as described in Table 1 and *DATABASE 2 AIX/6000 Administration Guide*, SC09-1571-00. To compute the size of this file, compute the actual data length of all the LONG VARCHAR or LONG VARGRAPHIC data and multiply by 1.3 for overhead.

2.4.1.3 Index Space

For each index, the space can be estimated as the following:

```
(average index key size + 8) * number of rows * 2
```

Where the average index key size is the byte count of each column in the index key (using Table 1 on page 17). The factor of 2 is for overhead, such as non-leaf pages and free space.

Temporary space is required when creating the index. The maximum amount of temporary space required during index creation can be estimated as the following:

```
(average index key size + 8) * number of rows * 2
```

Where the factor of 2 is for index overhead as well as space required for the sorting needed to create the index.

2.4.1.4 Log File Space

The amount of disk space in bytes required for log files on each RISC/6000 SP node is minimally:

```
(logprimary * logfilsz * 4096) + 5024
```

where:

- The *logprimary* is the number of primary log files as defined in the database configuration file.
- The *logfilsz* is the number of pages in each logfile as defined in the database configuration file.
- 4096 is the number of bytes in one page.
- 5024 is the size (in bytes) of the log control file.

If the database is configured for circular logging and secondary logs are used during interaction with the database, space must be added for these files during run time:

```
(logsecond * logfilsz * 4096)
```

Where *logsecond* is the number of secondary log files defined in the database configuration file.

Note

For writing log files to disk, you should ensure that the disk can hold at least twice the number of rows that are inserted or deleted by the largest unit of work running on-line. In addition, if you have multiple insert, delete, or update transactions executing at the same time, the minimum space required is approximately twice the sum of the total amount of work being done.

No log space is required if you use the load utility to populate tables; logging does not occur in this case.

2.4.1.5 Work Space - Temp Space

Some SQL statements require temporary tables for processing (such as work files for sorts that cannot be done in memory). These require disk space for storage during the time they are used, and the amount required will be totally dependent on the nature of the queries and therefore cannot be estimated. In most cases using the same size for temp space as the raw data may be sufficient.

When you have the total disk space requirement, divide it by the number of RISC/6000 SP nodes to obtain the amount of disk space for each node.

In some situations it may be advisable to assume about 30 GB of disk space for each of the *database worker* nodes (non catalog and non coordinator nodes), which corresponds to approximately 10 GB of raw data. Depending on your response time requirements, this might be too much if your applications do most of the time full table scans (relational scans) or can be more if the applications do most of the time indexed accesses to the database.

For a fixed database size, the amount of data to be scanned on a single node is inversely proportional to the number of nodes used. For example, a single table scan on a RISC/6000 SP scales as follows:

```
Relational Scan of

1-million-row table of 560-byte row length

4-node        8-node         16-node
-------       ------         -------
115 sec       60 sec         30 sec
```

The above is not a real performance figure. The response time is affected by the number of columns, the width of those columns, and any data conversion that is being done.

2.5 DB2 Parallel Edition Disk Space Management and AIX

DB2 Parallel Edition utilizes the AIX Journaled File System to store its data. When DB2 Parallel Edition creates a database the database manager creates a directory structure within the path specified in the create database command on each node specified in the **db2nodes.cfg** file. This directory structure holds all of the objects associated with the database, which are data files, log files, configuration files and so on. The naming scheme used by the database manager is *database_path/$DB2INSTANCE/NODExxxxx/SQLnnnnn*. For example, if the db2nodes.cfg defines 10 nodes, the following directory tree is created on the nodes one to ten.

```
/db2pe/db2peadm/NODE00001/SQL00001     created on node one
/db2pe/db2peadm/NODE00002/SQL00001     created on node two
        ...........
/db2pe/db2peadm/NODE00010/SQL00001     created on node ten
```

Figure 5 on page 20 illustrates the directories and files generated during the create database process. This directory structure and files are generated on each DB2 Parallel Edition node.

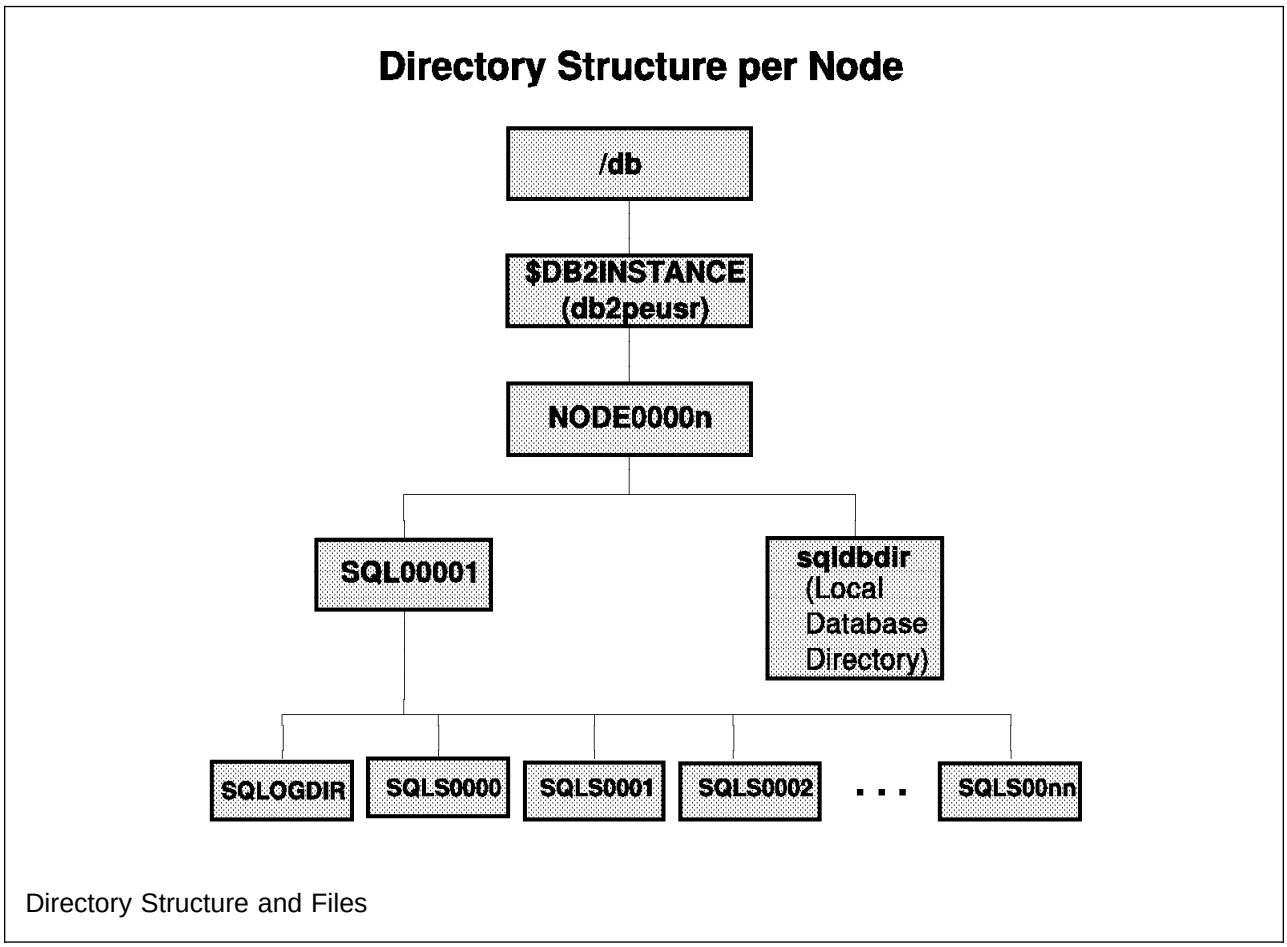


Figure 5. DIRLIST

For further explanations about the directories and files created please refer to *DB2 Parallel Edition for AIX/6000 Concepts and Facilities,* SG24-2514-00. and to *DATABASE 2 AIX/6000 Administration Guide,* SC09-1571-00.

In our previous example, the following command was issued:

```
create database tpcd on /db2pe
```

Where /db2pe is the *database_path*.

It is advisable to create a file system on one of the disks defined for DB2 Parallel Edition usage and mount it on database_path, (in our case /db2pe). This will ensure that the previously described directory structure is created within its own file system and allowing you to do the DB2 Parallel Edition disk space management tasks more easy.

We will discuss the various options you have to store database data and the reasons for doing so. The discussion will concentrate on the subdirectories SQLS0000 to SQLS00nn and SQLOGDIR which will be created during the create database process under the SQL00001 directory. These directories will hold the database data files and the log files.

Data should be distributed across multiple disks. This will enable concurrent access across many disk arms and can improve the overall performance of the system.

2.5.1 AIX File and File System Limitations

The maximum file system size in AIX Version 3 is 2 GB. For AIX Version 4 the maximum size of a file system is 64 GB. Any individual file stored within a file system of AIX has a maximum size of 2 GB. Without spreading the physical data of the database over different file systems, the maximum size of a database partition would be 2 GB or 64 GB per node, depending on the version of the AIX operating system. DB2 Parallel Edition allows you to split the database across many file systems and over many nodes. The maximum size of the database is therefore the file system size multiplied by the number of file systems per node multiplied by the number of RISC/6000 SP nodes.

2.5.2 Mapping DB2 Parallel Edition Objects to AIX Files

Each object, like a table or index, is given a corresponding AIX file name, for example SQL00019.DAT. The file name is created using the capital letters SQL, five digits (the File ID of the corresponding table) of FID and the suffix .DAT for a table or .INX for an index. The FID (File ID) can be found by the following:

```
db2 "select name,creator,fid from sysibm.systables where type = 'T'
```

FID is a unique identifier for a table used by the database manager. When a new table is created, it is allocated the next free FID.

For example:

```
NAME            CREATOR   FID
--------------- --------- ------
SYSTABLES       SYSIBM         2
SYSCOLUMNS      SYSIBM         3
SYSINDEXES      SYSIBM         4
  ......         ....          .
SYSNODEGROUPDEF SYSIBM        18
LINEITEM        DB2PEUSR      19
ORDERS          DB2PEUSR      20
```

Files, like SQL00019.DAT or SQL0019.INX, are stored in the segment directories SQLS0000 to SQLnnnn on each DB2 Parallel Edition node. It is important to make the distinction here between the following:

- A segment directory, which holds segments (like SQL00019.DAT) of a table
- A segment, which is part of a table (the SQL00019.DAT file itself)

2.5.3 DB Manager, Segment Manager Tool, and AIX Logical Volume Manager

Disk space management for DB2 Parallel Edition data objects involves three different levels of software.

- The database manager is responsible for data in tables, indices and databases.
- The segment manager is a tool that helps you create and manage file systems for database segments and segment directories.
- The AIX logical volume manager and file system is responsible for storing the files on the physical disk.

The database manager will only spread the database across multiple segment directories. If nothing else is done, these directories will be within one file system and all the files stored in the directory structure will be limited by the AIX file size

limit of 2 GB as well as by the AIX file system limit. This imposes a certain limit on the maximum size of a table and the whole database.

There is a process to mount these segment directories onto their own file systems and to move the contents of these directories, the segmented table and index files to these file systems. A tool, called the DB2 segment manager tool, is provided to do this. You can use the tool to do the following:

- Query the mount status and file system size of segment file systems
- Create new segment file systems in which to store segmented table files
- Extend the size of existing segment file systems
- Clean up segment file systems after dropping a database

The DB2 segment manager tool is called via the db2sgmgr command and is described in detail in Appendix A, “How to Manage the DB2 Parallel Edition Database Segment Directory File Systems” on page 241. See also the *DATABASE 2 AIX/6000 Command Reference,* SC09-1575-00.

2.5.4 Working with Multiple Segments

To work with segments, you must specify values for the database configuration parameters segpages and numsegs during database creation. The segpages parameter balances space consumption across the segment subdirectories. Its associated configuration parameter, numsegs, specifies the number of segment subdirectories across which consumption is balanced.

2.5.4.1 Segmented Tables and Segment Subdirectories

By controlling the amount of data written to a given table segment, the database manager evenly spreads the data written to database files over the segment subdirectories. When the number of 4 KB pages specified by the segpages parameter is written to the segmented table file in a segment subdirectory, the database manager writes segpages to the next one, and will continue to cycle through all of the segments. After writing the specified number of pages to all the segment subdirectories, it returns to the first segment subdirectory. This process is called striping. The cycle continues until one of the segmented table files reaches the 2 GB file size limit or the segment directory becomes full.

The number of segment subdirectories is defined by the numsegs parameter. The value for this parameter as well as the value for the segpages parameter is specified when the database is created.

Attention

Select an appropriate value for these parameters, because their value is not easily changed. If you discover that the parameters must be changed, you have to do the following:

1. Offload the data using BACKUP or EXPORT.
2. Drop the database.
3. Create a new database with appropriate values for numsegs and segpages.
4. Load the data using RESTORE or IMPORT into this new database.

For large databases in the gigabytes or even terabytes, this is a time-consuming process. Care should be taken when estimating the values for segpages and numsegs.

2.5.4.2 Estimating the Segpages and Numsegs Parameters

To determine useful values for the segpages and numsegs parameters, you must understand the files and directories representing the DB2 Parallel Edition database. Further, you must know the limitations that AIX imposes on the size of a file system and a single file, and how the segmented tables feature is designed to circumvent these limitations.

To choose appropriate values for segpages and numsegs, begin by estimating the size of your database and the largest table. For more information about this task, see 2.4, “Calculating Disk Space” on page 16 and 2.4.1.1, “User Data Tables” on page 17.

Maximum Size for Tables and Databases

The maximum size of a database is dependent on a number of factors, including numsegs, the AIX file system size, and the number of nodes in your configuration. Assuming each segment is mounted over a different file system and segpages has been chosen small enough to ensure a balanced distribution of table and index data through striping across segments, then the maximum database size on one node could be computed as follows:

```
Max database size on one node = ( NUMSEGS * min(file system size))
```

If we further assume that each node holds the same amount of data as a result of even data distribution, then the maximum database size would be:

```
Max database size = (NUMSEGS * min(file system size)) * number of nodes
```

The maximum size of a table depends on the value of numsegs, the AIX file size, and the number of nodes in your configuration.

```
Max table size = ( max(NUMSEGS) * max(file size) * number of nodes )
```

Determine the Minimum Number of Segment Directories

The numsegs may take the following values:

Minimum	1
Default	16
Maximum	256

The actual value for numsegs is calculated in the following manner:

$$\text{numsegs} >= \frac{\text{database size}}{\text{file system size} * \text{number of nodes}}$$

or

$$numsegs >= \frac{\text{largest table size}}{\text{file size} * \text{number of nodes}}$$

The larger value of both equations is the minimum value for numsegs.

Another consideration for the numsegs value is the number of disks per RISC/6000 SP node dedicated for DB2 Parallel Edition usage. This would allow to put each segment directory on a separate disk and have distributed (striped) the table files over multiple disks. However, if the value calculated by the equation above is larger, the larger value has to be taken as the minimum value for numsegs.

This value determines how evenly balanced the space consumption is across segment subdirectories. For example, if you have a very large setting, say, 100000, one segmented table file may consume 100000 pages, while the other segmented table files are empty.

Smaller values for segpages distribute the data more evenly across segment subdirectories. The default of 32 is probably sufficient in most situations.

The segpages may take the following values:

Minimum	4 4K pages
Default	32 4K pages
Maximum	524288 4K pages

2.5.4.3 Segment Considerations

When you choose a value for segpages keep the following in mind:

- Try to leave equal amounts of free space in each segment subdirectory. A smaller segpages value can help distribute the table more evenly over multiple segments.
- The database manager does not always start a segmented table file in the first segment subdirectory. Rather, it attempts to balance the load of the table files across the segments. The starting segment subdirectory for a table is rotated such that *table2* starts at segment subdirectory two. The starting segment subdirectory for tables and indexes is derived the following way:

  ```
  Segment Subdirectory Number = MODULO(FID/numsegs)
  ```

  So, for a table whose FID = 19 and, using the default of numsegs = 16, the starting segment subdirectory would be the following:

  ```
  Segment Subdirectory Number = MODULO(19/16)
  Segment Subdirectory Number = 3 which is SQLS0003
  ```

  For the value of FID, see 2.5.2, “Mapping DB2 Parallel Edition Objects to AIX Files” on page 21.
- Tables include table files (SQLnnnnn.DAT), index files (SQLnnnnn.INX), and long-field files (SQLnnnnn.LF). A large table with indexes will also likely have

large indexes. Keep this in mind when determining your values for numsegs and numsegs.

- If you want a large amount of data per node, you should set numsegs to a higher value, though you will have to manage a greater number of segment subdirectories.
- If you expect that the database will contain, for the most part, large tables, you should set segpages to a smaller (maybe default) value.

2.5.4.4 Segmented Tables and Segmented Table Files

Segment subdirectories are created in the database directory path. The number of subdirectories created will be equal to the value of the numsegs parameter. They are named sequentially, beginning with *SQLS0000*. Figure 6 on page 26 shows an illustration of the directory structure and the placement of segmented table files within these directories.

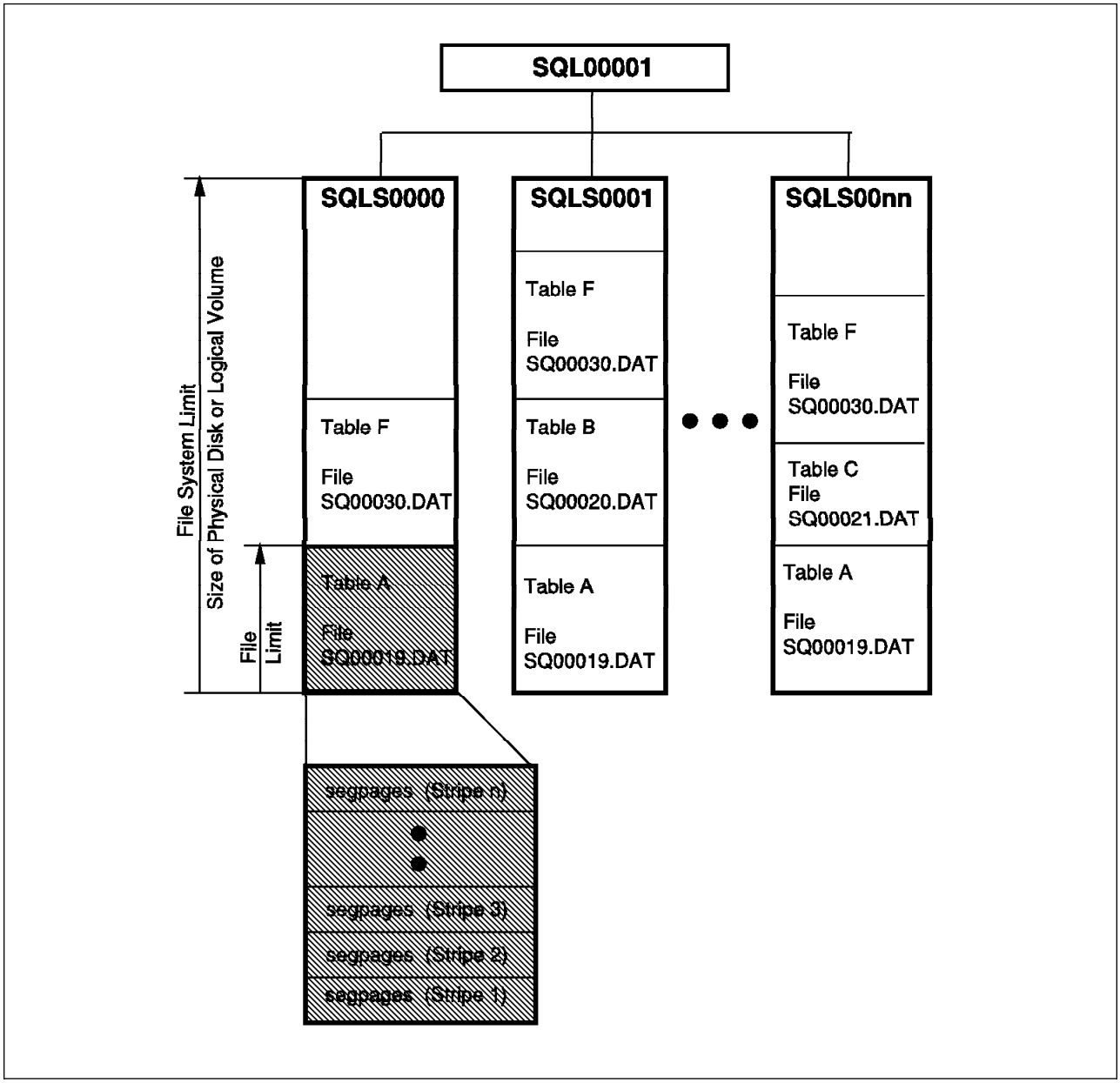


Figure 6. Segmented Tables Files and Directory Layout

Figure 6 indicates how the directory structure and files are generated on each DB2 Parallel Edition node. To allow a DB2 Parallel Edition database and individual tables to exceed the operating system file and file system limit, the DB2 Parallel Edition database manager stores table data in multiple files across segment directories. For example, the first table data file (SQL00019.DAT) is created in the first segment directory. The database manager will allow this *.DAT file to grow to the size defined by the segpages parameter. After the table file reaches this size, the database manager will write the next segpages pages of data to SQL00019.DAT in the second segment subdirectory. This process continues until all of the segment subdirectories contain SQL00019.DAT files that are segpages in size. Then the database manager will write another segpages pages of data to the SQL00019.DAT file in the first segment subdirectory. The database manager stripes through the segment subdirectories until a segmented table file reaches its

limit of 2 GB. The same mechanism is used for index files (SQLnnnnn.INX) and long-field files (SQLnnnnn.LF).

Figure 7 shows a potential problem when underestimating the value of numsegs and selecting a too large value for segpages.

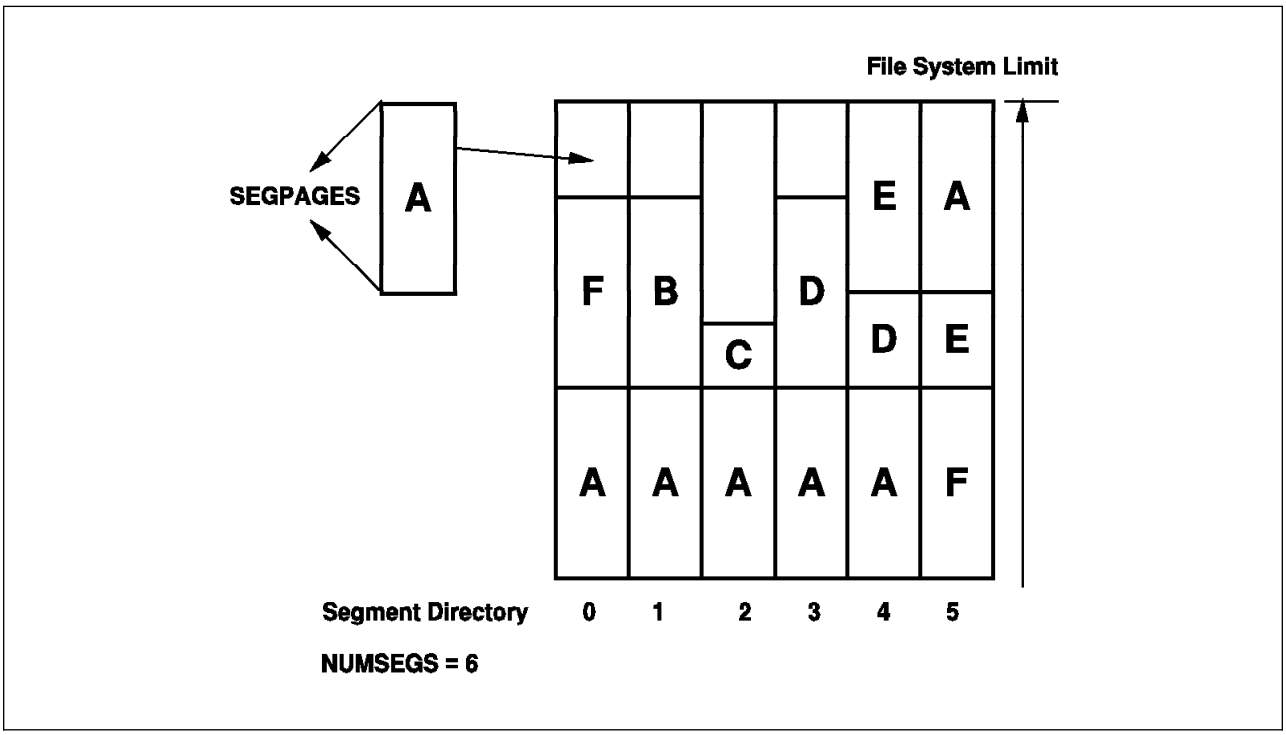


Figure 7. Placement of Table Segments within the Segment Subdirectories

Figure 7 shows one node of a DB2 Parallel Edition database that was created with numsegs = 6. This is a logical view. The Table Segments are actually allocated within the SQLnnnnn.DAT files. Table segments are indicated by the letters A to F. These segments are stored in the SQLnnnnn.DAT files in the different segment subdirectories. So, the first segpages pages of A would go to subdirectory SQLS0000, the next to subdirectory SQLS0001 and so on. The sequence of events in Figure 7 was the following:

1. Database manager places segpages pages in segment subdirectories 0 to 4 for table A.
2. Database manager places segpages pages in segment subdirectory 1 for table B.
3. Database manager places a part of segpages pages in segment subdirectory 2 for table C (might be a small table only).
4. Database manager places segpages pages in segment subdirectory 3 and a part of segpages pages in segment subdirectory 4 for table D.
5. Database manager places segpages pages in segment subdirectory 4 for table E.
6. Database manager places segpages pages in segment subdirectories 5 and 0 for table F.
7. Database manager extends table E by a part of segpages pages into segment subdirectory 5.

8. Database manager extends table A by segpages pages into segment subdirectory 5.
9. Database manager tries to extend table A by segpages pages into segment subdirectory 0. However segment subdirectory 0 has no space left to store this segment, even if space is left in other segment subdirectories. You will receive an SQL error code indicating that the file system is full.

2.5.4.5 Mapping Segment Directories to File Systems

The rationale behind mapping segment directories to file systems on separate disks is to overcome the file system limitations of AIX and perhaps performance improvements by striping table data files over multiple physical disks. For each table, segpages pages of the table data files (.DAT, .INX, .LF) are written to a different disk. This allows to take advantage of some parallel I/O effects. The following are the steps to be done to move segment directories to separate file systems:

1. The database must be stopped to avoid an inconsistent database while moving database data to another file system.
2. Create the new file system (or systems) on the additional disk (disks).
3. Make the new directories for the segments on the new disks.
4. Copy the data files to the new file system on the additional disk (or disks).
5. Delete the data on the old file system for these segments.
6. Mount the new file system (or file systems) on the old segment directories.

A detailed description of these steps can be found in Appendix A, “How to Manage the DB2 Parallel Edition Database Segment Directory File Systems” on page 241. Since this procedure is error prone you should use it very careful to avoid loss of your database data. The preferred method to accomplish this tasks should be by using the DB2 Parallel Edition segment manager tool. The DB2 Parallel Edition segment manager tool provides a command line interface as well as an AIX SMIT interface.

Figure 8 on page 29 shows an example of mapping ten segment subdirectories contained in one file system on one disk to ten file systems on ten different disks.

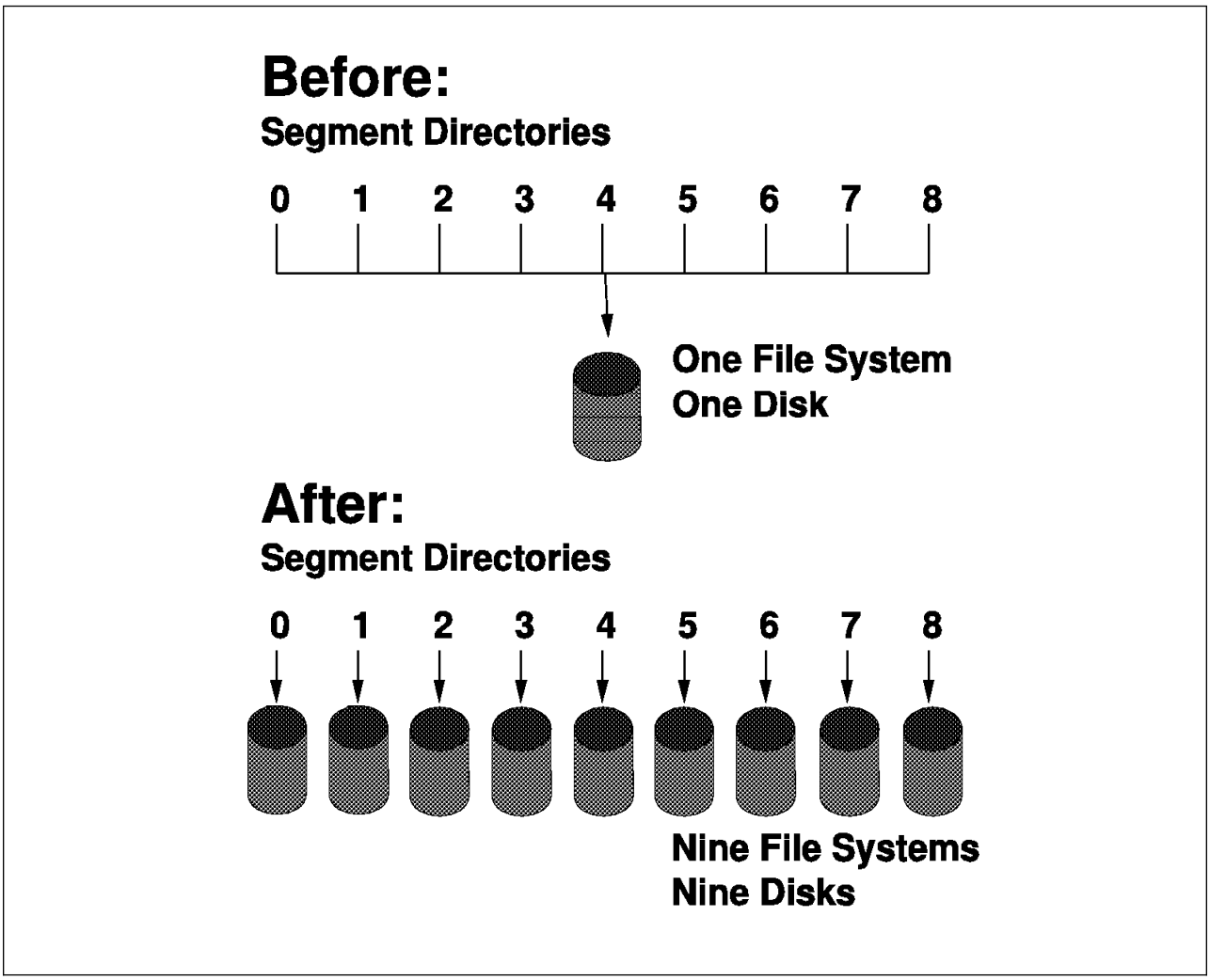


Figure 8. Mapping Segment Directories to Separate File Systems

2.6 Separating Log Files and Database Files

If you expect large update or insert workloads for your database it might be helpful to move the log files to separate disk. If your disk subsystem offers different RAID modes, you can define the disks for the database files in RAID-5 mode (to have data protection) and the disks for the log files in RAID-1 mode (to have data protection and better performance).

To separate the logfiles from the database files do the following:

- Use the AIX SMIT interface to define a new volume group with additional disks (if necessary), and create a file system in that volume group.
- Use the *update database configuration* command to change the *newlogpath* configuration parameter to the new directory path for the logs.

For details about newlogpath and the update database configuration command, see the *DB2 Parallel Edition for AIX, Administration Guide and Reference,* SC09-1982.

2.7 Coordinator Nodes

User interaction with DB2 Parallel Edition is through one node. This node is known as the *coordinator node* for that user. It might be the same node that the application is running on, or in the case of a remote application, the node to which the application is connected. All DB2 Parallel Edition nodes can be used as the coordinator node. You may consider spreading out users across nodes to distribute the coordinator function.

It is possible to use a dedicated coordinator node. This is a node that does not contain user data in the database partitions. You may want to use a dedicated coordinator node to offload the coordinator task from the database worker nodes. That is, for example, if you have a large database system and run queries in which work is done at the end by the coordinating agent (for example, a sort or merge).

Since a dedicated coordinator node holds no user data, you lose the capacity of that node for parallel query processing. In a small system the impact of using a dedicated coordinator node is higher than in a large system. For example, in an eight node system, one dedicated coordinator node represents 12.5% of the processing resources and in a 30 node system, a dedicated coordinator node represents 3% of the processing resources.

A dedicated coordinator node might be useful if a lot of users connect to the database and issuing relatively short running but many queries.

If you plan to have a dedicated catalog node, you can use it as a coordinator node.

2.8 Choosing a Catalog Node

When you create a database, the node on which the create database command is issued becomes the catalog node for that particular database, and all system catalog tables for that database are stored there. Because all access to system tables must go through this node, this is a critical resource. You should regularly back up the catalog node, and avoid putting user data on it, as this data increases the time required for backup (or restore) operations.

A dedicated catalog node may gain some performance improvements if your applications are using short dynamic SQL queries most of the time.

2.8.1 Using a Dedicated Catalog Node

If you want to separate the database catalog from database data you can do it in two ways:

- Dedicate one node to handle all the database catalog function. Ensure that no user tables are created on this user node, and do not include this node in any nodegroups that will contain tables. You can use a dedicated catalog node together with a coordinator node.
- If you do not want to dedicate a RISC/6000 SP processor node for catalog functions, you can run two DB2 Parallel Edition logical nodes on one processor: one logical node dedicated to the database catalog function (as above) and the other logical node as part of a nodegroup that contains data.

2.9 Nodegroups

A nodegroup is a named subset of nodes defined in a database. Nodegroups can contain from one node to the entire number of nodes defined for the DB2 Parallel Edition database system in the *db2nodes.cfg* file. Each subset that contains more than one node is known as a *multinode nodegroup*. Figure 9 shows the relation of tables to nodegroups to database and instance.

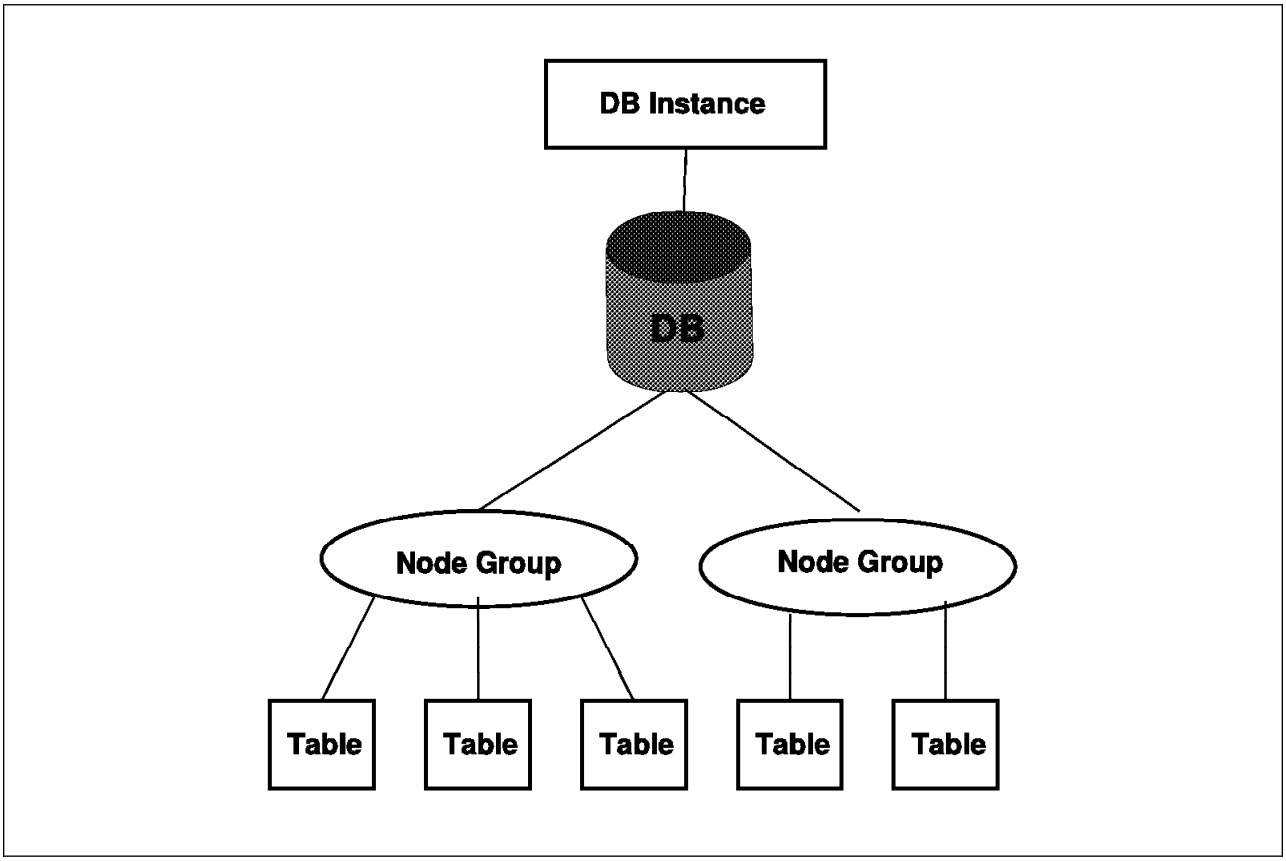


Figure 9. The Table to Nodegroups to Database and Instance Relation

When you create tables for the DB2 Parallel Edition database, you first create the nodegroups where the tables will be stored. Then you create the tables and the data to be stored in the nodegroup is partitioned across the nodes of the nodegroup. Two nodegroups will already exist after the creation of each database: the default nodegroup (IBMDEFAULTGROUP) and a single node nodegroup (IBMCATGROUP) on the catalog node. The IBMDEFAULTGROUP contains all the nodes defined in the db2nodes.cfg file.

You can create a nodegroup with the create nodegroup statement. This statement specifies the set of nodes on which the tables are to reside. Figure 10 on page 32 shows the default nodegroup, the catalog nodegroup and two user defined nodegroups.

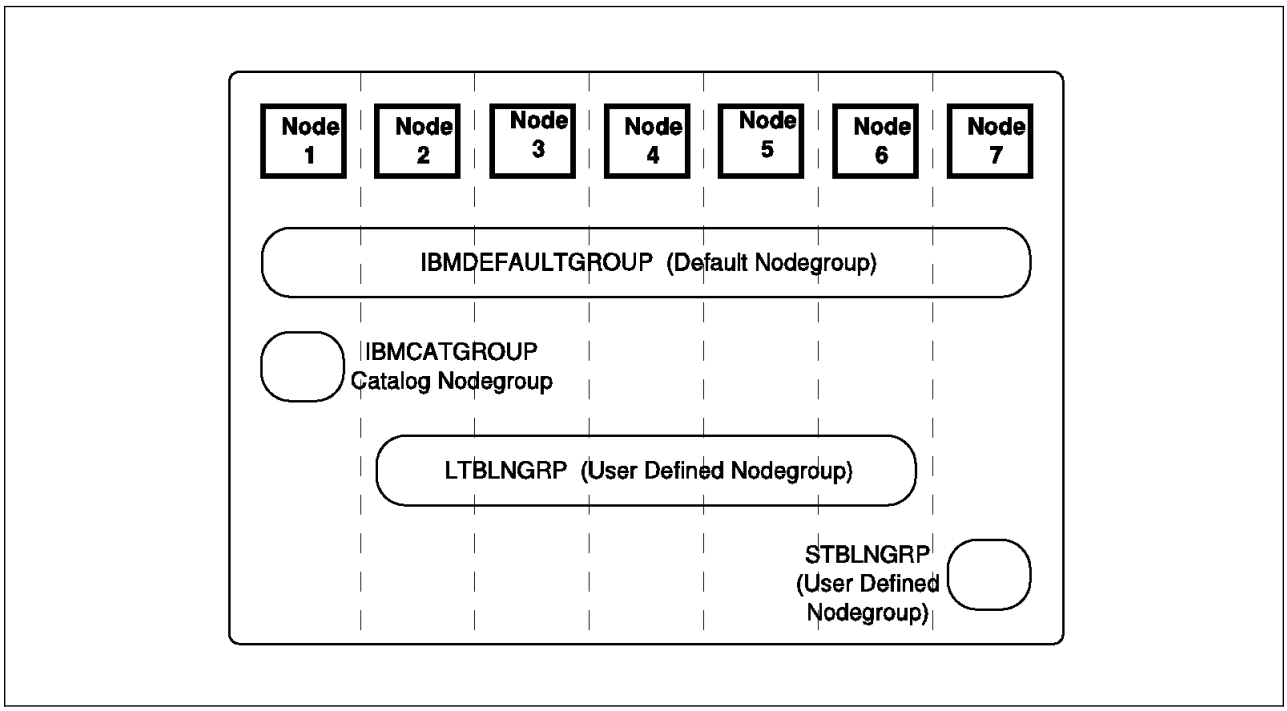


Figure 10. Nodegroups in a DB2 Parallel Edition Database.

You can drop nodes from a nodegroup, or if new nodes have been defined in db2nodes.cfg, you can add them to an existing nodegroup. As your database increases in size, adding nodes to an existing nodegroup gives you the possibility to scale your database according to your needs. The additional nodes will be utilized by redistributing the data in that nodegroup.

2.9.1 Nodegroup Considerations

If you require fast recoverability, avoid placing user data on the catalog node. To do this, place user tables in nodegroups that exclude the catalog node. This helps to ensure fast recovery of the catalog node.

Place small tables in single-node nodegroups, except if you want collocation with a larger table. This might not be always compatible with performance considerations.

Since there is a performance trade-off between distributing table data across all nodes and keeping table size on each node large enough to avoid suboptimal disk usage, it might be beneficial to keep medium-sized tables on a subset of the available nodes. For example, a 100 MB table may perform better on a 16-node nodegroup than on a 32-node nodegroup.

You can use nodegroups to separate on-line transaction processing tables from decision support tables to ensure that the performance of OLTP transactions is not impacted by decision support applications.

Chapter 3. Installation of Hardware and Software for MVS Connectivity

When you install DB2 Parallel Edition you may want to connect your IBM S/390 mainframe to your RISC/6000 SP so you can transfer data between them. The software that you may wish to install includes the following:

- TCP/IP
- CLIO/S
- SNA
- DDCS
- DataPropagator
- BatchPipes/MVS

Chapter 6, “Data Migration and Loading” on page 101 describes how you may use these products.

This chapter describes the installation and customization of the following:

- Hardware
- Software

Useful related reading includes:

- *RISC/6000 370 Channel Support ESCON and Block Multiplexer,* SG24-4589-00.
- *IBM CLIO/S User's Guide and Reference CLIO-user-2, that can be retrieved from /pub/pps/clios/clio_user_ref.ps.z on the anonymous FTP server on lscftp.kgn.ibm.com*.
- *AIX Version 4.1 Enterprise System Connection Adapter: User's Guide and Service Information,* SC31-8197-00.
- *AIX Version 4.1 Block Multiplexor Channel Adapter: User's Guide and Service Information,* SC31-8196-00.

3.1 Hardware

This section includes:

- An overview of S/390 Channel Architecture, to aid the communication between RISC/6000 SP planners and the MVS planners.
- Hardware planning
- SMITTY definitions of the hardware

3.1.1 Overview of S/390 Channel Architecture

The implementation of I/O hardware on the mainframe is different than the implementation on the RISC/6000 SP. As you are going to have to talk with the mainframe planners and system programmers, it is important that you understand the structure and terminology they use and vice versa. This section is an overview of the mainframe implementation, relating it to an RISC/6000 SP configuration. It is not intended to be precise, nor to describe all the power and complexity of either

system; it is designed to enable meaningful communication between the two parties.

Figure 11 on page 35 shows a simple mainframe configuration. The processor is connected to the I/O via channels. A channel is analogous to the microchannel on the RISC/6000 SP. However, a typical mainframe will have many channels. There are several different channel types. ESCON is the most modern, and the one recommended for attaching to an RISC/6000 SP. The other type is a block-multiplexer but this is older technology and much slower and is not discussed further in this chapter. Each channel has an identifier knows as a Channel Path Identifier (CHPID) that is two hexadecimal digits, for example, 3C.

Connected to a channel you can have one physical control unit, for example an adapter on an RISC/6000 SP node. Defined within the physical control unit there can be one or more logical control units. The logical control units can be of different types, such as CLAW or SNA. A logical control unit provides a similar function to a device adapter but is part of a physically separate box outside the computer. The logical control unit is identified in the following two ways:

1. By a control unit address (CUADD). This is one hex digit, for example 1, that distinguishes the different logical control units in the physical control unit.
2. By a control unit number (CUNUMBER). This is four hex digits and is used in the hardware definition of the mainframe.

The CUADD is a part of the RISC/6000 SP definition, whereas the CUNUMBER is invisible to the RISC/6000 SP.

Attached to a control unit are one or more devices, connected in a similar way as several devices are attached to one RISC/6000 SP adapter. The individual devices are identified in the following two ways:

1. The unit address is two hex digits, and it identifies the unit on the control unit, for example 02.
2. The device number is four hex digits and uniquely identifies the device. It is this number that is used by an MVS application when it wishes to communicate with the device.

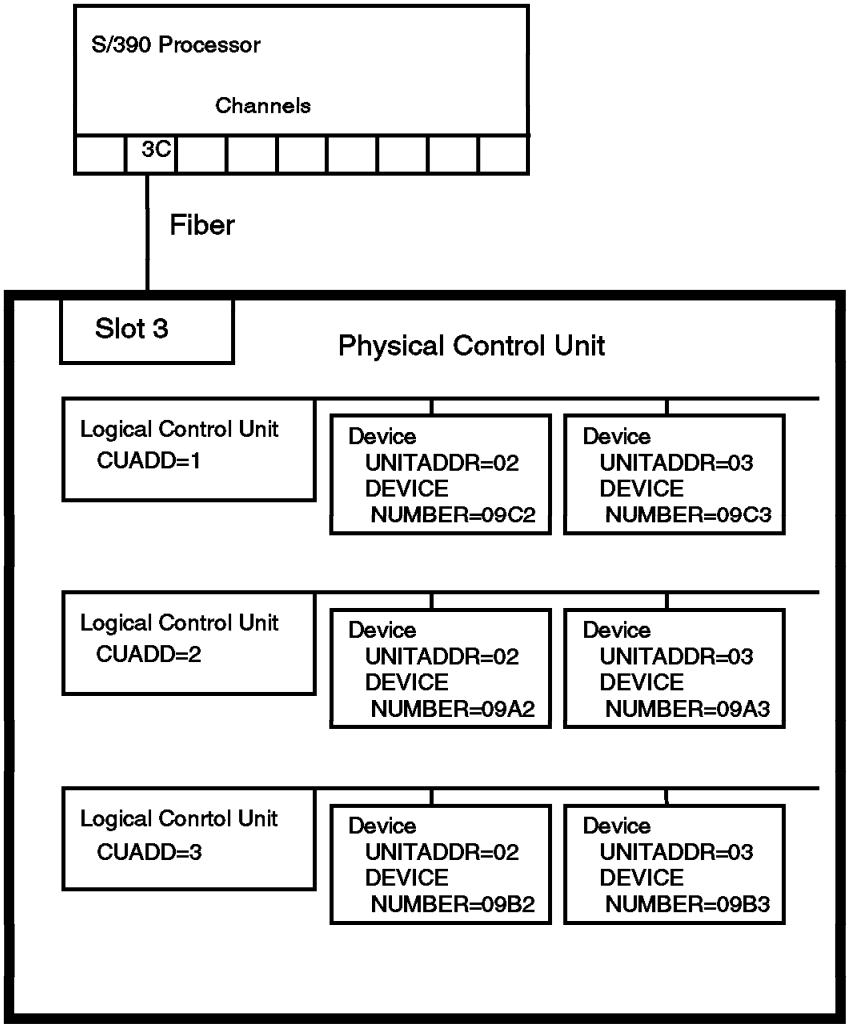


Figure 11. Simple Mainframe Configuration

This simple configuration can be made more flexible, and complex, by connecting physically control units via a switch, known as an ESCON director, to multiple channels on the mainframe (see Figure 12 on page 37).

The director has a number of ports. The ports connected to the channels have a Source Link Address (SLA) and those attached to the physical control units have a Destination Link Address (DLA). You will need to know the DLA and SLA to complete the installation of the connection.

One SLA can be connected to multiple DLAs enabling one channel to be connected to multiple physical control units. For example in the figure DB is connected to AC and AB.

This enables multiple RISC/6000 SP nodes to connect to one channel. This may be an economic solution if you need to connect many nodes directly to the mainframe and you do not use many of them concurrently. This would have to be compared to connecting nodes via the High Performance Switch (HPS) to a node that is attached to the mainframe. This configuration could also be used to connect one channel to multiple RISC/6000 SPs or to share the channel with other non-RISC/6000 SP devices. You should discuss the usage patterns of the connection and the performance requirement with your hardware planner to see if sharing a channel is sensible.

One DLA can be connected to multiple SLAs enabling one physical control unit to be attached to multiple channels. For example AB is connected to DB and DA.

This will allow one RISC/6000 SP node to attach to more than one channel (the channels could be on different MVS images). This would enable data to be retrieved by one node from two different operational systems. To do this you will need to define different logical control units to attach to the different channels because RISC/6000 SP logical control units can only be defined with one path.

A S/390 processor complex can be divided into multiple logical units, known as LPARs. The relationship between channels and LPARs can be one of the following:

- Dedicated (DED) means that a channel is always connected to one LPAR.
- Reconfigurable (REC) means that the channel can only be attached to one LPAR at a time, but can be connected to another by the operator reconfiguring it.
- Shared (SHR) means that a channel is connected to more than one LPAR at the same time.

The RISC/6000 SP ESCON adapter cannot be connected to a shared (SHR) channel.

An esoteric name may be given to a group of devices of the same type. This allows an application to ask for a device by type rather than specifying a specific device. CLIO/S uses this facility.

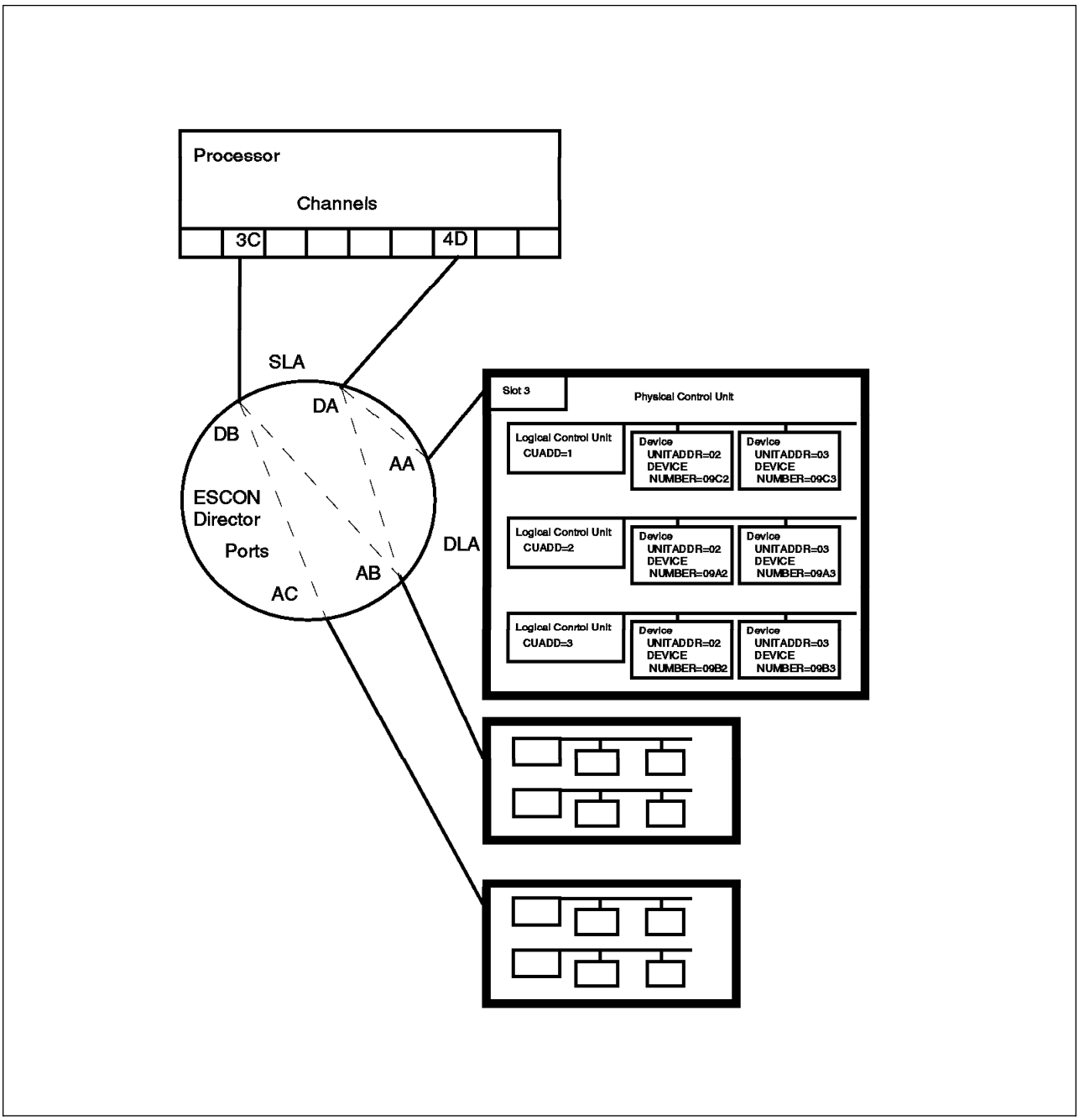


Figure 12. ESCON Directors

3.1.2 Hardware Planning

Depending on what software you are installing you will need to define different types of logical control units. These include the following:

- TCP/IP CLAW requires an RS6K
- CLIO/S requires RS6K
- SNA requires 3174

These must be defined to the mainframe before the RISC/6000 SP is connected. These definitions and the physical configuration must be defined by the people responsible for mainframe hardware planning and system programming.

The table below shows all the information required to define the link. For each piece of information there is a short explanation and the responsibility of the S/390 and SP hardware planners and system programmers.

D Responsible for choice and definition.

K Needs to know this piece of information but needs to be given it by someone else.

I This information is useful, but not essential.

Number Points at a note below.

Table 2 (Page 1 of 2). Parameters for Defining an ESCON-RISC/6000 SP Connection

Name	Explanation	SP	S/390
Channel Information			
LPAR Name	The name of the LPAR that CU is to be attached to.	D	K
CHPID	ID of the channel the CU is to be connected to either directly or through a director.		D
Type	Channel type. For an ESCON attached SP this must be CNC.	D	K
Mode	Is the CHPID to be dedicated (DED) or reconfigurable (REC).	D	K
Logical Control Unit Information			
Control Unit Number (CUNUMBER)	An internal S/390 identifier.		D
Control Unit Type (CUTYPE)	The type of control unit, for example RS6K or 3174.	DK	K
Path (CHPID)	Same as CHPID repeated here for simplicity of MVS definition.	I	D1
Switch Number	Identifier of the ESCON Director that connects the SP to the S/390. If direct connect say DIRECT.	I	D1
DLA	Destination Link Address	K	D1
SLA	Source Link Address	K1	D
Control Unit Address (CUADD)	This uniquely identifies this CU within the SP. Start at 1 and number upwards	DK	K
Lowest Unit Address (UNITADD)	The lowest unit address for this CU, normally start at 00.	DK	K
Range	The number of units on this control unit.	DK	K
Node Identifier	The name you use to identify the physical node, so that the cablers can attach the fiber to the correct node.	D	I
Slot Number	The number of the slot,within the node, that the fiber is attached to.	DK	I
Device Information			

Table 2 (Page 2 of 2). Parameters for Defining an ESCON-RISC/6000 SP Connection			
Name	**Explanation**	**SP**	**S/390**
Device Number	The lowest device number of the range.	K	D
Number of Devices	Same as range, repeated here to simplify S/390 definition.	D	K
Device Type	Type of device, for example RS6K or 3179L, assume all the devices are the same for the CU.	D	K
Control Unit Number (CUNUMBER)	Same as CUNUMBER above, repeated here to simplify S/390 definition.		D
Unit Address (UNITADD)	Same as UNITADD above, repeated here to simplify S/390 definition.	D	K
MVS Information			
MVS Configuration ID	The ID of the specific MVS the connection is going to be defined to. SP has to decide which MVS and get the name it is known by from S/390.	D	K
EDT #	Needed by MVS.		D
Esoteric Name	The esoteric name to be given to the group of devices (for example, RS6K2).	DK	K
Device Number to Esoteric Name	Which devices have this name, normally all of the devices on the CU.	DK	K
OFFLINE	Use default unless specified below.	D	K
DYNAMIC	Use default unless specified below.	D	K
LOCANY	Use default unless specified below.	D	K
Missing Interrupt Handler	Value for missing interrupt handling.	D	K
SMITTY Information			
Subchannel Name	Needed for SMITTY.	DK	
Fiber Name	Needed for SMITTY.	DK	

Note:

1. The values for CHPID, SLA and DLA will depend on whether the RISC/6000 SP is directly connected to the channel or connected via a director.
   - If the connection is via a director then:
     - The CHPID is for information only for the SP.
     - The DLA and SLA are the relevant ports of the director.
   - If the connection is direct to the channel then:
     - The DLA is the same as the CHPID.
     - The SLA is defined as 01.

Planning and communication can be done using the form below. The process should be started by the SP system programmer who should fill in all the items with a D in the SP column. It should then be given to the S/390 hardware planner and system programmer who will fill in the remaining items.

When the form is complete the SP system programmer should have everything required for SMITTY.

Table 3. Work Sheets for Defining an ESCON-RISC/6000 SP Connection			
Name	**SP**	**S/390**	**Value**
Channel Information			
LPAR Name	D	K	
CHPID		D	
Channel Type	D	K	
Mode	D	K	
Control Unit Information			
Control Unit Number (CUNUMBER)		D	
Control Unit Type (CUTYPE)	DK	K	
Path (CHPID)	I	D1	
Switch Number	I	D1	
DLA	K	D1	
SLA	K	D1	
Control Unit Address (CUADD)	DK	K	
Lowest Unit Address (UNITADD)	DK	K	
Range	DK	K	
Node Identifier	D	I	
Slot Number	DK	I	
Device Information			
Device Number	K	D	
Number of Devices	D	K	
Device Type	D	K	
Control Unit Number		D	
Unit Address (UNITADD)	D	K	
MVS Information			
MVS Configuration ID	D	K	
EDT #		D	
Esoteric Name	DK	K	
Which Devices Have This Name	DK	K	
OFFLINE	D	K	
DYNAMIC	D	K	
LOCANY	D	K	
Missing Interrupt Handler	D	K	
SMITTY Information			
Subchannel Name	DK		
Fiber Name	DK		

3.1.3 SMITTY Definition of Hardware

Having received the answers from the hardware planners you have some of the information you need for SMITTY. The rest you will get when planning a particular software environment. Table 4 shows the relationship between the MVS values and the SMITTY parameters.

Table 4. Relation between Work Sheets and SMITTY Hardware Definition

Name	**Use in SMITTY Hardware Definition**
Channel Information	
Control Unit Type (CUTYPE)	Dependent on software.
DLA	Subchannel local address
SLA	Subchannel path (first two characters).
Control Unit Address (CUADD)	Subchannel path (last character).
Lowest Unit Address (UNITADD)	Remote address for first unit in range (add one for each subsequent unit).
Range	Number of units to include in range.
Node Identifier	This is the node you are running SMITTY on/for.
Slot Number	Fiber slot number.
Device Information	
Device Number	Sub-channel local address, are the bottom two characters of the lowest device number.
MVS Information	
Esoteric Name	Used in CLIO/S definition.
SMITTY Information	
Subchannel Name	Subchannel name and in fiber subchannel list.
Fiber Name	Fiber name.

Note:

1. You can define any name for the subchannel; however some subchannel types expect a particular name (for example CLAW expect es0-es8) (es0).
2. The subchannel path is three hex digits:
   - The first two hex digits are the ESCON path.
   - The last hex digit is the control unit address.

   The ESCON path is:
   - 01 if the channel is directly attached to the ESCON.
   - The ID of the port on the ESCON director attached to the MVS channel if the channel is attached via a director (011).
3. The local address is the bottom two hex digits of the device number (A0).
4. The remote address is the unit address (00).
5. The fiber name is an arbitrary name that distinguishes multiple channel adapters on the same node (use fiber1 if you only have one).

6. The fiber slot number is the number of the channel adapter on the RISC/6000 SP node (3).
7. The fiber subchannel name list must contain an entry for the subchannel you are defining (es0,...).

3.2 Software

All the software products are installed using the standard IBM tools, SMP/E on MVS and the install function of SMITTY on AIX. Therefore the basic installation of the products is straightforward. However the environment, with two different computer systems coming together, is complex, especially if you are installing multiple new products at the same time. Therefore the need for good preparation cannot be overstated, please:

- Check all prerequisite and corequisite products.
- Read all relevant installation documentation.
- Check with IBM for any defect information or installation tips.
- Stop if any step of the installation fails and resolve the problem before carrying on.
- Discuss and agree on any activities that are to be carried out by colleagues in other areas.

The following sections describe the configuration and customization of each of the software environments.

3.2.1 TCP/IP Customization

The customization of TCP/IP is concerned with correctly relating IP identifiers. Each end of the link between the RISC/6000 SP and MVS must have an IP address. These will be allocated by a coordinator within your organization:

- The SP node will already have been allocated at least two other IP addresses (one for the Ethernet and one for the high-speed switch). An extra one is required for the channel adapter.
- The MVS node can share a single IP address for all the channels attached.

Each of these IP addresses should have a name associated with it. It is simpler if the name has a structure such as SP2SW01 for the high-speed switch on node 1 and SP2ES01 for the ESCON adapter on node 1. This is further reflected in the IP address, for example 9.12.21.1.

You must specify to your hardware planner that you need the following:

- A control unit of type RS6K, with two units on it.
- Non-dyanmic UCB (for Version 3 and below).
- Missing interrupt handling set to MIH=00.

The relationship between these parameters and SMITTY and MVS is summarized in Figure 13 on page 44

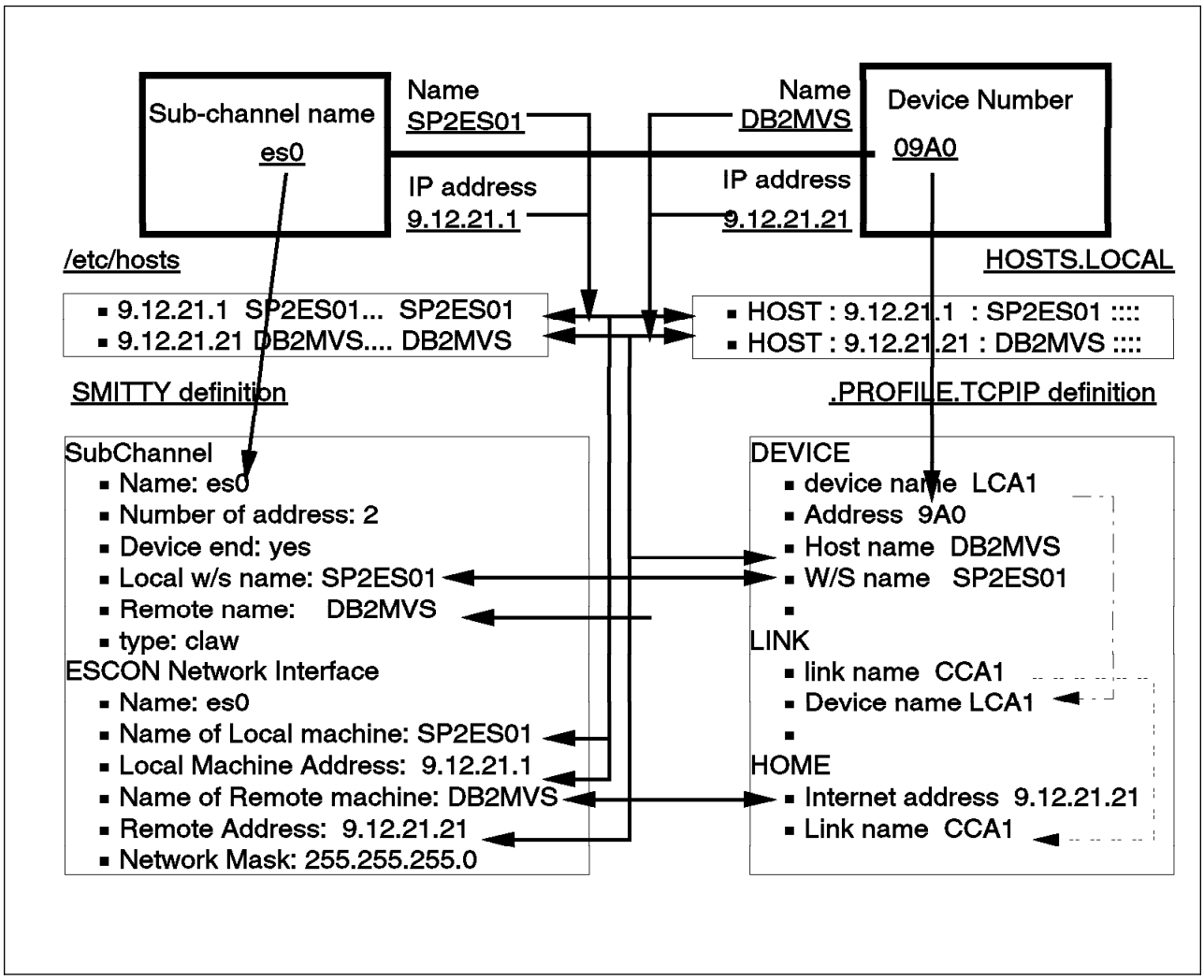


Figure 13. TCP/IP Customization

Note:

1. The IP addresses and names should be added to the tcpip.HOSTS.LOCAL file on MVS, and the /etc/hosts file on the node.
2. In SMITTY esca, fill in the following:
   a. Subchannel:
      - Name is es0 unless you have multiple adapters on the same node.
      - Number of addresses in group is 2 for TCP/IP.
      - Device End is yes.
      - Local W/S name is the IP name of the SP channel adapter.
      - Remote name is the name of MVS.
      - Type is CLAW for TCP/IP.
      - The other parameters (path, local address and remote address) are defined by the hardware, see Table 4 on page 42.

   b. ESCON network interface:
      - Name relates to the subchannel defined above.

- SMITTY will automatically copy the local and remote machine names.
- The local and remote addresses are the related IP addresses.
- The network mask will be given to you by the TP coordinator.

3. In the .PROFILE.TCPIP file you need to specify:
   a. DEVICE
      - Device name is an arbitrary name that is unique within the profile.
      - Address is the device number specified in the hardware.
      - Host name is the IP name of MVS.
      - W/S name is the IP name of the related SP adapter.
   b. LINK
      - Device name is the name of the device above.
      - Link name is an arbitrary name that is unique within the profile.
   c. HOME
      - Link name is the name of the above link.
      - Internet address is the IP address associated with the host name.

3.2.2 Customizing CLIO/S

Configuring CLIO/S correctly requires an understanding of the TCP/IP environment and the underlying hardware configuration. There are a number of parameters in CLIO/S that have to accurately relate to definitions in TCP/IP and the hardware. More detailed descriptions can be found in *IBM CLIO/S User' s Guide and Reference CLIO-user-02* .

CLIO/S uses TCP/IP to set up the connection and then uses a channel-to-channel protocol for the data transfer.

You must set up a TCP/IP session as described in 3.2.1, “TCP/IP Customization” on page 43.

CLIO/S can run several data transfers across the same channel simultaneously. Each transfer uses a separate device in MVS. Define as many devices as possible on the channel as this costs very little and gives the greatest flexibility. The number of simultaneous transfers you can drive depends on the transfer rate on each device and this is dependent on the I/O speed of the disk at either end. You are unlikely to be able to run more than six transfers at a time because you will run out of CPU on the node or capacity on the channel.

You therefore need to specify to your hardware planner that you need the following:

1. A control unit of type RS6K.
2. The number of devices.
3. The esoteric name you want. It has to be RS6K followed by one or two digits. It is suggested that the number is the same as the node number.

You will also need the host name of the node. Your TP coordinator should be able to give this to you, or you can find it by typing `hostname` on the command line.

You are then ready to configure CLIO/S as shown in Figure 14 on page 46.

3.2.2.1 Configurating CLIO/S

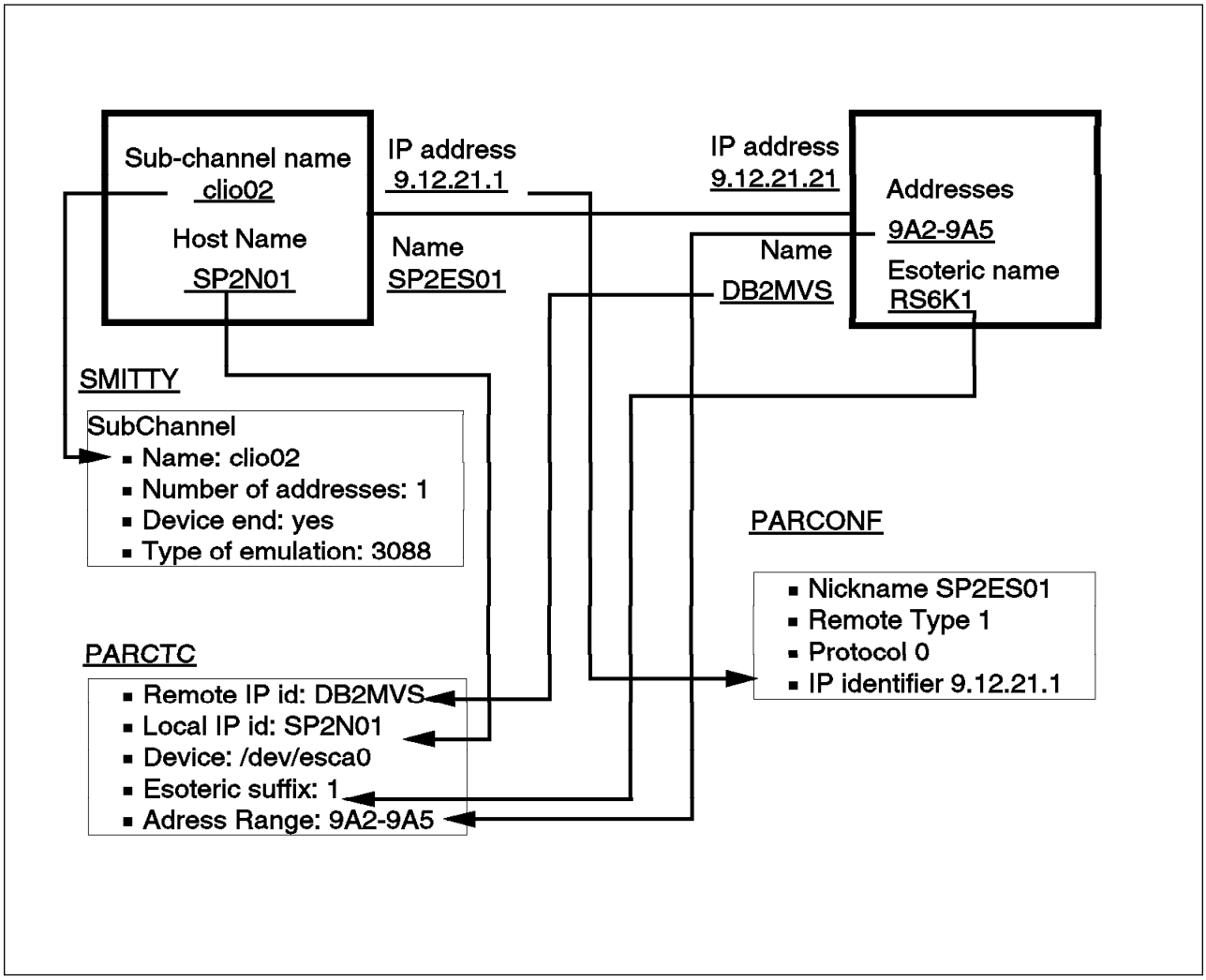


Figure 14. CLIO/S Customization

Note:

1. SMITTY subchannel

   You will have one SMITTY subchannel for each device you have asked for:

   - Name is an arbitrary name, use a name of the form clio02 where the number is the number of the device.
   - Number of addresses is always 1 for CLIO/S.
   - Device end is always yes.
   - Type of emulation is 3088.

2. PARCONF (SP)

   This file is required for any CLIO/S data transfers as well as for CLIO/S tape access. The .netrc is used for obtaining the userid and passwords for the machines specified in parconf.

3. PARCTC (one entry per node)
    - The remote IP ID is the IP name of the MVS system.
    - The local IP ID is the host name of the node (not the IP name of the adapter).
    - Esoteric suffix is the number you gave the hardware planner.
    - The address range you will have been given by the hardware planner. You should ensure you have as many SMITTY subchannels as addresses.
4. PARCONF (MVS)
    - Nickname is an arbitrary name but is simpler if it is the same as the IP name of the channel adapter.
    - Remote type is 1 to signify that you are connected to an RISC/6000 SP.
    - Protocol is 0 to let CLIO/S choose or 4 to insist on ESCON.
    - IP identifier is the IP address of the adapter

If you have set the system up correctly you should now be able to issue a parping command from the node.

If you have any problems, issue the command export CLIODEB=-1 which will set a trace and re-issue the parping command.

3.2.2.2 MVS JCL Procedures for CLIO/S

JCL procedures can be set up to reduce the amount you have to write each time you run a CLIO/S process from batch MVS. The following are two examples:

1. CLFTPB for running CLIO/S FTP in batch
2. CLLINKW for setting up a CLIO/S link to write from MVS to AIX

```
//CLFTPB  PROC NODE=
//*
//* RUN CLIO FTP IN BATCH
//*
//* NODE specifies the target node name
//*
//* Add a SYSIN card for FTP commands
//*
//FCFTP    EXEC PGM=CLFTP,PARM='&NODE'
//STEPLIB  DD   DISP=SHR,DSN=SYS1.SFCFLOAD    CLIO/S SFCFLOAD DATA SET
//         DD   DISP=SHR,DSN=PLI.V2R3M0.SIBMLINK       PLI 70 RUNTIME
//         DD   DISP=SHR,DSN=SYS1.EDC.V2R1M0.SEDCLINK
//SYSPRINT DD SYSOUT=*
//EDCMTF   DD   DISP=SHR,DSN=SYS1.SFCFLOAD    CLIO/S SFCFLOAD Data Set
//         DD   DISP=SHR,DSN=PLI.V2R3M0.SIBMLINK       PLI 70 Runtime
//         DD   DISP=SHR,DSN=SYS1.EDC.V2R1M0.SEDCLINK
//CLIOENV  DD   DSN=ENDY.CLIOENV,DISP=SHR
```

Figure 15. Procedure for Running CLIO/S FTP in Batch

An example of using this to move DB2MVS.CUSTOMER to AIX is show n in Figure 16 on page 48.

```
//FCFTP    EXEC CLFTPB,NODE='SP2ES09'                                    01290001
//SYSIN    DD   *                                                        01300001
echo                                                                    01330001
put 'DB2MVS.CUSTOMER' /db2mvs/customer                                  01340001
quit                                                                    01350001
//                                                                      01360001
```

Figure 16. Example of Using CLFTPB

Figure 17 shows a procedure for writing from MVS to AIX using a LINK.

```
//CLFTPB  PROC DDIN='IN',PIPE='frommvs',NODE=
//*
//* Run CLIO LINK in batch from MVS to AIX
//*
//* Parameters:
//*
//*    DDIN is the ddname for the input to the link
//*         the default name is IN
//*
//*    PIPE is the name of the pipe, or file, on AIX
//*         that the link is attached to
//*         the default is frommvs
//*
//*    NODE is the nodename of the target
//*         there is no default
//*
//CLPLINK  EXEC PGM=CLPLINK,
//  PARM='-r dd:&DDIN -h &NODE -x &PIPE'
//STEPLIB  DD   DISP=SHR,DSN=SYS1.SFCFLOAD    CLIO/S SFCFLOAD DATA SET
//         DD   DISP=SHR,DSN=PLI.V2R3M0.SIBMLINK       PLI 70 RUNTIME
//         DD   DISP=SHR,DSN=SYS1.EDC.V2R1M0.SEDCLINK
//SYSPRINT DD SYSOUT=*
//EDCMTF   DD   DISP=SHR,DSN=SYS1.SFCFLOAD    CLIO/S SFCFLOAD Data Set
//         DD   DISP=SHR,DSN=PLI.V2R3M0.SIBMLINK       PLI 70 Runtime
//         DD   DISP=SHR,DSN=SYS1.EDC.V2R1M0.SEDCLINK
//CLIOENV  DD   DSN=ENDY.CLIOENV,DISP=SHR
```

Figure 17. Procedure CLLINKW for Writing from MVS Via a CLIO/S LINK

Note:

1. The PIPE parameter can define an AIX pipe or a file name, thus CLPLINK can be used to transfer files. It is in fact simpler to use CLPLINK rather than CLPFTP if you want to transfer a single file, because you can do it all in the JCL without the need of a SYSIN file.
2. The

   `-r dd:`

   parameter is used rather than a file name as it enables JCL parameters to be added to the DD card, for example the DD card can include VOL=SER, DISP and DCB information. The DCB information is important when connecting CLPLINK with batch pipes.

Figure 18 on page 49 shows an example of the use of CLLINKW using the defaults so that data is sent from the dd card IN to the AIX pi pe frommvs on node sp2es01.

```
//ENDYLINK JOB (999,POK),'CBIPO INSTALL',NOTIFY=&SYSUID,REGION=20M,     01420001
//             CLASS=A,MSGCLASS=T,MSGLEVEL=(1,1)                        01430001
//FCFTP    EXEC CLLINKW,NODE='sp2es01'
//IN       DD  DSN=DB2MVS.CUSTOMER,DISP=OLD                           01451001
```

Figure 18. Using CLLINKW

3.2.2.3 Operations

When CLFTP or CLPLINK complete successfully they clean up at either end by ending the job on MVS and the process on AIX. If either end ends abnormally, CLIO/S will attempt to clean up both ends, however there are situations where this will not happen and it will be necessary to cancel the job or kill the AIX process.

One situation that will cause this problem is if a job is stopped rather than cancelled. It is important therefore to avoid issuing stops against CLIO/S MVS jobs.

3.2.3 Batch Pipe Installation

The installation of batch pipes is via SMP/E and requires no customization or configuration.

The following recommendations should simplify its use:

- Either end of a batch pipe may start first. The DCB parameters for the pipe will be defined by the end that starts first. Therefore, you should ensure that the DD cards for each end of the pipe are identical.
- To avoid confusion the data set name for a pipe should be different from any existing file and should make it obvious that it is a pipe. For example start all pipes with the qualifier BP01 which is the default subsystem name for batch pipes.

```
//ENDYUNC  JOB (999,POK),NOTIFY=&SYSUID,                              00001001
//         CLASS=A,MSGCLASS=T,MSGLEVEL=(1,1),TIME=1440                00002000
//UNLOAD  EXEC TIAUL                                                  01170000
//SYSIN    DD  DSN=DB2TOPE.UNLOAD(CUSTOMER)                           01340000
//SYSREC00 DD  DSN=BP01.DB2MVS.CUSTOMER,                              01250001
//             DCB=(RECFM=FB,LRECL=249,BUFNO=10),                     01251001
//             SUBSYS=BP01                    <=========PIPE          01270001
//*
//*
//ENDYLINK JOB (999,POK),'CBIPO INSTALL',NOTIFY=&SYSUID,REGION=20M,   01420001
//             CLASS=A,MSGCLASS=T,MSGLEVEL=(1,1)                      01430001
//FCFTP    EXEC CLLINKW,                                              01440001
//  NODE='sp2es01',PIPE='db2mvs.customer.aix'                         01450001
//IN       DD  DSN=BP01.DB2MVS.CUSTOMER,                              01451001
//             DCB=(RECFM=FB,LRECL=249,BUFNO=10),                     01452001
//             SUBSYS=BP01                    <=========PIPE          01453001
```

Figure 19. Example of JCL for a Batch Pipe

Note: The DD cards SYSREC00 (that writes the pipe) and IN (that reads the pipe) are identical.

Chapter 4. Implementing DB2 Parallel Edition on the RISC/6000 SP

The installation of DB2 Parallel Edition on a cluster has already been described in *DB2 Parallel Edition for AIX Administration Guide and Reference,* SC09-1982-00. This section is devoted to what is specific to the RISC/6000 SP environment and to the tools that it provides.

4.1 Preliminary Steps

First, you should do the following:

- Verify if Automounter has been enabled on your system.
- Choose the home directory location for the DB2 Parallel Edition instance owner.
- Consider installation space requirements.
- Create an instance group and owner.
- Propagate file collections.
- Increase *maxuproc* value.
- Verify locale definition.

4.1.1 Automounter

Although Automounter (AMD) is not a prerequisite to running DB2 Parallel Edition on a RISC/6000 SP, it can be found running in many RISC/6000 SP environments.

This section describes how to do the following:

- Verify Automounter enablement on your RISC/6000 SP.
- Start Automounter.

More details on Automounter can be found in *RS/6000 SP System Management: Easy, Lean and Mean,* GG24-2563-00.

4.1.1.1 Verifying Automounter Enablement

1. Start SMIT (or smitty) as root:

```
sp2cw0-root / -> smitty
```

2. Select **9076 SP System Management** from the menu shown in Figure 20 on page 52.

```
                              System Management

Move cursor to desired item and press Enter.

  Software Installation and Maintenance
  Software License Management
  Devices
  System Storage Management (Physical & Logical Storage)
  Security & Users
  Communications Applications and Services
  Print Spooling
  Problem Determination
  Performance & Resource Scheduling
  System Environments
  Processes & Subsystems
  Applications
  9076 SP System Management
  Using SMIT (information only)


F1=Help             F2=Refresh          F3=Cancel           F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 20. Smit System Management Menu

3. Select **9076 SP Configuration Database Management** from the menu shown in Figure 21.

```
                           9076 SP System Management

Move cursor to desired item and press Enter.

  9076 SP Configuration Database Management
  9076 SP Cluster Management
  9076 SP Configuration Information
  9076 SP Users
  9076 SP Installation/Configuration Verification
  Parallel I/O File System Management
  RS/6000 SP Log Management


F1=Help             F2=Refresh          F3=Cancel           F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 21. SP System Management Menu

4. Select **Enter Database Information** from the menu shown in Figure 22 on page 53.

```
                  9076 SP Configuration Database Management

Move cursor to desired item and press Enter.

  Enter Database Information
  List Database Information
  Delete Database Information


F1=Help             F2=Refresh          F3=Cancel           F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 22. SP Database Management Menu

5. Select **Site Environment Information** from the menu shown in Figure 23.

```
                          Enter Database Information

Move cursor to desired item and press Enter.

  Site Environment Information
  Frame Information
  Node Database Information
  System Partition Configuration
  Run setup_server Command


F1=Help             F2=Refresh          F3=Cancel           F8=Image
```

Figure 23. Enter Database Information

6. Check that the AMD configuration value is set to true as shown in the menu in Figure 24 on page 54.

```
                        Site Environment Information

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

[TOP]                                                   [Entry Fields]
  Default Network Install Image                        [bos.obj.ssp.41]
  Remove Install Image after Installs                   false                +

  NTP Installation                                      consensus            +
  NTP Server Hostname(s)                               [""]
  NTP Version                                           3                    +

  AMD Configuration                                     true                 +

  Print Management Configuration                        false                +
  Print system secure mode login name                  [""]

  User Administration Interface                         true                 +
  Password File Server Hostname                        [sp2cw0]
[MORE...12]

F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 24. Site Environment Information Menu

If you find the AMD configuration value set to false, Automounter has not been enabled on your system.

4.1.1.2 Starting the Automounter Daemon

1. Check that the Automounter daemon (amd) is running on all nodes by issuing the following command:

```
sp2cw0-root / -> dsh -a "ps -ef | grep amd"
sp2n01.itsc.pok.ibm.com:     root 11934     1   0   Nov 13      -  0:26  /et
c/amd/amd -t 16.120 -x all -
sp2n01.itsc.pok.ibm.com:     root 16778 27272   2 13:29:29      -  0:00  gre
p amd
 .
 .
sp2n16.itsc.pok.ibm.com:     root  7048 14726   1 13:29:30      -  0:00  gre
p amd
sp2n16.itsc.pok.ibm.com:     root 11898     1   0   Nov 07      -  0:08  /et
c/amd/amd -t 16.120 -x all -
sp2cw0-root / ->
```

Note: The distributed shell (*DSH*) is a RISC/6000 SP command allowing execution of a command on a group of nodes.

Note: The *-a* option indicates that the DSH command will be issued on all the available nodes

2. If the Automounter has been enabled and is not running, start it on the appropriate nodes with the following command:

```
sp2cw0-root / -> dsh -w sp2n05, sp2n11, sp2n15 /etc/amd/amd_start
```

Note: the *-w* option tells the command that it will apply to the list of comma separated nodes indicated after the *-w* option. In our example, the list consists of nodes sp2n05, sp2n11 and sp2n15.

4.1.2 Choosing the Home Directory Location for the Instance Owner

This home directory ($HOME) contains the system database catalog needed for all calls to system tables.

DB2 Parallel Edition requires that it be accessible from all the RISC/6000 SP nodes belonging to the DB2 Parallel Edition database.

The DB2 Parallel Edition user's home directory should be mounted via NFS on all RISC/6000 SP nodes where DB2 Parallel Edition is installed.

The node where the instance owner's home directory ($HOME) is physically created could impact database availability. If this directory cannot be accessed, no access to the database will be possible.

Two locations could be privileged for physical $HOME creation:

- One of the catalog nodes

  The catalog node of a database is the node the CREATE DATABASE command for that database is issued.

  A catalog node is already a node of particular importance since the data it contains is required by DB2 Parallel Edition. If it is down, you will not be able to access its associated database.

  Typically, a catalog node will be chosen if, for query performance reasons, all the catalogs of all the databases of the instance are concentrated on a single node.

  Disk mirroring and frequent backup is recommended.

- The control workstation

  This might be a good place to physically create your instance owner's home directory, especially as the control workstation does not constitute a single point of failure anymore.

  Typically, the control workstation will be chosen if you are not using the High Performance Switch (HPS) and if, for system table call performance reasons, all the catalogs of all the databases of the instance are spread across more than one node.

  Installing High Availability Control Work Station (HACWS) and performing frequent backup is recommended.

We chose to create our DB2 Parallel Edition instance owner's $HOME on a node dedicated to storing the catalogs as we wanted DB2 Parallel Edition to use HPS.

4.1.3 Installation Space Requirements

4.1.3.1 Memory

It is estimated that DB2 Parallel Edition requires a minimum of 110 MB of RAM on each node. Systems running large databases or a great number of users should use 512 MB.

More information on memory requirements can be found in *DB2 Parallel Edition for AIX Administration Guide and Reference,* SC09-1982-00.

4.1.3.2 Disk

DB2 Parallel Edition can be installed in two different ways:

1. On one node.

   The node should then be mounted via NFS on all the other nodes that are to access DB2 Parallel Edition code.

   DB2 Parallel Edition code requires approximately 30 MB on the /usr filesystem of the installed node or on a filesystem mounted over

   `/usr/lpp/db2pe_vv_rr`

   where vv is the version number and rr the release number.

2. On all nodes

   DB2 Parallel Edition code requires approximately 30 MB on the /usr filesystem on each node or on a filesystem mounted over /usr/lpp/db2pe_vv_rr.

The DB2 Parallel Edition Version 1 Release 1 code files are installed under the /usr/lpp/db2pe_01_01 directory. Check that you have the available space.

This 30 MB requirement does not include database space. That part is described in *DB2 Parallel Edition Administration Guide and Reference,* SC09-1982-00.

Each instance requires a minimum of 1.5 Mb in the DB2 Parallel Edition instance owner's $HOME.

As there is only one $HOME per instance, the instance space requirement should not be multiplied by the number of nodes.

4.1.4 Creating an Instance Group on the Control WorkStation

1. Start SMIT as root:

   ```
   sp2cw0-root / -> smitty group
   ```

2. Select **Add a Group**.

3. Indicate a group name for the DB2 Parallel Edition instance in the appropriate field (group NAME).

   ```
   * Group NAME                                          [db2pegrp]
   ```

   The other options can be left to their default values.

4.1.5 Creating an Instance Owner As a 9076 SP User

For the massively parallel RISC/6000 SP to be considered as a unit, user creation must be possible from a single administration point while the created users must exist on all the nodes that constitute the RISC/6000 SP. They must maintain the same working environment from one node to the other (uid, gid, $HOME, etc).

This is partly achieved with the 9076 SP user management commands (/usr/lpp/ssp/bin/spmkuser, /usr/lpp/ssp/bin/sprmuser, etc) whose role, apart from that of regular user management commands, is to prepare the NFS mounting across all nodes.

A similar environment will have to be maintained across all nodes. This can be achieved by propagating to all nodes all changes made to any of the reference environment definition files (/etc/passwd, /etc/group, etc).

The supper RISC/6000 SP command will be used to force the propagation of the changes to all nodes as explained in 4.1.6, “Propagate File Collections” on page 60.

1. Start SMIT as root from the Control Workstation:

```
sp2cw0-root / -> smitty
```

2. Select **9076 SP System Management** from the menu shown in Figure 25.

```
                                System Management

Move cursor to desired item and press Enter.

  Software Installation and Maintenance
  Software License Management
  Devices
  System Storage Management (Physical & Logical Storage)
  Security & Users
  Communications Applications and Services
  Print Spooling
  Problem Determination
  Performance & Resource Scheduling
  System Environments
  Processes & Subsystems
  Applications
  9076 SP System Management
  Using SMIT (information only)

F1=Help             F2=Refresh          F3=Cancel           F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 25. SP System Management

3. Select **9076 SP Users** from the menu shown in Figure 26 on page 58

```
                          9076 SP System Management

Move cursor to desired item and press Enter.

  9076 SP Configuration Database Management
  9076 SP Cluster Management
  9076 SP Configuration Information
  9076 SP Users
  9076 SP Installation/Configuration Verification
  Parallel I/O File System Management
  RS/6000 SP Log Management


F1=Help             F2=Refresh          F3=Cancel           F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 26. Selecting SP User Menu

4. Select **Add a User**.

   Note: This menu is only available from the control workstation.

```
                                9076 SP Users

Move cursor to desired item and press Enter.

  Add a User
  Change/Show Characteristics of a User
  Remove a User

F1=Help             F2=Refresh          F3=Cancel           F8=Image
F9=Shell            F10=Exit            Enter=Do
```

5. Provide the necessary information in the Add a User menu shown in Figure 27.

```
                                  Add a User

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
* User NAME                                          [db2peusr]
  User ID                                            []
  LOGIN user?                                         true                    +
  PRIMARY group                                      [db2pegrp ]              +
  Secondary GROUPS                                   []                       +
  HOME directory                                     []
  Initial PROGRAM                                    []                        /
  User INFORMATION                                   []

F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 27. Add a User Menu

- Choose any user name.
- Choose the group name defined in 4.1.4, “Creating an Instance Group on the Control WorkStation” on page 56.

- Choose a $HOME directory:
  - Default value

    Leaving the $HOME directory blank will create a default 9076 SP user's $HOME in the following:

    ```
    control_ws:/home/control_ws/user_name
    ```

    Where control_ws is the control workstation hostname and user_name the instance owner's username.

    The user's $HOME is physically created on the control workstation.

    The following line would appear in the output of the *df* command:

    ```
    control_ws:(pidXXXX)  0  0  -1%   -  - /u
    ```

    As the user's $HOME is accessed (through *su*, *cd* ... commands), /home/control_ws is mounted over /u.

    The following lines would then appear in the output of the df command:

    ```
    control_ws:(pidXXXX)  0  0  -1%   -  - /u
    control_ws:/home/control_ws  8192  4312  48%   -  - /a/home/control_ws/ho
    me/control_ws
    ```

    For further details concerning Automounter's indirect mounts, see *RS/6000 SP System Management: Easy, Lean and Mean,* GG24-2563-00.

    The user's access path to $HOME then becomes /u/user_name (even on the Control Workstation). It will be mounted using NFS whenever needed and will time out after a certain period of inactivity.

  - Non-default values

    If you specify NODE:/dir1/dir2/user_name as a user's $HOME, NODE being any node but the control workstation, and /dir1 the exported filesystem, the 9076 SP user's $HOME becomes /u/dir2/user_name.

    The following lines would appear in the output of the *df* command if the user's $HOME is being or has recently been accessed:

    ```
    NODE:(pidXXXX)  0  0  -1%   -  - /u
    NODE:/dir1         8192   4312  48%   -  - /a/home/dir1
    ```

  The following default $HOME name structure should be preserved whenever possible:

  ```
  NODE:/home/NODE/user_name
  ```

  Where NODE should be replaced with the hostname of the node you have decided to use in 4.1.2, “Choosing the Home Directory Location for the Instance Owner” on page 55.

If you did not follow the /home/NODE/user_name structure, you may get the following message when creating a 9076 SP User:

```
spmkuser:  0027-122 Warning:  User directory, /dir1, is in an
unexported file system on sp2n10.
```

You should then export the filesystem using the following command:

```
rsh sp2n10 /usr/sbin/mknfsexp -d '/dir1' -t 'rw' '-B'
```

where sp2n10 is the hostname of the node where the user's $HOME has been physically created, and /dir1/user_name is the chosen path indicated as the user's $HOME in step 4.1.2, “Choosing the Home Directory Location for the Instance Owner” on page 55.

Note: Remote commands normally would require a .rhosts file in a user's $HOME. In the case of the RISC/6000 SP, Kerberos is used. The previous command is issued by root from the control workstation.

The root user is referenced as a Kerberos user, thus using the Kerberos version of remote commands that do not need the .rhosts.

An unreferenced user would get a permission denial unless he is referenced in the remote node's .rhosts file.

6. Set the instance owner's password on the control workstation.
7. Login to that user from the control workstation to do the change password on first connection procedure.

4.1.6 Propagate File Collections

The database instance owner is now defined across all nodes, but his environment has not yet been made consistent across all nodes. That can be achieved by using the supper update RISC/6000 SP command.

The supper command should be present on all nodes, but it is not needed on the control workstation.

The supper command relies on a set of file collections indicating which files should be propagated across nodes. These file collections are located in the /var/sysman/sup/lists directory on all nodes, including the control workstation.

Whenever file propagation is triggered, the files mentioned for update in the selected file collections are copied over to the nodes, overwriting those previously present on the nodes.

Note: Do not change any of the files mentioned for update in the file collections from one of the nodes. They will be overwritten at the next file collection propagation.

4.1.6.1 Patience Vs Force

- A supper command is started by the cron daemon:

```
sp2n13-root / -> crontab -l | grep supper
10 * * * *  /var/sysman/supper update sup.admin user.admin
node.root 1>/dev/null 2>/dev/null
```

By default, supper will run every hour, at ten past. It will use the sup.admin and user.admin file collections in /var/sysman/sup/lists. It will propagate the following files:

```
/etc/passwd
/etc/group
/etc/security/group
/etc/security/passwd
/etc/amd/amd-maps/amd.*
```

Your environment will not be consistent until the required files have been propagated.

- Propagation can be forced by issuing the following command from the control workstation:

```
dsh -a /var/sysman/supper update user.admin
```

4.1.7 Increase Default Maxuproc

As DB2 Parallel Edition applications start up a large number of processes, it is recommended to increase the value of the default AIX *maxuproc* parameter to 500. This should be done on all nodes.

Issue the following command:

```
sp2cw0-root / dsh -a chdev -l sys0 -a maxuproc='500'
```

4.1.8 Locale Definition

Verify that the same locale is defined on all nodes of the RISC/6000 SP.

The nodes of a RISC/6000 SP are usually installed from a single image. The locale should then be the same on all nodes unless someone has deliberately changed it.

```
sp2cw0-root / dsh -a echo $LANG
```

4.1.9 Clock Synchronization

DB2 Parallel Edition requires that all nodes be synchronized. On a cluster, the system administrator would have to synchronize the clocks on all the nodes.

The RISC/6000 SP relies on the Network Time Protocol (NTP), a public domain software, to synchronize clocks across nodes. The DB2 Parallel Edition administrator no longer need worrying about it.

4.2 Installing DB2 Parallel Edition Code

The DB2 Parallel Edition code can be installed in the following two ways:

- A shared environment where the libraries of the product are NFS mounted across all nodes. Installation is only done on one node.

  The execution on DB2 Parallel Edition can be completely affected if the node where the code is installed is not accessible.

- A non-shared environment where each node has its own set of libraries. Installation has to be done on every node.

The installation procedure covered in this section allows the creation of a DB2 Parallel Edition non-shared environment on an RISC/6000 SP machine.

DB2 Parallel Edition on an RISC/6000 SP machine will handle very large databases (from tenths of gigabytes to terabytes), therefore disk space is not be a critical resource in this environment. The installation of the DB2 Parallel Edition code on each node is recommended in order to increase the availability of the product libraries and to avoid any communication cost for accessing these modules.

All the steps that follow, needed to create the non-shared environment, are initiated from the Control Workstation under root authority.

1. Ensure that you have at least 30 MB of free space in /usr directory on each RISC/6000 SP node where DB2 Parallel Edition will be installed.

   ```
   sp2cw0-root / -> dsh -a "lsvg rootvg | grep FREE " | pg
   ```

2. Retrieve the DB2 Parallel Edition product libraries you want to install from the installation media and put them on a file system.

   ```
   smit
     Software Installation and Maintenance
        Install and Update Software
   ```

   Figure 28 on page 63 shows the smit menu display.

```
                Copy Software to Hard Disk for Future Installation

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
* INPUT device / directory for software               /dev/cd0
* SOFTWARE package to copy                            [all]
* DIRECTORY for storing software package              [/db2peinstall]
  DIRECTORY for temporary storage during copying      [/tmp]
  EXTEND file systems if space needed?                yes                          +



F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 28. Smit Menu Display

3. Add the file system where you have copied the code to the list of exported file systems.

```
sp2cw0-root / -> mknfsexp -d /db2peinstall -t 'rw' '-B'
```

4. Mount this file system on all nodes where you want to install DB2 Parallel Edition.

   The *dsh* command that enables the parallel execution of remote commands on all nodes (-a option) or groups of nodes (using the collective environment variable, WCOLL) can be used to automate this task in the following way:

   a. Make the WCOLL variable point to the group of nodes you want by:

      1) Getting a list of all the RISC/6000 SP nodes and redirect the output to a file

      2) Editing the file created on the previous step and selecting the RISC/6000 SP nodes where you want to install DB2 Parallel Edition

      3) And setting the collective environment variable to the name of the file you edited

```
sp2cw0-root / -> hostlist -av > ./XYZ123
sp2cw0-root / -> vi ./XYZ123
sp2cw0-root / -> export WCOLL=./XYZ123
```

   b. Create a mount point for the exported file system with the code and mount this file system on the previously created mount point.

```
sp2cw0-root / ->dsh mkdir /db2peinstall
sp2cw0-root / -> dsh mount sp2cw0:/db2peinstall /db2peinstall
```

5. Install the code on each node either using *smit* or the *installp* command.

You can send the *installp* command to all the selected nodes with the dsh command and redirect the output to a file to look at the results.

```
sp2cw0-root / -> dsh "installp -q -a -d /db2peinstall -g -X all" >
/tmp/instdb2pe.log
sp2cw0-root / -> view /tmp/instdb2pe.log
```

Unmount the file system with the DB2 Parallel Edition installable image and remove the mount point you created for the installation.

```
sp2cw0-root / -> dsh umount /db2peinstall
sp2cw0-root / -> dsh rmdir /db2peinstall
```

6. Create an instance of the product on any of the DB2 Parallel Edition nodes. We chose to run the command on node 10 (sp2n10).

```
sp2cw0-root /->rsh sp2n10 '/usr/lpp/db2pe_01_01/instance/db2instance db2peusr'
```

7. Optionally, if you develop applications and want to avoid specifying the full path to the product libraries and include files you can create links on all the DB2 Parallel Edition nodes using the *dsh* command on that group of nodes (using WCOLL collective variable)

```
sp2cw0-root / -> export WCOLL=./XYZ123
sp2cw0-root / -> dsh '/usr/lpp/db2pe_01_01/cfg/db2ln'
```

4.2.1.1 Setting the Environment Variables:

Modify the *.profile* file of the instance owner to include the DB2 Parallel Edition environment variables specific to the instance you have created. You can do this by adding following line to the db2peusr user .profile:

```
. ./sqllib/db2profile
```

4.2.1.2 Creating the Node Configuration File

Before you can start DB2 Parallel Edition you must create the node configuration file *db2nodes.cfg* under $HOME/sqllib, where $HOME is the home directory of the instance owner. This file contains configuration information for all nodes within a DB2 Parallel Edition instance. The script we used to do this is shown in Figure 29 on page 65.

```
# Listing the nodes and switches:
# DON'T FORGET TO PASS THE DB2 Parallel Edition INSTANCE OWNER NAME as $1

if   $# = 0   ; then echo "USAGE: db2cfg.create
      instance_owner_name " ; exit fi

 splstdata -a | awk '/ en0 / { print $1 " " $5 }' | awk 'BEGIN
 {FS="."} {print $1 " 0 "}' > /tmp/XYZ567

 splstdata -a | awk '/ css0 / { print $5 }' | awk 'BEGIN {FS="."}
 {print $1}' > /tmp/XYZ890

 su  $1 "-c paste /tmp/XYZ567 /tmp/XYZ890 >
 /u/$1/sqllib/db2nodes.cfg; vi /u/$1/sqllib/db2nodes.cfg"

 rm /tmp/XYZ567 /tmp/XYZ890
```

Figure 29. Node Configuration Script

4.2.1.3 Enable FCM Communications

To enable internodal communication between the DB2 Parallel Edition nodes, supported by the Fast Communication Manager, you must create a service entry for each logical port number in the */etc/services* file.

If you are using logical nodes in the instance you should define a range of ports as documented in *DB2 Parallel Edition for AIX Administration Guide and Reference,* SC09-1982-00.

We recommend having a central definition of port numbers in the RISC/6000 SP environment and to propagate it to all the RISC/6000 SP nodes using the file collections.

The steps to accomplish this are as follows:

1. Edit the /etc/services file to add services that will enable the FMC communication:

   ```
   DB2_db2peusr      9500/tcp
   ```

   Note: It is possible to define a range of ports for FMC communications but this is only relevant when multiple logical DB2 Parallel Edition nodes are defined (typically in an SMP environment) in the *db2nodes.cfg*.

2. Include the */etc/services* file to the File Collections to get it propagated to all nodes. To do this edit the list file under the /var/sysman/sup/user.admin directory and add:
   upgrade /etc/services

3. Force the propagation of the files in the File Collections:

   ```
   dsh -a /var/sysman/supper upgrade user.admin
   ```

4.2.1.4 Making DB2 Parallel Edition Accessible

In the home directory of the DB2 Parallel Edition instance owner create an *.rhosts*file owned by the instance owner. You can use the sample script shown in Figure 30.

```
# Set the following variables according to your environment
INSTOWNWER=db2peusr
DIRINST=/u/db2peusr

hostlist -av > /tmp/XYZ910
# Edit the hostlist to select the correct DB2 Parallel Edition nodes
vi /tmp/XYZ910

# Create the .rhosts file
su $INSTOWNWER "-c cp /tmp/XYZ123  $DIRINST/.rhosts; chmod 600 $DIRINST/.rhosts"

# Remove the temporary file
rm /tmp/XYZ910
```

Figure 30. Sample Script

4.2.1.5 Starting DB2 Parallel Edition

To start DB2 Parallel Edition use the db2start command.

4.2.1.6 Using DB2 Parallel Edition

All other DB2 Parallel Edition actions (create database, create nodegroup, load, get database manager configuration, etc) apply similarly to a cluster or to a RISC/6000 SP. Refer to *DB2 Parallel Edition for AIX Administration Guide and Reference,* SC09-1982-00 for further information related to using DB2 Parallel Edition or to designing a DB2 Parallel Edition database.

Enabling Database Parameter Consistency: Though this issue is not specific to RISC/6000 SP, a word should be mentioned on database parameter consistency.

Database configuration parameters are set at node level. In order for all nodes to share the same database configuration parameters, you should choose between the two following options:

- Use the *db2_all* command to issue *db2 update configuration database* commands that will be carried out to all the nodes defined in *db2nodes.cfg*:

  ```
  db2_all "db2 update configuration database for XXXX using userexit on"
  ```

  Note: This method imposes on the database administrator to always check database parameter consistency.

- Copy the database configuration file on the catalog node (for example) to an NFS directory and link the database configuration file of all the nodes participating in the database to that particular shared file.

  The database instance owner's directory is already accessible from all the RISC/6000 SP nodes.

  As the database instance owner:

```
rsh sp2n10 "cp /db/dbusr/NODE0000*/SQL00001/SQLDBCON /u/dbusr/SAMPLE.CONF"
db2_all "ln -sf /u/dbusr/SAMPLE.CONF /db/dbusr/NODE0000*/SQL00001/SQLDBCON"
```

sp2n10 is the hostname of the catalog node of the database whose configuration file we want to link, */db* is where the database has been created, */u/dbusr* the instance owner's home directory, and *SAMPLE.CONF* the name we chose so that this database's configuration file would be clearly identified.

Chapter 5. Data Extractions

This chapter examines the different methods of data extractions from the mainframe to RISC/6000 SP with DB2 Parallel Edition.

5.1 Sources of Data Extraction for DB2 Parallel Edition

Data to load the DB2 Parallel Edition tables in RISC/6000 SP can come from the following sources:

- Other operating systems, like MVS/ESA, VM/ESA, VSE/ESA, and so forth
- Data base managers, like DB2 MVS, IMS-DB (DL/I), or others
- Data sets or files managed by different access methods, like VSAM, BSAM, DAM

For the data to be loaded into the DB2 Parallel Edition tables, the following intermediate steps have to be performed, not necessarily in the order shown:

- Extraction of data from its source
- Conversion of data to a format suitable to the next steps
- Optional temporary storage of data at the source system
- Transmission/transportation of data to the RISC/6000 SP system
- Splitting of data for distribution to the selected RISC/6000 SP nodes
- Optional temporary storage of data at the target system
- Loading data to the DB2 Parallel Edition tables, partitioned over the selected RISC/6000 SP nodes

The splitting of data can be performed either at the originating system or at the target system, but the conversion has to be performed before the transmission step. Due to the nature of the required EBCDIC to ASCII conversion, only externalized character data formats are allowed for transmission. Normally, the internal representations of numeric data values are binary, packed decimal, floating point internal format, and so forth. These internal formats are not suitable for transmission to the RISC/6000 SP and they must be converted to external, character formats.

For the extraction/conversion of data from its source, there are the following two alternatives:

1. Extraction and conversion in one step, with one program.
2. Extraction and conversion in two steps, with two programs.

Since it is more productive to accomplish both tasks in one step, the alternatives for the required program are the following:

1. Write your own program.
2. Adapt some existing program, for example the DSNTEP2 PL/I sample program used during the Installation Verification Phase (IVP) of DB2 MVS. It is a dynamic SQL program that support SQL statements, except SELECTs.
3. Use some existing program, utility or tool that may be provided by other manufacturers. Here, we will explore the use of the DSNTIUAL assembler sample program that is part of the DB2 MVS Program Product.

From the many alternatives for data sources, data extraction and conversion, we have experienced and documented the following:

Source of Data and Extraction/conversion Documented

1. DB2 MVS as source of data.
2. DB2 MVS DSNTIAUL sample program for extraction and conversion of data in one step.

5.2 Data Extraction from DB2 MVS Using DSNTIAUL

DSNTIAUL is a sample program used during the Installation Verification Phase (IVP) after DB2 MVS is installed or migrated (the last three letters, AUL, stands for Assembler UnLoad). It is an assembler program, whose source code is part of the DB2 MVS machine readable materials, that processes dynamic SQL. Inputs to DSNTIAUL are SQL statements, including SELECT. Outputs are two types of Basic Sequential Access Method (BSAM) fixed format blocked files, containing data suitable for input to the DB2 MVS LOAD utility:

1. LOAD statements used for input control by the DB2 MVS LOAD utility
2. Data for loading/reloading a DB2 MVS table/view

   Note: In the same run, more than one SELECT statement can be issued to DSNTIAUL, resulting in the same amount of output files containing the corresponding data.

Due to these characteristics, the Assembler UnLoad program DSNTIAUL is very useful to move data between different DB2 MVS subsystems, or between tables in the same DB2 MVS as well.

5.2.1 DSNTIAUL Externalized Data Formats

DSNTIAUL can also produce outputs suited for transmission, splitting, and loading of data, from DB2 MVS to DB2 Parallel Edition. The generated records are a fixed length, no matter whether the table rows to be unloaded contain variable length data or not. This is explained by the fact that the externalized data is intended for input to the DB2 MVS LOAD utility that only accepts fixed column formatting, so far. Then, to produce data suitable for the LOAD CMD command of DB2 Parallel Edition, there is a choice of the following two fixed formats:

1. With delimiter characters
2. Without delimiter characters

In the first case, a selected character is used to delimit or mark the end of every one and all the externalized columns. This implies that a little more work has to be done on the MVS side and that more data will be generated, processed and transmitted from one system to the other.

In the second case, there are savings if no extra data is generated, but the column format has to be coded at the RISC/6000 SP side for the DB2 Parallel Edition LOAD CMD command to work properly.

To examine both alternatives, let's first review what DSNTIUAL produces without further elaboration, that is when simple SELECTs of the following forms are processed:

SELECT * FROM TABLE ownernam.tablenam ...

SELECT colname, ... FROM TABLE ownernam.tablenam ...

Note: The SELECTs can have any of the clauses supported by DB2 MVS, so they can be applied to views as well.

Table 5 shows the format and the space occupied by the different data types, used internally by DB2 MVS and as they are externalized by DSNTIAUL.

Table 5 (Page 1 of 2). DSNTIAUL Externalized Default Formats. The output produced by default is suited for input to the DB2 MVS LOAD utility, but it is not suited for transmission from MVS to RISC/6000 SP due to the binary content of the externalized data.

Note: *The output produced does not have separations between columns, thus no extra space is wasted.*

Data types	DB2 MVS internal format (bytes)	Data externalized by DSNTIAUL (columns/spaces)	Comments
SMALLINT	2	2	Externalized format is binary.
INTEGER or INT	4	4	Externalized format is binary.
DECIMAL(p,s) or DEC(p,s)	(p + 1)/2 fraction rounded up	(p + 1)/2 fraction rounded up	Externalized format is packed decimal with p=precision and s=scale.
DECIMAL(15,s) or DEC(15,s)	8	8	Externalized format is packed decimal with s=scale. **Note:** The recommended precision to declare for a packed decimal is an odd number.
DECIMAL(14,s) or DEC(14,s)	8	8	Externalized format is packed decimal with s=scale. **Note:** An even precision number is represented as if the precision was that of the next upper odd number.
FLOAT(21) or REAL	4	4	Externalized format is equivalent to the internal, full word, floating point representation.
FLOAT(53) or FLOAT or DOUBLE PRECISION	8	8	Externalized format is equivalent to the internal, double word, floating point representation.
CHARACTER or CHAR(n)	n	n	Externalized format is an EBCDIC character string of n=length.

Table 5 (Page 2 of 2). DSNTIAUL Externalized Default Formats. The output produced by default is suited for input to the DB2 MVS LOAD utility, but it is not suited for transmission from MVS to RISC/6000 SP due to the binary content of the externalized data.

Note: *The output produced does not have separations between columns, thus no extra space is wasted.*

Data types	DB2 MVS internal format (bytes)	Data externalized by DSNTIAUL (columns/spaces)	Comments
VARCHAR(n)	2 + n	2 + n	Externalized format is an EBCDIC character string of n=length, preceded by two columns in binary format containing only the length (n) of the corresponding character string. **Note:** If the actual character string length is less than (n), the first two columns still indicate the real length, but the column is externalized occupying the (n) columns, padded with the necessary blanks to the right.
'Character constant of length n'	2 + n	2 + n	Externalized format is the same coded EBCDIC character string of n=length, preceded by two columns in binary format containing only the length (n).
DATE	4	4	Externalized format is equivalent to the internal format YYYYMMDD, using one byte/column/space for every two digits.
TIME	3	3	Externalized format is equivalent to the internal format HHMMSS, using one byte/column/space for every two digits.

5.2.2 Using Built-In Functions (BIF) with DNSTIAUL

Since the externalized data has to be converted to character format (for transmission from the MVS host to RISC/6000 SP, input to the splitter db2split and input to the LOAD CMD of DB2 Parallel Edition), we will now examine the alternatives that DSNTIAUL combined with DB2 MVS built-in functions provide for the following data types:

- INTEGER or INT, and SMALLINT data types, in Table 6.
- DECIMAL(p,s) or DEC(p,s) data type with p=decimal precision and s=decimal scale, in Table 7 on page 75.
- FLOAT(p) data type with p=bit precision, REAL, FLOAT, or DOUBLE PRECISION in Table 8 on page 75.
- CHARACTER(n) or CHAR(n), and VARCHAR(n) data types with n=number of characters, and the creation of column delimiters, in Table 10 on page 78.
- DATE and TIME data types, in Table 11 on page 78.

Table 6 (Page 1 of 2). DSNTIAUL Output Formats for Integer-Binary Data Types. The output produced using DB2 MVS Built-In Functions (BIF) is suited for data transmission, splitting, and loading into DB2 Parallel Edition.

BIFs applied to data types	External format produced	Number of cols./spaces required	Comments
CHAR(DECIMAL(smallint_data_type))	s99999.	7	Externalized format is character. The decimal point (.) is suited for a decimal data type target column in DB2 Parallel Edition, but cannot be loaded into an integer data type. (s) represents a minus sign, (-), when the number is negative and, a blank/space when positive.
CHAR(DECIMAL(integer_data_type))	s09999999999.	13	Externalized format is character. The decimal point (.) and the (s) have the same meaning as in smallint_data_type, above. **Note:** The conversion pads zeroes (0) at the left and you always get one leftmost zero (0) as long as the maximum amount that an integer can accommodate has a precision of 10 but DSNTIAUL provides for 11 places.
SUBSTR(CHAR(DECIMAL(smallint_d_t)),1,6)	s99999	6	Externalized format is character. The absence of the decimal point makes the data suited for integer data type input as well as for decimal input to DB2 Parallel Edition.

Table 6 (Page 2 of 2). DSNTIAUL Output Formats for Integer-Binary Data Types. The output produced using DB2 MVS Built-In Functions (BIF) is suited for data transmission, splitting, and loading into DB2 Parallel Edition.

BIFs applied to data types	**External format produced**	**Number of cols./spaces required**	**Comments**
SUBSTR(CHAR(DECIMAL(integer_d_t)),1,12)	s09999999999	12	Externalized format is character. The absence of the decimal point has the same implications as for smallint_data_type, above.
DIGITS(smallint_data_type)	99999	5	Externalized format is character. The absence of the decimal point makes data suited for integer data type input as well as for decimal to DB2 Parallel Edition, but you lack the minus sign. **Note:** Negative values are not used very often with this data type. This solution uses a simpler and more efficient conversion BIF.
DIGITS(integer_data_type)	9999999999	10	Externalized format is character. The same considerations as for smallint_data_type above, applies.

Table 7. DSNTIAUL Output Formats for Decimal-Packed Data Types. The output produced using DB2 MVS Built-In Functions (BIF) is suited for data transmission, splitting, and loading into DB2 Parallel Edition.

BIFs applied to data types	External format produced	Number of cols./spaces required	Comments
CHAR(decimal_data_type) Assuming DEC(p,s)	Minus sign (-) for negative numbers or one blank/space for positive numbers, followed by (p) significant digits (padded with zeros at left, if necessary), with a decimal point (.) before the last (s) digits.	p + 2	Externalized format is character. The decimal point (.) is suited for decimal data type input. **Note:** There may be zeroes padding to the left to complete the (p + 2) fixed positions.
CHAR(decimal_data_type) Assuming DEC(15,2)	s9999999999999.99	17	Externalized format is character. The decimal point (.) is suited for decimal data type input.
CHAR(decimal_data_type) Assuming DEC(14,2)	s999999999999.99	16	Externalized format is character. The decimal point (.) is suited for decimal data type input. **Note:** There are 16 columns/spaces used to honor the specification when the corresponding DEC(14,2) column was created/added (14 spaces for the digits, plus 1 for the sign and 1 for the decimal point, equals 16).

Table 8 (Page 1 of 2). DSNTIAUL Output Formats for Floating-Point Data Types. The output produced using DB2 MVS Built-In Functions (BIF) is suited for data transmission, splitting, and loading into DB2 Parallel Edition.

Note: *Data is externalized in decimal external format, and not in floating point external format as it would be desired. At present there is not a BIF that allows to convert internal numeric values to external floating point format.*

BIFs applied to data types	External format produced	Number of Cols./spaces required	Comments
CHAR(DECIMAL(float_data_type)) Assuming FLOAT(21) or REAL	s999999999999999.	17	Externalized format is decimal character. The decimal point (.) is suited for decimal data type input. **Note:** The externalized format has the decimal point located at the right side of the string. This means that any fractional part is lost. Any absolute number less than zero (0) becomes zero (0).

Table 8 (Page 2 of 2). DSNTIAUL Output Formats for Floating-Point Data Types. The output produced using DB2 MVS Built-In Functions (BIF) is suited for data transmission, splitting, and loading into DB2 Parallel Edition.

Note: *Data is externalized in decimal external format, and not in floating point external format as it would be desired. At present there is not a BIF that allows to convert internal numeric values to external floating point format.*

BIFs applied to data types	External format produced	Number of Cols./spaces required	Comments
CHAR(DECIMAL(float_data_type)) Assuming FLOAT(53), FLOAT or DOUBLE PRECISION	s999999999999999.	17	Externalized format is decimal character. The decimal point (.) is suited for decimal data type input. **Note:** The externalized format has the same implications as for the above short floating point data type.
CHAR(DECIMAL(float_dta_type),p,s) Assuming any float_data_type: Float(21) or FLOAT(53), REAL or FLOAT, or DOUBLE PRECISION	s999999999.9999999 p=decimal precision d=decimal scale	p + 2	Externalized format is decimal character. The decimal point (.) is suited for decimal data type input. **Note:** To avoid the lost of significance when converting from floating point to decimal, you can work with a combination of the precision (p) and the decimal scale (d), with the following restrictions: 1. p = 0 to 31 2. d = 0 to 21 3. p - d >= 10

Table 9 on page 77 shows some examples for extreme big and small numbers. The results show that you may overcome the problem of moving floating point data without lost of significance in the range of:

- s0.999...E+31 to s0.999...E+00 for FLOAT(53)
- s0.999...E+31 to s0.999...E-10 for FLOAT(21).

Table 9. Examples of DSNTIAUL Output Formats for Floating-Point Data Types. The output produced using DB2 MVS Built-In Functions (BIF) is externalized in decimal external format. Playing with precision (p) and decimal scale (d), and multiplying by 1E+nn or by 1E-nn you can prevent from lost of data significance when the range of the numeric values is not too wide.

Note: *When DSNTIAUL detects that the resulting values may be bigger that the floating point capacity that is 7.2E+75, produces an extra output column to indicate null values, that you have to consider.*

BIFs applied to data types	External format produced	Number of cols./spaces required
CHAR(DECIMAL (-0.12345678901234567E+31,31,0))	-12345678901234600000000000000000.	31 + 2 = 33
CHAR(DECIMAL (-0.12345678901234567E+71 * 1E-40,31,0))	-12345678901234600000000000000000. **Note:** When multiplying by 1E-40, you get the same result as above.	31 + 2 = 33
CHAR(DECIMAL (-0.12345678901234567E+00,31,21))	-0000000000.123456789012346000000	31 + 2 = 33
CHAR(DECIMAL (-0.12345678901234567E-05,31,21))	-0000000000.000012345678900000000	31 + 2 = 33
CHAR(DECIMAL (-0.12345678901234567E-05 * 1E+05,31,21))	-0000000000.123456789012346000000 **Note:** When multiplying by 1E+05, you get the value with the corresponding significant digits.	31 + 2 = 33
CHAR(DECIMAL (-0.12345678901234567E-10,31,21))	-0000000000.000000000012346000000	31 + 2 = 33
CHAR(DECIMAL (-0.12345678901234567E-13,31,21))	-0000000000.000000000000012000000	31 + 2 = 33
CHAR(DECIMAL (-0.12345678901234567E-14,31,21))	-0000000000.000000000000001000000	31 + 2 = 33
CHAR(DECIMAL (-0.12345678901234567E-15,31,21))	-0000000000.000000000000000000000	31 + 2 = 33
CHAR(DECIMAL (-0.12345678901234567E-15 * 1E+15,31,21))	-0000000000.123456789012346000000 **Note:** Compare with above example. The significant data is not lost when you multiply the number by 1E+15.	31 + 2 = 33

Table 10. DSNTIAUL Output Formats for Character Data Types. The output produced using DB2 MVS Built-In Functions (BIF) against VARCHAR data types and for delimiters is suited for data transmission, splitting, and loading into DB2 Parallel Edition.

Note: *CHAR data type does not require conversion.*

BIFs applied to data types	External format produced	Number of cols./spaces required	Comments
Character_data_type Assuming CHAR(n)	CCC...CCC	n	Externalized format is character string of n=length. **Note:** No BIF required.
SUBSTR(varchar_data_type,1,m) Assuming VARCHAR(n)	CCC...CCC	m	Externalized format is character string of m=length. The use of the SUBSTRing BIF eliminates the presence of the two binary column/spaces used by default by DSNTIAUL for the string length indication. **Note:** You may play with (m) > = < (n) to get a string longer (padded with blanks to the right), same size (still padded with blanks up to the m=n=length if the stored string is shorter than the maximum length of the column), or shorter (truncated to m=length), than the stored string.
SUBSRT('d',1,1)	d	1	Externalized format is the character (d) of length one, used as delimiter. Allowed delimiters for the DB2 Parallel Edition LOAD CMD command are: " % & ' ( ) * , . / : , = > ? _ | **Note:** A simpler solution would be to include a character constant like '|' after each column but, DSNTIUAL externalizes three columns/spaces, the first two in binary format indicating a length of one, followed by the desired character constant.

Table 11. DSNTIAUL Output Formats for Date and Time Data Types. There is no need to use DB2 MVS Built-In Functions (BIF). The externalized data is suited for data transmission, splitting, and loading into DB2 Parallel Edition, unless your installation uses a LOCAL format for the time and/or data types that is not compatible. In this case, you may have to use the formatting functions ISO, USA, EUR or JIS, accordingly.

BIFs applied to data types	External format produced	Number of cols./spaces required	Comments
date_data_type Assuming ISO format	YYY-MM-DD	10	Externalized format is character string of length 10. No BIF or date formatting required.
time_data_type Assuming ISO format	HH.MM.SS	8	Externalized format is character string of length eight. No BIF or date formatting required.

5.2.3 Application of DSNTIAUL

The following excerpt from the table DB2MVS.ORDERS in DB2 MVS is the source of data for the target table DB2PE.ORDERS in DB2 Parallel Edition:

DB2MVS.ORDERS Excerpt

```
CREATE TABLE DB2MVS.ORDERS
             (O-ORDERKEY      INTEGER       NOT NULL WITH DEFAULT
             ,O_CUSTKEY       INTEGER       NOT NULL WITH DEFAULT
             ,O_ORDERSTATUS   CHAR   (01)   NOT NULL WITH DEFAULT
             ,O_TOTALPRICE    DECIMAL(12,2) NOT NULL WITH DEFAULT
             ,O_ORDERDATE     DATE          NOT NULL WITH DEFAULT
             ,O_ORDERPRIORITY CHAR   (15)   NOT NULL WITH DEFAULT
             ,O_CLERK         CHAR   (15)   NOT NULL WITH DEFAULT
             ,O_SHIPPRIORITY  INTEGER       NOT NULL WITH DEFAULT
             ,O_COMMENT       VARCHAR(79)   NOT NULL WITH DEFAULT ...
             )
       IN DBDB2MVS.TSORDERS ...
;
```

5.2.3.1 Application of DSNTIAUL with Delimiters

The statement we used for DSNTIAUL with delimiters was:

SELECT Statement with Delimiters for DSNTIAUL

```
SELECT
        DIGITS(O-ORDERKEY)
       ,SUBSTR('|',1,1)
       ,DIGITS(O_CUSTKEY)
       ,SUBSTR('|',1,1)
       ,       O_ORDERSTATUS
       ,SUBSTR('|',1,1)
       ,CHAR  (O_TOTALPRICE)
       ,SUBSTR('|',1,1)
       ,       O_ORDERDATE
       ,SUBSTR('|',1,1)
       ,       O_ORDERPRIORITY
       ,SUBSTR('|',1,1)
       ,       O_CLERK
       ,SUBSTR('|',1,1)
       ,DIGITS(O_SHIPPRIORITY)
       ,SUBSTR('|',1,1)
       ,SUBSTR(O_COMMENT,1,79)
       ,SUBSTR('|',1,1)
       FROM DB2MVS.ORDERS
;
```

From the above we can calculate and add up the space required for every column and the corresponding delimiters to know the exact fixed record length that will be produced by DSNTIAUL. This fixed record length figure can be used to calculate, select and specify optimum block sizes, intermediate storage space (if required), and data volumes for transmission. The result is:

Record of Fixed Length Produced by DSNTIAUL

```
Columns                    Space required (EBCDIC characters)

O-ORDERKEY                                  10
O_CUSTKEY                                   10
O_ORDERSTATUS                                1
O_TOTALPRICE                                14
O_ORDERDATE                                 10
O_ORDERPRIORITY                             15
O_CLERK                                     15
O_SHIPPRIORITY                              10
O_COMMENT                                   79
Delimiters (1 per column)                    9
_________________________                  ___
TOTALS                                     173
```

In this case, with the column delimiters approach, there are nine more columns/spaces to handle, store, and transmit apart from the data itself. An extra 5.5%, that for large numbers of rows can be significant but makes the coding of the DB2 Parallel Edition LOAD CMD command as simple as follows:

DB2 Parallel Edition LOAD CMD Command Using Delimiters

```
LOAD FROM filename OF DEL MODIFIED BY COLDEL |
     REPLACE INTO DB2PE.ORDERS
```

Note: The fact that DSNTIAUL produces records of fixed length, even with VARCHAR columns, means that padding blanks will reach the DB2 Parallel Edition LOAD CMD command, but this will ignore blanks loading the right VARCHAR into the corresponding table column.

5.2.3.2 Application of DSNTIAUL without Delimiters

The case without delimiters is trivial. For the same table DB2MVS.ORDER, the SELECT statement for DSNTIAUL would be:

SELECT Statement without Delimiters for DSNTIAUL

```
SELECT
        DIGITS(O-ORDERKEY)
       ,DIGITS(O_CUSTKEY)
       ,       O_ORDERSTATUS
       ,CHAR  (O_TOTALPRICE)
       ,       O_ORDERDATE
       ,       O_ORDERPRIORITY
       ,       O_CLERK
       ,DIGITS(O_SHIPPRIORITY)
       ,SUBSTR(O_COMMENT,1,79)
       FROM DB2MVS.ORDERS
;
```

In this case the fixed record length is 164 (173 minus 9), and the corresponding DB2 Parallel Edition LOAD CMD command must be coded as follows:

DB2 Parallel Edition LOAD CMD Command without Delimiters

```
LOAD FROM filename OF ASC MODIFIED BY T
     METHOD L (1 10, 11 20, 21 21, 22 35, 36 45,
                    46 60, 61 75, 71 85, 86 164)
     REPLACE INTO DB2PE.ORDERS
```

Note: The exact column positions must be calculated and coded for the DB2 Parallel Edition LOAD CMD command as it is done for the DB2 MVS LOAD utility.

5.2.3.3 Intermediate Storage Space for DB2MVS.ORDERS

The following are some attributes of table DB2MVS.ORDERS:

- Number of columns per row: 9
- DB2 MVS internal row length: 143
- DSNTIAUL externalized record length (with delimiters): 173
- Number of rows: 12,000,000

One of the data transmission alternatives experienced was that of using 3390 DASD devices at the MVS host as intermediate storage, with which we can get an optimal maximum of 2 blocks of 27,998 bytes per track, or 2 blocks/track * 15 tracks/cylinder => 30 blocks/cylinder.

According to these, we need the following space to store the content of the whole table in 3390 DASD:

- 27,998 bytes/block / 173 bytes/record => 161 records/block, truncated
- 161 records/block * 173 bytes/record => 27,853 bytes/block
- 161 records/block * 30 blocks/cylinder => 4,830 records/cylinder
- 12,000,000 records / 4,830 records/cylinder => 2,485 cylinders, rounded up

Based on the above, the DSNTIAUL job control language (JCL) statements to externalize data from table DB2MVS.ORDERS are:

Job to Run DSNTIAUL against Table DB2MVS.ORDERS

```
//...      JOB ...
//JOBLIB   DD  DSN=DSN410.SDSNLOAD,DISP=SHR
//UNLOAD   EXEC PGM=IKJEFT01,DYNAMNBR=20
//SYSTSIN  DD  *
           DSN SYSTEM(DB41)
           RUN PROGRAM(DSNTIAUL) PLAN(DSNTIB41) PARMS('SQL')
               LIB('DSN410.RUNLIB.LOAD')
//SYSPRINT DD  *
//SYSUDUMP DD  *
//SYSPUNCH DD  *
//SYSREC00 DD  DSN=DB2MVS.ORDER,DISP=(,CATLG),UNIT=SYSDA,
//            VOL=SER=DBPE02,SPACE=(CYL,(2485,0),RLSE)
//SYSREC01 DD  ...    |
//SYSREC02 DD  ...    |   optional
//...      ... ...    |
//SYSREC99 DD  ...    |
//SYSIN    DD  *
           LOCK TABLE DB2MVS.ORDER IN EXCLUSIVE MODE
           SELECT
                  DIGITS(O-ORDERKEY)
                 ,SUBSTR('|',1,1)
                 ,DIGITS(O_CUSTKEY)
                 ,SUBSTR('|',1,1)
                 ,       O_ORDERSTATUS
                 ,SUBSTR('|',1,1)
                 ,CHAR  (O_TOTALPRICE)
                 ,SUBSTR('|',1,1)
                 ,       O_ORDERDATE
                 ,SUBSTR('|',1,1)
                 ,       O_ORDERPRIORITY
                 ,SUBSTR('|',1,1)
                 ,       O_CLERK
                 ,SUBSTR('|',1,1)
                 ,DIGITS(O_SHIPPRIORITY)
                 ,SUBSTR('|',1,1)
                 ,SUBSTR(O_COMMENT,1,79)
                 ,SUBSTR('|',1,1)
                 FROM DB2MVS.ORDERS;
           SELECT ... |
           SELECT ... |   optional
           ...    ... |
           SELECT ... |
//*
```

The parameter PARMS('SQL') in the invocation of DSNTIAUL directs the program to honor complete SQL SELECT statements up to the length allowed by DB2 MVS, otherwise DSNTIAUL only expects to receive lines of 72 bytes/characters each containing a table name, followed by an optional WHERE predicate.

The record length and the block length are calculated and optimized by DSNTIAUL, so you do not have to provide for them. The above calculations are used merely to provide an adequate number of cylinders for the whole file. In fact, the record length created by DSNTIAUL was 27,853 Bytes length and the blocking factor was 161 records/block. The file fit fine into the 2485 cylinders space provided.

To be on the safe side and not waste space, the cylinder amount can be coded larger than necessary, and the RLSE parameter will shorten the data set to the space really occupied by data at the time the data set is closed.

The optional //SYSREC01 to //SYSREC99 are DDs that relate one-to-one to the corresponding optional SELECT statements that follow the first one. This allows you to request DSNTIAUL to unload more than one table in one run.

Note: This unload will be serial, one SELECT statement executed at a time. The SYSRECnn DDs can be used to direct the output to another type of data set, like tape/cartridge, or to a pipe that can be directed to another process, like splitting or transmission.

The assignment of SYSPUNCH to printer, instead of addressing it to a BSAM data set is recommended for this case, since the DB2 MVS LOAD command directions will not be used for loading data to DB2 MVS. They may be used to code the DB2 Parallel Edition LOAD CMD command control statement when you select the option of loading data to DB2 Parallel Edition without using delimiters. The SYSPUNCH output produced for the unload of table DB2MVS.ORDERS with delimiters was:

DSNTIAUL SYSPUNCH Output for DB2MVS.ORDERS with Delimiters

```
LOAD DATA LOG NO INDDN SYSREC00 INTO TABLE
    TBLNAME
 (
 " "                                POSITION(       1         )
 CHAR(                    10) ,
 " "                                POSITION(      11         )
 CHAR(                     1) ,
 " "                                POSITION(      12         )
 CHAR(                    10) ,
 " "                                POSITION(      22         )
 CHAR(                     1) ,
 " "                                POSITION(      23         )
 CHAR(                     1) ,
 " "                                POSITION(      24         )
 CHAR(                     1) ,
 " "                                POSITION(      25         )
 CHAR(                    14) ,
 " "                                POSITION(      39         )
 CHAR(                     1) ,
 " "                                POSITION(      40         )
 CHAR(                    10) ,
 " "                                POSITION(      50         )
 CHAR(                     1) ,
 " "                                POSITION(      51         )
 CHAR(                    15) ,
 " "                                POSITION(      66         )
 CHAR(                     1) ,
 " "                                POSITION(      67         )
 CHAR(                    15) ,
 " "                                POSITION(      82         )
 CHAR(                     1) ,
 " "                                POSITION(      83         )
 CHAR(                    10) ,
 " "                                POSITION(      93         )
 CHAR(                     1) ,
 " "                                POSITION(      94         )
 CHAR(                    79) ,
 " "                                POSITION(     173         )
 CHAR(                     1) ,
 )
```

The above printout is of no use when you load data to DB2 Parallel Edition with delimiters and is also too crowded with the extra positions for the delimiters themselves. If the SELECT statement is executed without delimiters, the result is useful and much simpler:

??

```
LOAD DATA LOG NO INDDN SYSREC00 INTO TABLE
    TBLNAME
 (
 " "                                    POSITION(       1         )
 CHAR(                     10) ,
 " "                                    POSITION(      11         )
 CHAR(                     10) ,
 " "                                    POSITION(      21         )
 CHAR(                      1) ,
 " "                                    POSITION(      22         )
 CHAR(                     14) ,
 " "                                    POSITION(      36         )
 CHAR(                     10) ,
 " "                                    POSITION(      46         )
 CHAR(                     15) ,
 " "                                    POSITION(      61         )
 CHAR(                     15) ,
 " "                                    POSITION(      76         )
 CHAR(                     10) ,
 " "                                    POSITION(      86         )
 CHAR(                     79) ,
 )
```

5.2.3.4 Intermediate Storage Space Required for MVS

Table 12 shows the column table definitions at the DB2 MVS side, the number of records extracted/converted, and the number of 3390 cylinders required for intermediate storage at the MVS site. The calculation were made following the same procedure used in 5.2.3.3, “Intermediate Storage Space for DB2MVS.ORDERS” on page 81.

Table 12 (Page 1 of 3). DB2MVS.tablenames, Record Lengths and 3390 Cylinders Required. Once the former calculations are applied to the corresponding tables, we get the required minimum number of 3390 cylinders for intermediate storage.

Notes:

1. The DB2 MVS internal row length considers the row overhead and represents the maximum possible value, as if all the variable length columns in the row have their maximum lengths occupied.
2. Only an excerpt of the DB2MVS table definitions have been documented to show the columns attributes in the first column. All the columns have been defined as NOT NULL WITH DEFAULT.
3. A 3390 volume holds up to 3,337 cylinders of data, 15 tracks/cylinder, with a maximum block size of 27,998 bytes/block, 2 blocks/track.

DB2MVS table column definitions	Number of columns	Number of rows/records	DB2 MVS internal row length	DSNTIAUL record length with delimiters	3390 cylinders for intermediate storage
Table DB2MVS.NATION: N_NATIONKEY INTEGER N_NAME CHAR(25) N_REGIONKEY INTEGER N_COMMENT VARCHAR(152)	4	25	195	201	1
Table DB2MVS.REGION: R_REGIONKEY INTEGER R_NAME CHAR(25) R_COMMENT VARCHAR(152)	3	5	191	190	1
Table DB2MVS.PART: P_PARTKEY INTEGER P_NAME VARCHAR(55) P_MFGR CHAR(25) P_BRAND CHAR(10) P_TYPE VARCHAR(25) P_SIZE INTEGER P_CONTAINER CHAR(10) P_RETAILPRICE DECIMAL(12,2) P_COMMENT VARCHAR(23)	9	1,600,000	177	191	366
Table DB2MVS.SUPPLIER: S_SUPPKEY INTEGER S_NAME CHAR(25) S_ADDRESS VARCHAR(40) S_NATIONKEY INTEGER S_PHONE CHAR(15) S_ACCTBAL DECIMAL(12,2) S_COMMENT VARCHAR(199)	7	80,000	208	222	22

Table 12 (Page 2 of 3). DB2MVS.tablenames, Record Lengths and 3390 Cylinders Required. Once the former calculations are applied to the corresponding tables, we get the required minimum number of 3390 cylinders for intermediate storage.

Notes:

1. The DB2 MVS internal row length considers the row overhead and represents the maximum possible value, as if all the variable length columns in the row have their maximum lengths occupied.
2. Only an excerpt of the DB2MVS table definitions have been documented to show the columns attributes in the first column. All the columns have been defined as NOT NULL WITH DEFAULT.
3. A 3390 volume holds up to 3,337 cylinders of data, 15 tracks/cylinder, with a maximum block size of 27,998 bytes/block, 2 blocks/track.

DB2MVS table column definitions	Number of columns	Number of rows/records	DB2 MVS internal row length	DSNTIAUL record length with delimiters	3390 cylinders for intermediate storage
Table DB2MVS.PARTSUPP: PS_PARTKEY INTEGER PS_SUPPKEY INTEGER PS_AVAILQTY INTEGER PS_SUPPLYCOST DECIMAL(12,2) PS_COMMENT VARCHAR(199)	5	6,400,000	228	248	1905
Table DB2MVS.CUSTOMER: C_CUSTKEY INTEGER C_NAME VARCHAR(25) C_ADDRESS VARCHAR(40) C_NATIONKEY INTEGER C_PHONE CHAR(15) C_ACCTBAL DECIMAL(12,2) C_MKTSEGMENT CHAR(10) C_COMMENT VARCHAR(117)	8	1,200,000	236	249	358
Table DB2MVS.ORDERS: O_ORDERKEY INTEGER O_CUSTKEY INTEGER O_ORDERSTATUS CHAR(1) O_TOTALPRICE DECIMAL(12,2) O_ORDERDATE DATE O_ORDERPRIORITY CHAR(15) O_CLERK CHAR(15) O_SHIPPRIORITY INTEGER O_COMMENT VARCHAR(79)	9	12,000,000	143	173	2485

Table 12 (Page 3 of 3). DB2MVS.tablenames, Record Lengths and 3390 Cylinders Required. Once the former calculations are applied to the corresponding tables, we get the required minimum number of 3390 cylinders for intermediate storage.

Notes:

1. The DB2 MVS internal row length considers the row overhead and represents the maximum possible value, as if all the variable length columns in the row have their maximum lengths occupied.
2. Only an excerpt of the DB2MVS table definitions have been documented to show the columns attributes in the first column. All the columns have been defined as NOT NULL WITH DEFAULT.
3. A 3390 volume holds up to 3,337 cylinders of data, 15 tracks/cylinder, with a maximum block size of 27,998 bytes/block, 2 blocks/track.

DB2MVS table column definitions	Number of columns	Number of rows/records	DB2 MVS internal row length	DSNTIAUL record length with delimiters	3390 cylinders for intermediate storage
Table DB2MVS.LINEITEM: L_ORDERKEY INTEGER L_PARTKEY INTEGER L_SUPPKEY INTEGER L_LINENUMBER INTEGER L_QUANTITY DECIMAL(12,2) L_EXTENDEDPRICE DECIMAL(12,2) L_DISCOUNT DECIMAL(12,2) L_TAX DECIMAL(12,2) L_RETURNFLAG CHAR(1) L_LINESTATUS CHAR(1) L_SHIPDATE DATE L_COMMITDATE DATE L_RECEIPTDATE DATE L_SHIPINSTRUCT CHAR(25) L_SHIPMODE CHAR(10) L_COMMENT VARCHAR(44)	16	47,989,007	147	223	12,798 **Note:** This space required three full 3390 volumes plus 2,787 cylinders in a fourth one: DBPE03, DBPE04, DBPE05 and DBPE06.

5.2.3.5 Pros and Cons of Using DSNTIAUL

The following are the pros of using DSNTIAUL:

Reliability Since it is a program provided with the DB2 MVS Program Product materials, it is fully supported as an updated component of DB2 MVS.

Performance It is written in Assembler language for people that know the DB2 MVS product.

Usability It is popular and widely used by the DB2 MVS community for transfer of data among DB2 MVS systems.

Features The ability to realize and use the required record length and the type of the output device, producing the best blocksize from the combined information, relieving the user from these details.

Price None. It is part of the Program Product materials.

Availability It is ready as part of the Installation Verification Procedures (IVPs) during DB2 MVS installation or migration.

The following are the cons of using DSNTIAUL:

Fixed format Either you use delimited ASCII format (DEL) or non-delimited ASCII format (ASC), the externalized data is fixed format.

Note: This is not a problem for the receiving, variable length columns of the DB2 Parallel Edition. Tables will still get only the usable data.

DB2 MVS BIF The use of built-in functions is a little tricky but the combinations are few and are well documented in this book.

Floating Point The externalization of this format has the problems stated in Table 8 on page 75 and Table 9 on page 77.

5.3 Splitting Data in MVS

One of the alternatives you have is to split data in MVS before transmitting them to the RISC/6000 SP side for loading into the DB2 Parallel Edition nodes. For this to be accomplished, you have to do the following:

1. Externalize data in a Basic Sequential Access Method (BSAM) data set for splitting
2. Prepare the db2split program, which is coded in C language, at the MVS environment
3. Set the job for running db2split and run it under MVS

Note: The splitter program is written as db2split in DB2 Parallel Edition. Here we will refer to the splitter version in MVS as DB2SPLIT.

One way of performing step 1 was described in 5.2.3, “Application of DSNTIAUL” on page 79. The output so created is ready, either for transmission to the RISC/6000 SP and splitting at that environment, or for splitting at the MVS environment followed by the transmission. We are now describe the other two steps required to split data in MVS.

5.4 Program Preparation for DB2SPLIT in MVS

To prepare the DB2SPLIT program in MVS you have to follow these steps:

- Transmit selected JCL, source, header and object components from RISC/6000 SP to MVS
- Modify and run the JCL files BUILDLIB and COMPILE

Note: There is an alternative to the above consistent in transmitting an executable of the db2split C program module to MVS. We do not recommend this and favor the compilation of the program using the actual C libraries at MVS to avoid incompatibilities.

5.4.1 Transmitting Components for DB2SPLIT Program Preparation

The files to be transmitted and the procedure we used are as follows:

1. Transmit BUILDLIB.JCL, COMPILE.JCL and GO.JCL scripts from RISC/6000 SP to an MVS library. The MVS library can be any Partitioned Organization (PO) MVS data set with attributes RECFM=FB and LRECL=80. The scripts are located in the DB2 Parallel Edition /usr/lpp/db2pe_01_01/samples/splitter directory.

The sequence for transmitting the script, effected from the RISC/6000 SP side was:

- FTP IP-address
- UserID
- Password
- cd /usr/lpp/db2pe_01_01/samples/splitter
- Binary
- get BUILDLIB.JCL 'DB2TOPE.PROJECT.JCL(BUILDLIB)'

The same procedure was repeated for COMPILE.JCL and GO.JCL. The GO.JCL will be used for executing DB2SPLIT once it is prepared.

2. Transmit the source files rdinput.c, init.c, sqludchd.c, partitn.c, optmap.c, cleanup.c and rdconfig.c found in the /usr/lpp/db2pe_01_01/samples/splitter directory.

   Note: Some of the lines in these scripts are more than 80 characters/columns wide. To avoid truncation or overflow of the lines we received the members in an existent MVS PO library with RECFM=FB and LRECL=128, named DB2TOPE.PROJECT.SOURCE.

   The sequence for transmitting the script, effected from the RISC/6000 SP side was:

   - FTP IP=address
   - UserID
   - Password
   - cd /usr/lpp/db2pe_01_01/samples/splitter
   - Binary
   - get rdinput.c 'DB2TOPE.PROJECT.SOURCE(RDINPUT)'

   The same procedures was repeated for the other source files.

3. Transmit the header files db2split.h, sqlca.h, sqluti.h, sqlsystm.h, sql.h, and cp500850.h for code page 850 (or cp500819 for code page 810), located in the /usr/lpp/db2pe_01_01/include directory.

   Note: Some of the lines in these scripts are more than 80 characters/columns wide. To avoid truncation or overflow of the lines we received the members in a PO library with RECFM=FB and LRECL=128, named DB2TOPE.PROJECT.INCLUDE. We selected the code page 850 for is the most widely used.

   The sequence for transmitting the script, effected from the RISC/6000 SP side was:

   - FTP IP-address
   - UserID
   - Password
   - cd /usr/lpp/db2pe_01_01/include
   - Binary
   - get db2split.h 'DB2TOPE.PROJECT.INCLUDE(DB2SPLIT)'

   The same procedure was repeated for the other header files.

4. Transmit the object modules @@DC370$, SQLEGSCA, SQLKBHSH, SQLRXDCN, SQLRXDTS, SQLUAPI, SQLUGRPI, and SQLVCONV that will be used during the linkage editing of the DB2SPLIT program under MVS. These modules are located in the /usr/lpp/db2pe_01_01/sqllib/lib directory under one single member named grpic370.mvsobj for the C/370 compiler (or

grpiad.mvsobj for the AD/CYCLE compiler). We selected the modules for the C/370 compiler.

Note: There is more than one way to transmit the modules. The procedure we are documenting worked fine when executed from the MVS side. None of the MVS libraries was created in advance.

The sequence for transmitting the modules, effected from the MVS side, at the MVS ISPF Command Shell panel, was:

- FTP IP-address
- UserID
- Password
- cd /usr/lpp/db2pe_01_01/sqllib/lib directory under one
- Binary
- Locsite REC=FB LR=80 BL=3120
- get grpic370.mvsobj 'DB2TOPE.DB2SPLIT.RECEIVE'
- Quit
- Receive indataset('DB2TOPE.DB2SPLIT.RECEIVE')
- Restore dsn
- 'DB2TOPE.GRPIAPI.OBJ'

The above procedure loads the eight modules at once, and creates both data sets.

Notes:

a. The Block Length BL=3120 parameter for the locsite command documented in the README DB2 Parallel Edition file, was not accepted by the system. Instead, the value BL=6080 was automatically set, followed by the messages EZA2360W Operand not allowed on BLOCKS option. Option ignored., and EZA2280W ... blocksize set to 6080.

b. If the data sets are created in advance, they must have the following attributes: RECFM=FB and LRECL=80. The organizations must be Physical Sequential (PS) for the RECEIVE data set and Partitioned Organization (PO) for the OBJ data set.

5.4.2 Modifying and Running JCL for DB2SPLIT Program Preparation

The two final steps to prepare the DB2SPLIT program must be run in the following order:

1. Modification and execution of BUILDLIB job

   Job BUILDLIB uses the C/370 compiler procedures EDCCLIB and EDCLIB. The first, compiles the source modules and loads the object modules to DB2TOPE.PROJECT.OBJ. EDCLIB runs the object library utility that loads the language environment module @@DC370$ for MVS. We modified the job as follows:

Job BUILDLIB Compiles Required Modules for DB2SPLIT

```
//...     JOB ...
//STEP1   EXEC EDCCLIB,CPARM='NOSEQ,NOMAR,LONGNAME,DEF(VMMVS)',
//             INFILE='DB2TOPE.PROJECT.SOURCE(RDINPUT)',
//             LIBRARY='DB2TOPE.PROJECT.OBJ',MEMBER='RDINPUT'
//USERLIB DD   DSNAME='DB2TOPE.PROJECT.INCLUDE',DISP=SHR
//*
//STEP2   EXEC EDCCLIB,CPARM='NOSEQ,NOMAR,LONGNAME,DEF(VMMVS)',
//             INFILE='DB2TOPE.PROJECT.SOURCE(INIT)',
//             LIBRARY='DB2TOPE.PROJECT.OBJ',MEMBER='INIT'
//USERLIB DD   DSNAME='DB2TOPE.PROJECT.INCLUDE',DISP=SHR
//*
//STEP3   EXEC EDCCLIB,CPARM='NOSEQ,NOMAR,LONGNAME,DEF(VMMVS)',
//             INFILE='DB2TOPE.PROJECT.SOURCE(PARTITN)',
//             LIBRARY='DB2TOPE.PROJECT.OBJ',MEMBER='PARTITN'
//USERLIB DD   DSNAME='DB2TOPE.PROJECT.INCLUDE',DISP=SHR
//*
//STEP4   EXEC EDCCLIB,CPARM='NOSEQ,NOMAR,LONGNAME,DEF(VMMVS)',
//             INFILE='DB2TOPE.PROJECT.SOURCE(OPTMAP)',
//             LIBRARY='DB2TOPE.PROJECT.OBJ',MEMBER='OPTMAP'
//USERLIB DD   DSNAME='DB2TOPE.PROJECT.INCLUDE',DISP=SHR
//*
//STEP5   EXEC EDCCLIB,CPARM='NOSEQ,NOMAR,LONGNAME,DEF(VMMVS)',
//             INFILE='DB2TOPE.PROJECT.SOURCE(RDCONFIG)',
//             LIBRARY='DB2TOPE.PROJECT.OBJ',MEMBER='RDCONFIG'
//USERLIB DD   DSNAME='DB2TOPE.PROJECT.INCLUDE',DISP=SHR
//*
//STEP6   EXEC EDCCLIB,CPARM='NOSEQ,NOMAR,LONGNAME,DEF(VMMVS)',
//             INFILE='DB2TOPE.PROJECT.SOURCE(SQLUDCHD)',
//             LIBRARY='DB2TOPE.PROJECT.OBJ',MEMBER='SQLUDCHD'
//USERLIB DD   DSNAME='DB2TOPE.PROJECT.INCLUDE',DISP=SHR
//*
//STEP7   EXEC EDCCLIB,CPARM='NOSEQ,NOMAR,LONGNAME,DEF(VMMVS)',
//             INFILE='DB2TOPE.PROJECT.SOURCE(CLEANUP)',
//             LIBRARY='DB2TOPE.PROJECT.OBJ',MEMBER='CLEANUP'
//USERLIB DD   DSNAME='DB2TOPE.PROJECT.INCLUDE',DISP=SHR
//*
//OBJLIB  EXEC EDCLIB,OPARM='MAP',LIBRARY='DB2TOPE.PROJECT.OBJ'
//
```

2. Modification and execution of COMPILE job

   Job COMPILE uses the C/370 compiler procedure EDCCPLG that contains four steps that compiles, pre-links, link-edits and executes a C/370 program. Since at this stage we are interested only in the first three steps that deals with DB2SPLIT program preparation, we had to modify both, the COMPILE job and the EDCCPLG procedure. To represent its real functions, that is compile, pre-link and link, we renamed it to EDCCPL.

   The modifications to procedure EDCCPLG consist in changing the first two lines and commenting or deleting the other nine as indicated in the following JCL:

Procedure EDCCPL. Modification of EDCCPLG

```
//EDCCPL  PROC CREGSIZE='4M',
//...     ...
//        LNGPRFX='EDC.V1R2M0'   last (,) moved or deleted
//*       GREGSIZ='2048K',
//*       GPARM=
//...     ...
//*GO     EXEC PGM=*.LKED.SYSLMOD,
//*       COND=(((4,LT,COMPILE),(4,LT,PLKED),(8,LE,LKED)),
//*       REGION=&GREGSIZ,PARM='&&GPARM'
//*STEPLIB  DD  DSNAME=SYS1.CEE.V1R5M0.SCEERUN,DISP=SHR
//*SYSPRINT DD  SYSOUT=*
//*CEEDUMP  DD  SYSOUT=*
//*SYSUDUMP DD  SYSOUT=*
```

Modifications to job COMPILE consisted mainly in deleting the references to the GO step that executes the program just prepared. The resulting job was:

Job COMPILE, Compiles, Pre-links and Links DB2SPLIT

```
//...     JOB ...
//CPL     EXEC EDCCPL,LPARM='AMODE(31),RMODE(ANY)',
//        CPARM='NOSEQ,NOMAR,LONGNAME,DEF(VMMVS)',
//        INFILE='DB2TOPE.PROJECT.SOURCE(DB2SPLIT)',
//        OUTFILE='DB2TOPE.PROJECT.MOD(DB2SPLIT)',DISP=SHR
//COMPILE.USERLIB DD DSNAME=DB2TOPE.PROJECT.INCLUDE,DISP=SHR
//PLKED.SYSLIB    DD DSNAME=DB2TOPE.GRPIAPI.OBJ,DISP=SHR
//                DD DSNAME=DB2TOPE.PROJECT.OBJ,DISP=SHR
//
```

The executable module DB2SPLIT is link-edited to DB2TOPE.PROJECT.MOD library.

5.5 Executing the DB2SPLIT under MVS

To execute the DB2SPLIT program under MVS you have to prepare a job that considers the following:

- The input BSAM data set with the externalized data.
- A configuration sequence of statements (CONFIG), containing a description of the splitting process used as parameters for DB2SPLIT.
- The output data sets that will receive the splitted data.

For the following example, let us consider the DB2MVS.ORDERS data set created according to the guidelines given in 5.2.3, “Application of DSNTIAUL” on page 79.

5.5.1 Describing the Splitting Settings Via CONFIG

The following parameters were coded in the PO MVS data set DB2TOPE.PROJECT.RUN(CONFIG) with attributes RECFM=FB, LRECL=80:

DB2TOPE.PROJECT.RUN(CONFIGH) Parameters for DB2SPLIT

```
Description=ORDERS
InFile=DD:MYINPUT
RecLen=500
Nodes=(1,2,3,4,5,6,7,8,11,12)
OutoutNodes=(1,2,3,4,5,6,7,8,11,12)
MapFilo=DD:OUTMAP
DistFile=DD:DISTFILE
LogFile=DD:LOG
OutFile=DD:NOD
CDelimiter=|
SDelimiter=
RunType=PARTITION
Msg_Level=NOWARN
Partition=O_ORDERKEY,1,,,NN,INTEGER
Trace=10
```

The above parameters are described in detail in the manual *DB2 Parallel Edition for AIX, Administration Guide and Reference*, chapter "Loading Data in the Parallel Environment", SC09-1982-00. We will focus on some of the values selected for our environment:

Parameter Comments

InFile=DD:MYINPUT is the MVS DD-name for the externalized data to be split.

RecLen=500 The value ranges from 0 to 32000, and for delimited data you must use a value greater than the maximum record length of the input file. According to the values calculated for 5.2.3.4, “Intermediate Storage Space Required for MVS” on page 86, our records with delimiters are ranging from 190 to 249 bytes length so, with 500 we are in the safe side.

Nodes=(1,2,3,4,5,6,7,8,11,12) are the ten nodes where our DB2 Parallel Edition tables are created. These values are used for generating a partitioning map.

OutNodes=(1,2,3,4,5,6,7,8,11,12) are the output files that will be loaded by DB2SPLIT. They have to be a subset of the Nodes= parameter, meaning you may produce splitted data for a subset of the nodes. In our case we are producing all the ten files for all the corresponding ten nodes.

MapFilo=DD:OUTMAP is the MVS DD-name of the output partitioning map data set generated at DB2SPLIT run time.

DistFile=DD:DISTFILE is the MVS DD-name of the output distribution data set that can be used by the DB2 Parallel Edition data redistribution utility.

Note: It is a required file.

LogFile=DD:LOG is the MVS DD-name of the log data set. The log data set traces the hashing values up to the amount indicated in the Trace=number parameter. Then it traces the amounts of records processed every 50,000 input records. Finally, it records some statistics and the amount of records derived to each of the receiving files that represents the corresponding DB2 Parallel Edition nodes.

Outfile=DD:NOD are the default three prefix characters that you can select to give the MVS DD-names of the data sets that will be receiving the splitted data. DB2SPLIT appends a 5-character suffix (ranging from 00000 to 00999), to the end of the prefix to complete an eight character MVS DD-name.

Note: For MVS the prefix is always NOD and the suffix is always five zeroes ending with the corresponding node numbers coded in the former OutoutNodes=(...) parameter.

CDelimiter=| is the Column Delimiter. It cannot be blank and, if not coded, implies that the data is not delimited.

RunType=PARTITION indicates DB2SPLIT to partition the data. ANALYZE indicates DB2SPLIT to produce a customized partitioning map.

Msg_Lvel=NOWARN is the default, DB2SPLIT will not stop at every warning message. WARN makes DB2SPLIT to stop at every warning message.

Partition=O_ORDERKEY,1,,,NN,INTEGER There are six fields separated by commas:

1. The column name that will be used for the partitioning.
2. Valid only for DEL (delimited ASCCI format) data, is the cardinal value (1, 2, ... ,n) indicating the position of the partitioning field in the input records.
3. Valid only for ASC (non-delimited ASCCI format) data, is the offset for the start of the partitioning key.
4. Valid only for ASC (non_delimited ASCCI format) data, is the length of the partitioning key.
5. NN indicates Not Null, data must not be null. N indicates Null data is allowed.
6. INTEGER indicates that data will be converted into a 4-byte integer for hashing into the partition index.

Trace=10 Only the first ten input record entries will be traced to the LOG data set.

5.5.2 Output Data Sets for DB2SPLIT

DB2SPLIT can produce four data set named OUTMAP, DISTFILE, LOG and the NOD00nnn, as described above. To define them to MVS, let's estimate the spaces required for each:

1. OUTMAP

   This data set has 410 records, with ten entries each but the last, which has six entries, for a total of 4096 entries (409 x 10 + 6 = 4096). Each entry represents a node number from those specified in the CONFIG data set, in character format. The node numbers are separated from each other by a blank/space.

   According to this, the maximum file size required for a RECFM=FB data set format will be:

   - LRECL=29 (10 x 2-digits-per-node + 9 blanks/spaces = 29 bytes)
   - BLKSIZE=11890 (410 records x 29 bytes/record = 11890 bytes)
   - SIZE=(TRK,(1,0)), one track will suffice.

2. DISTFILE

   This data set has 4096 records, with one entry each containing a character digit 0 or 1. According to this, the maximum file size required for a RECFM=FB data set format will be:

   - LRECL=1
   - BLKSIZE=4096 (4096 records x 1 bytes/record = 4096 bytes)
   - SIZE=(TRK,(1,0)), one track will suffice.

3. LOG

   The log data set can be directed to a printer device. It has to accommodate about 30 header and trailer informational records, plus two records per traced values (Trace=number parameter), plus one record each 50,000 splitted records, plus one record per node. The records are variable length format with an average record length less than 40 bytes/record. According to this, for a RECFM=VB data set format, with Trace=1000 and 500,000,000 records to split, the maximum file size required will be:

   - LRECL=132
   - BLKSIZE=4096 (132 x 31 + 4 block-length-bytes = 4096 bytes). Each page of 4096 bytes will accommodate more than 80 entries.
   - SIZE=(CYL,(1,0)), one cylinder 3390 DASD will suffice.

     Note: One cylinder 3390 accommodates 15 trk/cyl x 12 pages/tr = 180 pages, that is 180 pages x 80 records/page = 14,400 records. For our example we require less than that:

     – 30 header and trailer records = 30 records

     – 500,000,000 inputs / 50,000 inputs/record = 10,000 records

     – Trace=1000 traces x 2 record/trace = 2,000 records

     – All above adds up to a total of 12,030 records

4. NOD00nnn

   The nodes data sets have to accommodate the splitted data. Every NOD00nnn data set has the following format:

   - One record containing the character number that represents the corresponding node.
   - 34 records containing 124 entries each but the last, that contains four entries, for a total of 4096 entries (33 x 124 + 4 = 4096), each containing the character digits representing the node numbers specified in the CONFIG data set, followed by one blank/space.

     The size of each record can be as small as 147 bytes (124 x 1 + 124 spaces = 147 bytes) if the node numbers involved are one character number each, or as big as 372 bytes (124 x 2 + 124 spaces = 372 bytes).
   - Then comes six small records documenting the key for hashing.
   - Finally, there are the splitted records derived to this data set. They have the same length that they have at the input data set. According to our work, the distribution of the splitted data was even among the node data sets, with less than +- 0.2% deviation.

Since the NOD00nnn data sets are the more voluminous, care has to be taken to make them small still capable of storing the corresponding data. Due to the fact that the prefix described above has a variable size that in general differs from the splitted data, we recommend using blocked variable data set organization, RECFM=VB.

The maximum record length must be the maximum of either 376 bytes (372 + 4 record length bytes for the variable format), or the length of the splitted records plus the four record length bytes. According to the values calculated for 5.2.3.4, “Intermediate Storage Space Required for MVS” on page 86, our records with delimiters are ranging from 190 to 249 bytes length, so with LRECL=376 we are in the safe side.

Since we are dealing with variable length records, you can code the maximum block size possible for a 3390 DASD device, that is BLKSIZE=27998.

Let us now calculate the space in 3390 DASD cylinders required for each of the ten NOD00nnn data sets. The following are the assumptions:

- 12,000,000 records for the table ORDERS in DB2MVS.ORDERS data set
- LREC=173 bytes per record, fixed length
- BLKSIZE=27998 bytes for 3390 DASD device
- 2 blocks/track x 10 tracks/cylinder = 30 blocks/cylinder
- To be in the safe side, let us consider the deviation from an even distribution to be +1% (our measures show +-0.2%).

With the above assumptions, the number of 3390 DASD cylinders required for each NOD00nnn data set will be the following:

- Effective LRECL = 173 + 4 record length bytes = 177 bytes/record
- Effective BLKSIZE = 27998 - 4 block length bytes = 27994 bytes/block
- Number of records per block = 27994 / 177 = 158, rounded down
- Number of records per cylinder = 158 x 30 = 4740 records/cylinder
- Total number of cylinders = 12,000,000 / 4740 + 1 (for the overhead records) = 2533, rounded up
- Cylinders per node = (2531 / 10) * 1.01 (to account for +1%) = 256, rounded up

5.5.3 Job to Execute DB2SPLIT

The following job was coded for the splitting of the data extracted from the table DB2MVS.ORDERS. With the adequate sizes and names modifications was also used for the other tables.

Job to Run DB2SPLIT against Externalized DB2MVS.ORDERS Data

```
//...      JOB ...
//DB2SPLIT PROC
//*-------------------------------------------------------------------
//DELETEDS EXEC PGM=IEFBR14
//OUTMAP   DD DSN=&USERNAME..&PNAME..OUTMAP,DISP=(MOD,DELETE,DELETE),
//         SPACE=(TRK,(1,0)),DCB=(RECFM=VB,LRECL=10,BLKSIZE=120),
//         UNIT=SYSDA
//DISTFILE DD DSN=&USERNAME..&PNAME..DISTFILE,DISP=(MOD,DELETE,DELETE),
//         SPACE=(TRK,(1,0)),DCB=(RECFM=VB,LRECL=10,BLKSIZE=120),
//         UNIT=SYSDA
//LOG      DD DSN=&USERNAME..&PNAME..LOG,DISP=(MOD,DELETE,DELETE),
//         SPACE=(TRK,(1,0)),DCB=(RECFM=VB,LRECL=10,BLKSIZE=120),
//         UNIT=SYSDA
//NOD0001  DD DSN=&USERNAME..&PNAME..NOD00001,DISP=(MOD,DELETE,DELETE),
//         SPACE=(TRK,(1,0)),DCB=(RECFM=VB,LRECL=10,BLKSIZE=120),
//         UNIT=SYSDA
//...      ...     ...      Same attributes for NOD00002 to NOD000011
//NOD0012  DD DSN=&USERNAME..&PNAME..NOD00012,DISP=(MOD,DELETE,DELETE),
//         SPACE=(TRK,(1,0)),DCB=(RECFM=VB,LRECL=10,BLKSIZE=120),
//         UNIT=SYSDA
//*-------------------------------------------------------------------
//SPLIT    EXEC PGM=DB2SPLIT
//STEPLIB  DD DSN=&USERNAME..&PNAME..MOD,DISP=SHR
//         DD DSN=&LIBPRFX..SCEERUN,DISP=SHR
//CONFIG   DD DSN=&USERNAME..&PNAME..RUN(CONFIG),DISP=SHR
//MYINPUT  DD DSN=DB2MVS.ORDERS,DISP=SHR
//OUTMAP   DD DSN=&USERNAME..&PNAME..OUTMAP,
//         DISP=(NEW,CATLG,DELETE),UNIT=SYSDA,VOL=SER=DBPE14,
//         DCB=(RECFM=FB,LRECL=29,BLKSIZSE=11890),SPACE=(TRK,(1,1),RLSE)
//DISTFILE DD DSN=&USERNAME..&PNAME..DISTFILE,
//         DISP=(NEW,CATLG,DELETE),UNIT=SYSDA,VOL=SER=DBPE14,
//         DCB=(RECFM=FB,LRECL=1,BLKSIZSE=4096),SPACE=(TRK,(1,1),RLSE)
//LOG      DD DSN=&USERNAME..&PNAME..LOG,
//         DISP=(NEW,CATLG,DELETE),UNIT=SYSDA,VOL=SER=DBPE14,
//         DCB=(RECFM=VB,LRECL=132,BLKSIZSE=4096),SPACE=(CYL,(1,1),RLSE)
//NOD00001 DD DSN=&USERNAME..&PNAME..NOD00001
//         DISP=(NEW,CATLG,DELETE),UNIT=SYSDA,VOL=SER=DBPE15,
//         DCB=(RECFM=VB,LRECL=376,BLKSIZSE=27998),
//         SPACE=(CYL,(256,1),RLSE)
//...      ...     ...    Similar attributes for NOD00002 to NOD00011
//NOD00012 DD DSN=&USERNAME..&PNAME..NOD00012
//         DISP=(NEW,CATLG,DELETE),UNIT=SYSDA,VOL=SER=DBPE16,
//         DCB=(RECFM=VB,LRECL=376,BLKSIZSE=27998),
//         SPACE=(CYL,(256,1),RLSE)
//         PEND
//*-------------------------------------------------------------------
//STEP1    EXEC DB2SPLIT,USERNAME=DB2TOPE,PNAME=PROJECT,
//         LIBPRFX=SYS1.CEE.V1R5M0
```

The job has the form of a two steps procedure called DB2SPLIT which is invoked from the step called STEP1, passing the parameters USERNAME, PNAME and LIBPRFX, at the bottom.

The first step of DB2SPLIT, has the purpose of deleting the OUTMAP, DISTFILE, LOG and NOD00nnn data sets, either they already exist or not. The second step will create and load the data sets.

Notes:

1. To avoid data set contention, it is good practice to code different volume serial numbers for the NOD00nnn data sets, provided you have the required space available.
2. Since the parameter RLSE was coded in the NOD00nnn Data Definitions (DDs), the final size was 254 cylinders for some of them and 253 for the rest, proving that our calculations were correct.

The former job can be simplified if some of the OUTMAP, DISTFILE and LOG file are not required in DASD and are derived to printed output. The job would be:

Simplified Job to Run DB2SPLIT

```
//...      JOB ...
//DB2SPLIT PROC
//*-------------------------------------------------------------------
//DELETEDS EXEC PGM=IEFBR14
//NOD0001  DD DSN=&USERNAME..&PNAME..NOD00001,DISP=(MOD,DELETE,DELETE),
//         SPACE=(TRK,(1,0)),DCB=(RECFM=VB,LRECL=10,BLKSIZE=120),
//         UNIT=SYSDA
//...      ...     ...      Same attributes for NOD00002 to NOD000011
//NOD0012  DD DSN=&USERNAME..&PNAME..NOD00012,DISP=(MOD,DELETE,DELETE),
//         SPACE=(TRK,(1,0)),DCB=(RECFM=VB,LRECL=10,BLKSIZE=120),
//         UNIT=SYSDA
//*-------------------------------------------------------------------
//SPLIT    EXEC PGM=DB2SPLIT
//STEPLIB  DD DSN=&USERNAME..&PNAME..MOD,DISP=SHR
//         DD DSN=&LIBPRFX..SCEERUN,DISP=SHR
//CONFIG   DD DSN=&USERNAME..&PNAME..RUN(CONFIG),DISP=SHR
//MYINPUT  DD DSN=DB2MVS.ORDERS,DISP=SHR
//OUTMAP   DD SYSOUT=*
//DISTFILE DD SYSOUT=*
//LOG      DD SYSOUT=*
//NOD00001 DD DSN=&USERNAME..&PNAME..NOD00001
//         DISP=(NEW,CATLG,DELETE),UNIT=SYSDA,VOL=SER=DBPE15,
//         DCB=(RECFM=VB,LRECL=376,BLKSIZSE=27998),
//         SPACE=(CYL,(256,1),RLSE)
//...      ...     ...    Similar attributes for NOD00002 to NOD00011
//NOD00012 DD DSN=&USERNAME..&PNAME..NOD00012
//         DISP=(NEW,CATLG,DELETE),UNIT=SYSDA,VOL=SER=DBPE16,
//         DCB=(RECFM=VB,LRECL=376,BLKSIZSE=27998),
//         SPACE=(CYL,(256,1),RLSE)
//         PEND
//*-------------------------------------------------------------------
//STEP1    EXEC DB2SPLIT,USERNAME=DB2TOPE,PNAME=PROJECT,
//         LIBPRFX=SYS1.CEE.V1R5M0
```

5.5.4 Intermediate Storage Space for Spliting in MVS

Table 13 shows the cylinders required for intermediate storage of splitted data in MVS for the DB2MVS tables. The calculations were based on the example 5.5.2, “Output Data Sets for DB2SPLIT” on page 95.

Table 13. Storage Required for Splitted Data Sets in MVS. Space is cylinders of 3390 DASD type. The calculations were based on the example 5.5.2, “Output Data Sets for DB2SPLIT” on page 95.

Note: *Data from tables DB2MVS.NATION and DB2MVS.REGION were not split because they are very small. Nevertheless, were included in this table for completeness.*

DB2MVS.tablenames data sets and the corresponding number of records	**LRECL = fixed length + 4 bytes**	**No. of records in 27998 - 4 = 27994 bytes/block**	**No. of records per cylinder**	**Total number of cylinders required per data set**	**No. of cylinders required for each NOD00nnn data set**
DB2MVS.NATION 25	201+4 = 205	27994/205 = 136	30*136 = 4080	25/4080 + 1 = 1	1*0.101 = 1
DB2MVS.REGION 5	190+4 = 194	27994/194 = 144	30*144 = 4320	5/4320 + 1 = 1	1*0.101 = 1
DB2MVS.PART 1,600,000	191+4 = 195	27994/195 = 143	30*143 = 4290	1,600,000/4290 + 1 = 374	374*1.01 = 38
DB2MVS.SUPPLIER 80,000	222+4 = 226	27994/226 = 123	30*123 = 3690	80,000/3690 +1 = 23	23*0.101 = 3
DB2MVS.PARTSUPP 6,400,000	248+4 = 252	27994/252 = 111	30*111 = 3330	6,400,000/3330 + 1 = 1923	1923*0.101 = 195
DB2MVS.CUSTOMER 1,200,000	249+4 = 253	27994/253 = 110	30*110 = 3300	1,200,000/3300 + 1 = 365	365*0.101 = 37
DB2MVS.ORDERS 12,000,000	173+4 = 177	27994/177 = 158	30*158 = 4740	12,000,000/4740 + 1 = 2533	2533*0.101 = 256
DB2MVS.LINEITEM 49,989,007	223+4 = 227	27994/227 = 123	30*123 = 3690	49,989,007/3690 + 1 = 13,007	13,007*0.101 = 1314

Chapter 6. Data Migration and Loading

This chapter examines the different methods of migrating data from a mainframe to RISC/6000 SP with DB2 Parallel Edition.

The first section describes the functions of the different software products that can be used.

The next section describes the various business requirements, possible technical solutions and their advantages and disadvantages. The business requirements are:

- Initial data load
- Complete refresh
- Addition of new rows
- Apply changes

The chapter concludes with a summary of the recommendations.

6.1 Software Products

Assuming there is an ESCON channel connection between MVS and the RISC/6000 SP, there are a number of software products to move the data from MVS and load it on to DB2 Parallel Edition, including the following described below:

- TCP/IP
- CLIO/S
- BatchPipes/MVS
- DDCS
- Splitter
- Load
- Autoloader
- DataPropagator

6.1.1 TCP/IP

TCP/IP provides several functions that can be used in the process of moving data from the mainframe to DB2 Parallel Edition.

The TCP/IP File Transfer Protocol (FTP) enables files to be transferred from the mainframe to the RISC/6000 SP across any TCP/IP network including ESCON. It can be used to transfer sequential files, for example the output of an unload can be transferred as an input to db2split. FTP can transfer the whole file in binary format or it can convert the whole file from EBCDIC to ASCII during the data transfer. The file transfer can be initiated from either end, either as a PUT from MVS or a GET to AIX.

Besides transferring sequential files FTP can read from DB2 for MVS tables. Any SELECT statement can be issued and the results transferred to the target. The data is transformed into external format and converted to ASCII. Unfortunately the

file formats available do not include one that is directly useable by load. A program could be written that uses this transfer mechanism and creates a load file.

TCP/IP MVS also includes a facility to issue AIX distributed shell commands from TSO or batch. These could be useful when automating load processes, for example, a command could be issued to make a pipe on AIX before an MVS program starts writing to it.

Further information on TCP/IP for MVS can be found in *IBM TCP/IP for MVS User's Guide,* SC31-7136-00.

6.1.2 CLIO/S

IBM Client Input Output/Sockets (CLIO/S) is a set of commands and application programming interfaces (API) that can be used for high-speed communications and accessing tape drives on a network of AIX and MVS mainframes. Communication is over the following channel-to-channel (CTC) connections:

- Block Multiplexer Channels
- ESCON Channels
- Serial Optical Channel Converter (SOCC) for RS/6000

CLIO/S is optimized for the reliable, point-to-point CTC connection and is therefore five to ten times faster than TCP/IP or SNA. This makes it a very attractive option for transferring large volumes of data for initial loads or refreshes of DB2 Parallel Edition tables, as well as backing up tables to tape on the mainframe.

CLIO/S File Transfer (CLFTP) provides a similar interface as TCP/IP FTP using the CLIO/S protocol. It is a good interface for on-line use or when transferring multiple files.

Cross System Pipe Link (CLPLINK) provides a facility for file transfer to and from pipes. The pipes can be AIX pipes, or MVS BatchPipes. This enables a sequential output from a program, for example an unload on MVS, to be piped directly to a process on AIX, such as db2split. It can also be used as a file transfer mechanism, as an alternative to CLFTP. CLPLINK's interface is well suited to automated procedures, such as JCL on MVS or shell scripts on AIX.

CLIO/S Tape Server enables MVS tapes to be read directly by AIX applications. This facility can be used to read tapes from MVS directly into load processes. Its most likely used for writing database backups directly to MVS tape, thereby removing the need for tapes directly connected to the RISC/6000 SP.

Further information on CLIO/S can be found in *IBM Client Input Output/Sockets General Information Version 2 Release 1,* GC23-3879.

6.1.3 BatchPipes/MVS

BatchPipes/MVS offers a way to connect jobs so that sequential data from one job can move through processor storage to another job without going to an intermediate file on disk or tape. The target job can read data as soon as the source writes it, thus allowing the two jobs to run in parallel. In this way BatchPipes reduces overall elapse time and removes the requirement for intermediate storage.

BatchPipes can be used to feed the output of an unload directly into db2split on MVS, or into CLIO/S CLPLINK for transfer to the RISC/6000 SP.

Further information on BatchPipes/MVS can be found in *IBM BatchPipes/MVS Introduction,* GC28-1214-02.

6.1.4 DDCS/6000

DDCS/6000 enables SQL statements to be issued on the RISC/6000 SP against DB2 tables on an MVS mainframe, or any other platform that is a DRDA application server.

DDCS/6000 is a prerequisite to using DataPropagator Apply.

DDCS/6000 can also be used to extract data from the source and transfer it to the RISC/6000 SP. DDCS will automatically convert all fields correctly from source format (for example EBCDIC or packed decimal on MVS) to target format (ASCII or decimal on the RISC/6000 SP). DB2 for MVS includes the application server function, so no extra software is required on MVS. The transfer is done directly from DB2 to the RISC/6000 SP, so no temporary files are needed on MVS, and the data transfer is done concurrently with the extract.

Further information on DDCS/6000 can be found in *IBM DDCS V2 Spec Sheet,* GC09-2211-02.

6.1.5 db2split

The db2split program is part of DB2 Parallel Edition. It takes a file to be loaded and splits it into separate files, one for each node of a DB2 Parallel Edition table. The load utility is then used to load the data files to each node. The loads are independent of each other and can run in parallel

The db2split program can run on the RISC/6000 SP or on the mainframe.

Further information on db2split can be found in Chapter 5, “Data Extractions” on page 69, and in *IBM DB2 Parallel Edition for AIX Administration Guide and Reference,* SC09-1982-00.

6.1.6 Load

Load is a standard part of DB2 Parallel Edition It takes a sequential file and loads the data into the DB2 Parallel Edition table.

Further information on loader can be found in Chapter 5, “Data Extractions” on page 69, and in *IBM DB2 Parallel Edition for AIX Administration Guide and Reference,* SC09-1982-00.

6.1.7 Autoloader

The autoloader process is a new function in DB2 Parallel Edition. It combines the function of db2split and loader thereby simplifying the process. However the loading now takes place in parallel with db2split and the elapse time will be dependent on the speed of db2split.

Further information on autoloader can be found in Chapter 5, “Data Extractions” on page 69, and in *IBM DB2 Parallel Edition for AIX Administration Guide and Reference,* SC09-1982-00.

6.1.8 DataPropagator Relational

DataPropagator Relational is a set of easy-to-use, automated copy tools. It provides automatic copying from relational sources to relational targets:

- Among the DB2 family of databases
- Across MVS, VM, VSE, OS/400, AIX, HP-UX, Solaris Operating Environment, and OS/2 platforms

DataPropagator Relational can be used to create, synchronize, automate and manage copy operations from a single control point for data across your enterprise for various platforms. You can tailor or enhance data as it is copied and deliver detailed, subset, summarized or derived data when it is needed. You can also distribute data to many targets or consolidate data from many sources.

DataPropagator Relational can extract data from DB2 for MVS to create different types of copy tables:

Point-in-time

A table containing a row for every primary key value of interest with only the current data of the source table, that is, a complete condensed copy of the source table.

Base aggregate

A target table that contains rows created as a result an SQL column function performed against the source table. A row is appended in the target table for each changed row in the source table.

Change aggregate

A history table in which a new row is appended for each changed row in the source table. The appended row contains the results of an SQL column function calculation against the changes recorded for the source table.

Condensed, noncomplete Consistent Change Data (CCD)

A table that contains only the last changed value of committed rows. When this type of copy is defined locally to the source, it is useful for providing a stable source for effecting synchronization of fan out copies.

Condensed, complete CCD

A table containing a row for each primary key value of interest that is updated by the Apply component with only committed changes. This type of copy is useful as a source for other copy operations which allows for remote copies to be both initialized and maintained without having to access the original source each time it is updated.

Noncondensed, noncomplete CCD

The table contains a history of committed changes to the data, it is appended by each insert, update, and delete; useful for auditing purposes.

Noncondensed, complete CCD

A table containing a row for every primary key value of interest. It starts out as a copy of the full-size source table and is appended by each committed insert, update and delete.

The copies are maintained by DataPropagator Relational using the frequency parameter specified during the subscription process.

The steps to implement data propagation from DB2 for MVS to DB2 Parallel Edition are presented in Chapter 7, “Implementing Data Propagation from DB2 for MVS to DB2 Parallel Edition” on page 171. For complete information on the product refer to the *DataPropagator Relational Guide Release 2,* SC26-3399-03.

6.2 Business Scenarios and Their Technical Solutions

This section describes the various business scenarios that require data to be transferred from the mainframe to DB2 Parallel Edition. For each scenario, possible technical solutions are described with the benefits and disadvantages of each.

The business scenarios are as follows:

- Initial data load.
- Complete table refresh.
- Addition of new rows (insert only).
- Apply changes (updates, inserts and deletes).

For all these scenarios, the data has to go through the following processes:

1. Extract the data from the source.
2. Translate the data from source formats to target formats.
3. Transfer the data from source platform to target platform.
4. Split the data to be loaded onto each node of the table.
5. Load the data into the table.

The order of the processes will vary between different scenarios; depending on:

- The software available
- Constraints on the elapse time of the extract (for example, is there a batch window that it has to complete in)
- Constraints on CPU and disk space on the source
- Constraints on CPU and disk space on the target
- Constraints on the elapse time for the load (for example, if this is a refresh, then the table is unavailable for use during the load)
- Total elapse time from start of extract to end of load (for example if the extract can not start before the end of overnight batch processing and the load has to be ready for processing the next morning)
- The level of independence required to simplify restart processing

Two extreme examples will demonstrate the differences that will be expanded in the various scenarios.

1. Step-by-step processing concentrating resource on the source:

   a. Run an extract program (for example DSNTIAUL) to extract from DB2 for MVS and create a sequential delimited external format file that is useable by load.

   b. Run the db2split program on MVS and create one sequential file for each node (these files are still on MVS).

   c. Transfer the files, using TCP/IP FTP, to sequential files on each node of the RISC/6000 SP. This will convert EBCDIC to ASCII.

   d. Run the loads in parallel.

   This procedure:

   - Minimizes the time for extract.
   - Minimizes the time for load.
   - Concentrates the CPU and disk load on MVS.
   - But because of its sequential nature it has a fairly lengthy total elapse time.

2. Minimize total elapse time:

   a. Run a program on the RISC/6000 SP using DDCS to DB2 on MVS, that creates a sequential output. This one step includes the extract, translate and transfer processes.

   b. Pipe the output of DDCS to autoloader. This one step includes the splitting and loading processes.

   This procedure will:

   - Minimize the total elapse time as all the processes are running in parallel
   - Marginally increase the export time
   - Increase the load time because it will be waiting on a sequential export
   - Minimize the temporary disk space (as there are no intermediate files)
   - Not be restartable

The variations are summarized in Figure 31 on page 107.

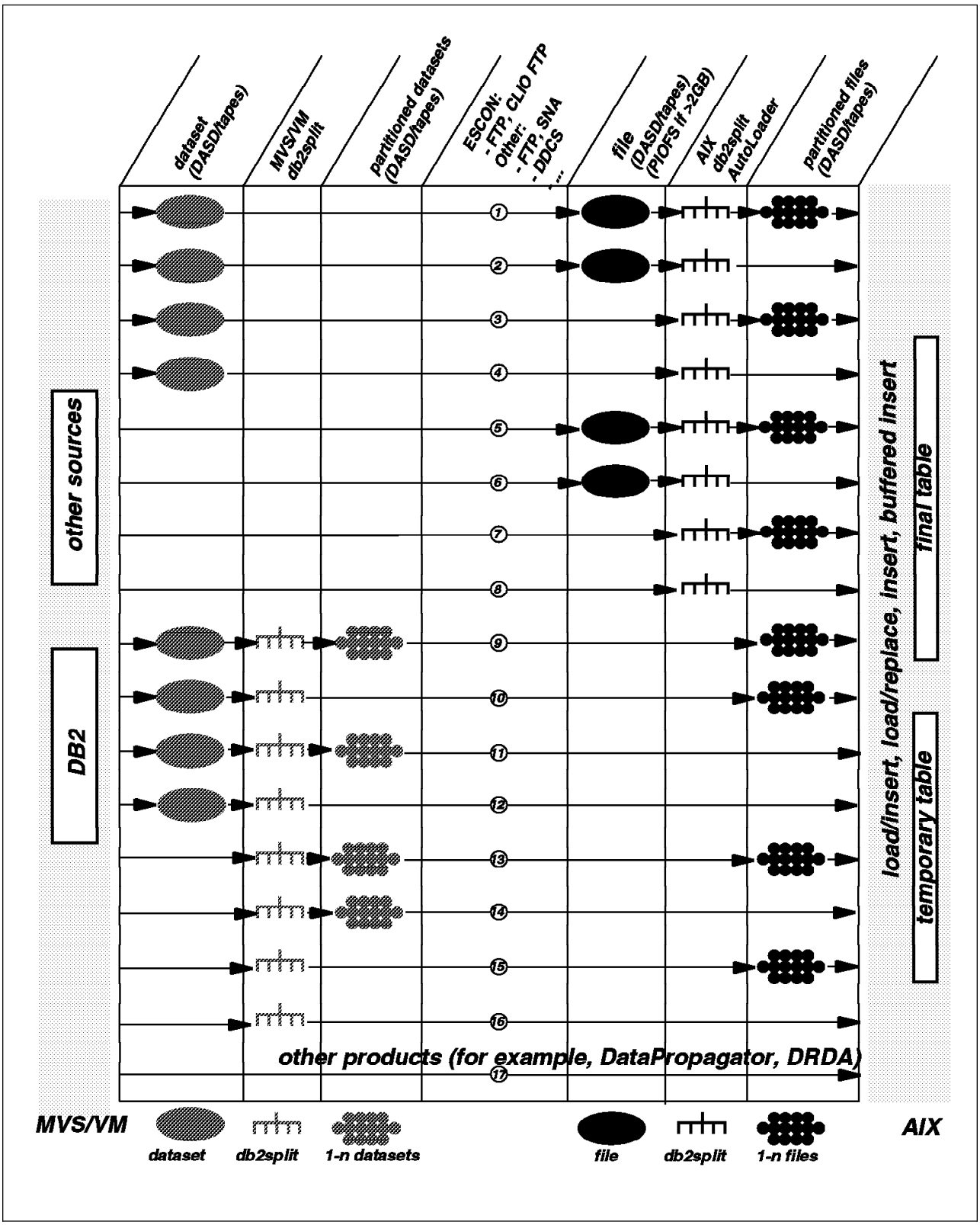


Figure 31. Summary of Transfer Variations

The two options described above are numbers 9 and 7 in Figure 31.

All of the variations are described in the following section. For each variation there is:

- A summary
- A diagram
- A description
- Details of how to set up the process
- A list of benefits
- A list of limitations

The variations are split into 5 groups based on where the unload file is stored and where the splitting is done:

- Unload File on MVS, db2split on AIX (Variations 1-4)
- No Unload File on MVS, db2split on AIX (Variations 5-8)
- Unload File on MVS, db2split on MVS (Variations 9-12)
- No Unload File on MVS, db2split on MVS (Variations 13-16)
- Other Solutions (Variation 17)

6.2.1 Unload File on MVS, db2split on AIX

In this set of variations, the first step unloads the data from the source into a sequential file on MVS. This file is then transferred to AIX either into a file or directly into the split process.

This set of variations:

- Makes the unload from DB2 (or other sources) independent of the RISC/6000 SP and the connection to it
- Concentrates the processing load on the RISC/6000 SP

6.2.1.1 Variation 1: All Files

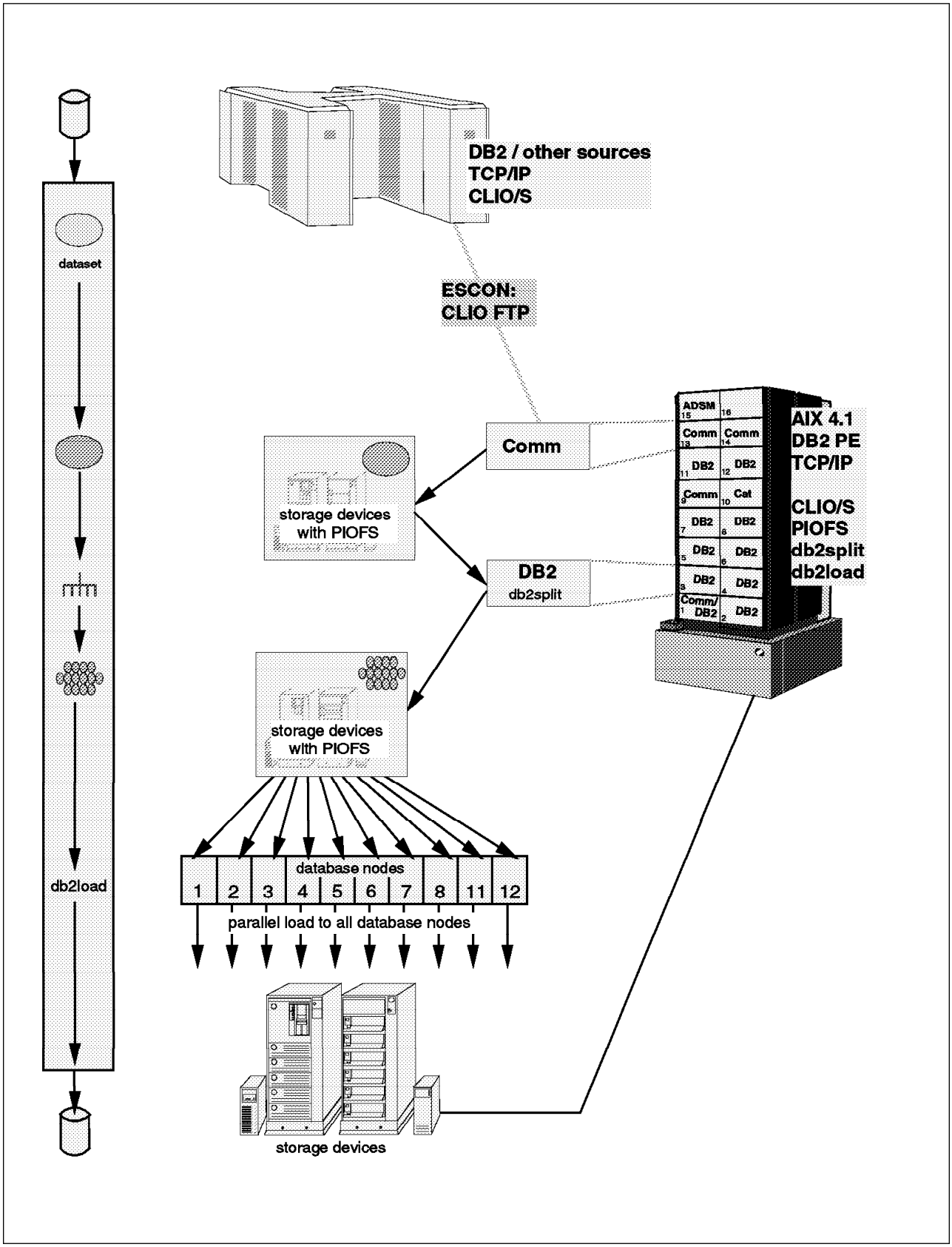


Figure 32. Variation 1 Overview

Figure 32 is an overview of variation 1.

Summary: This is a base for comparison with other variations. You should find that one of the other variations better meets your requirements either by reducing temporary disk space, total elapsed time or AIX CPU load.

This is the best option if you want to use tape transfer and you want to minimize the unavailability of the DB2 Parallel Edition table.

Description

1. An unload process, such as DSNTIAUL, creates a sequential file on MVS in a format suitable for loading into DB2 Parallel Edition.
2. The file is transferred to AIX using the fastest transfer method available (CLIO/S is the preferred method). This transfer can be initiated as part of the MVS job stream or the AIX script depending on how quickly you want to release the space on MVS.
3. db2split is run and creates an output file per node.
4. The load files are loaded on all nodes in parallel.

Process: The file transfer can be initiated from the following:

- MVS, as part of the unload job
- AIX, as part of the split shell script

Once the file transfer is complete, the file on MVS can be deleted. The decision as to which option to choose will depend on factors such as:

- Scheduling constraints on MVS
- Scheduling constraints on RISC/6000 SP
- Availability of disk space on each end

Examples of JCL and shell scripts for both options are included below. Obviously the CLFTP step should only be included at one end.

Figure 33 shows the JCL for running the unload and the transfer using CLIO/S CLFTP. The first job step unloads the data using DSNTIAUL. The second step uses CLFTP to put the data on to the RISC/6000 SP. CLPLINK could have been used instead, see Figure 38 on page 115.

```
//UNLOAD  EXEC TIAUL                                                    01170001
//SYSIN    DD  DSN=DB2TOPE.UNLOAD(CUST)                                 01171001
//SYSREC00 DD  DSN=DB2MVS.CUSTOMER,DISP=(,CATLG),                       01230001
//             DCB=(RECFM=FB,LRECL=249,BLKSIZE=27888),                  01240001
//             UNIT=SYSDA,SPACE=(CYL,(358,0)),                          01250001
//             VOL=SER=DBPE01                                           01260001
//FCFTP    EXEC CLFTPB,NODE='SP2ES09'                                   01290001
//SYSIN    DD   *                                                       01300001
echo                                                                    01330001
put 'DB2MVS.CUSTOMER' /db2mvs/customer                                  01340001
quit                                                                    01350001
//                                                                      01360001
```

Figure 33. JCL for Variation 1

Note:

1. The procedure CLFTPB is described in Figure 15 on page 47.

Figure 34 shows the AIX shell script for the file transfer and split. The first command gets the data from MVS and puts it into a file on AIX. The second command runs db2split. For more details on db2split, see Chapter 5, “Data Extractions” on page 69.

```
clftp db2mvs << EOF
get DB2MVS.CUSTOMER db2mvs.customer.aix
quit
EOF
db2split -c /u/db2peusr/cfgfiles/customer.cfg
EOF
```

Figure 34. AIX Script for Variation 1

Note:

1. The EOF's simulate pressing Enter on a terminal; without them, the shell script will go into a wait.
2. The db2split configuration file customer.cfg will point to db2mvs.customer.aix; it is important to ensure that it points to the correct directory.

It would also be possible to move the data from the mainframe to the RISC/6000 SP by writing a tape on MVS and reading it under AIX.

Figure 35 is an example of a script to load the data on to one node. A similar script is needed on each node that will be loaded.

```
db2 connect to tpcd
db2 "load from db2mvs.customer.aix of del modified by coldel| replace into customer"
db2 connect reset
```

Figure 35. Shell Script clload.customer01

Figure 36 is an example of a shell script that starts load script on each of the nodes.

```
for i in 01 02 03 04 05 06 07 08 11 12
do
rsh sp2n$i " . ./.profile  ; ./clload.customer$i "
done
```

Figure 36. Shell Script to Start Load on Remote Nodes

Note:

1. In this example the table is being loaded on nodes 1-8 and 11 and12.
2. The ./.profile command ensures that the command is run from the correct directory.

Benefits

- Fast unload from MVS.
- MVS unload independent of AIX platform.
- MVS dataset can be released as soon as the file transfer is complete.
- File transfer independent of database.
- If file transfer is by tape, then no hardware connection is required.
- The transfer can be initiated either from MVS or AIX. If you initiate it from MVS then as soon as the transfer is complete you can delete the file on MVS. If you initiate it from AIX you will not use space on AIX until this process starts.
- The transfer can be automated as part of a batch job on MVS or a script on AIX.
- Loads in parallel minimize outages on the DB2 Parallel Edition table.

Limitations

- The unload must either create an EBCDIC file or an all ASCII file.
- The steps are sequential so the total time will be long.
- There are temporary files at each step and space must be found for them.
- If the input file for splitter is more than 2 GB, it must be set up using PIOFS.

6.2.1.2 Variation 2: No Load Files

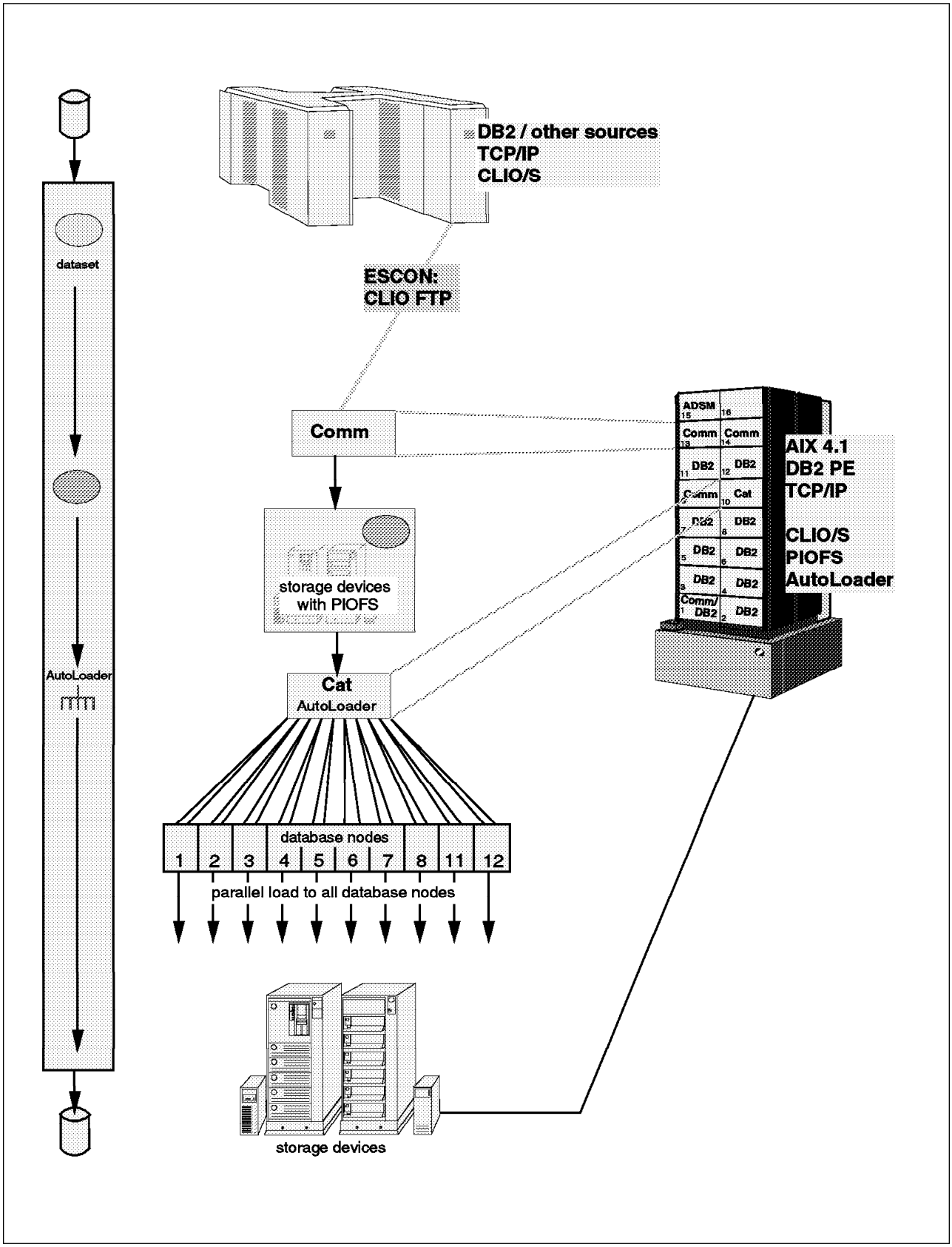


Figure 37. Variation 2 Overview

Summary: This is a good solution for initial load or table refreshes where the refresh window is not critical.

This is the best option if you want to use tape transfer and you want to minimize the time from receipt of the tape and the table being loaded.

Description: This is the same a 6.2.1.1, “Variation 1: All Files” on page 110 except that autoloader is used instead of db2split and loader.

Process: The split and load steps are replaced by an autoloader step. For more details on autoloader please see Chapter 5, “Data Extractions” on page 69. Figure 38 shows the AIX commands for the file transfer and autoloader.

```
clplink -h db2mvs -x db2mvs.nation \
        -w /u/db2peusr/tools/autoloader/frommvs \
        -U endy -P db2pevi
autoloader -d -s nation.spec tpcd
```

Figure 38. AIX Script for Variation 2

Note:

1. The clplink command defines the link that reads data from the file db2mvs.nation, on the remote host db2mvs, and writes it into the temporary file :.../frommvs.
2. The user ID (-U) and password (-P) are included here so that there is no need for operator intervention. If you use this technique you must ensure the security of the shell script.
3. The db2split command points to the config that points to the temporary file.

The autoloader specification file is shown below.

```
/u/db2peusr/tools/autoloader
frommvs nation /u/db2peusr/cfgfiles/nation.cfg
```

Figure 39. Autoloader Specification File for Loading NATION

The configuration file is shown below.

```
Description=tpcd
InFile=/u/db2peusr/tools/autoloader/frommvs
RecLen=32000
Nodes=(11,12)
OutputNodes=(11,12)
DistFile=nation.distfile
LogFile=nation.log,w
OutFile=nation
CDelimiter=|
SDelimiter= '
RunType= PARTITION
Partition=N_NATIONKEY,1,,,NN,INTEGER
Trace=100
```

Figure 40. Splitter Configuration File for Loading NATION

Benefits:

- Fast unload from MVS.
- MVS unload independent of AIX platform.
- File transfer independent of database.
- The transfer can be initiated either from MVS or AIX depending on where the disk space is more critical.
- File transfer can be by tape.
- The transfer can be automated as part of a batch job on MVS or a script on AIX.
- As autoloader runs splitting and loading in parallel, the total elapse time will be less.
- There are no temporary files on the individual load nodes, thereby reducing the disk space required.

Limitations

- The outage on the RISC/6000 SP table will be longer, as db2split takes longer to run than the load on an individual node.
- The unload must either create an EBCDIC file or an all ASCII file.
- If the input file for splitter is more than 2 GB it must be set up using PIOFS.

6.2.1.3 Variation 3: No Input Files to db2split

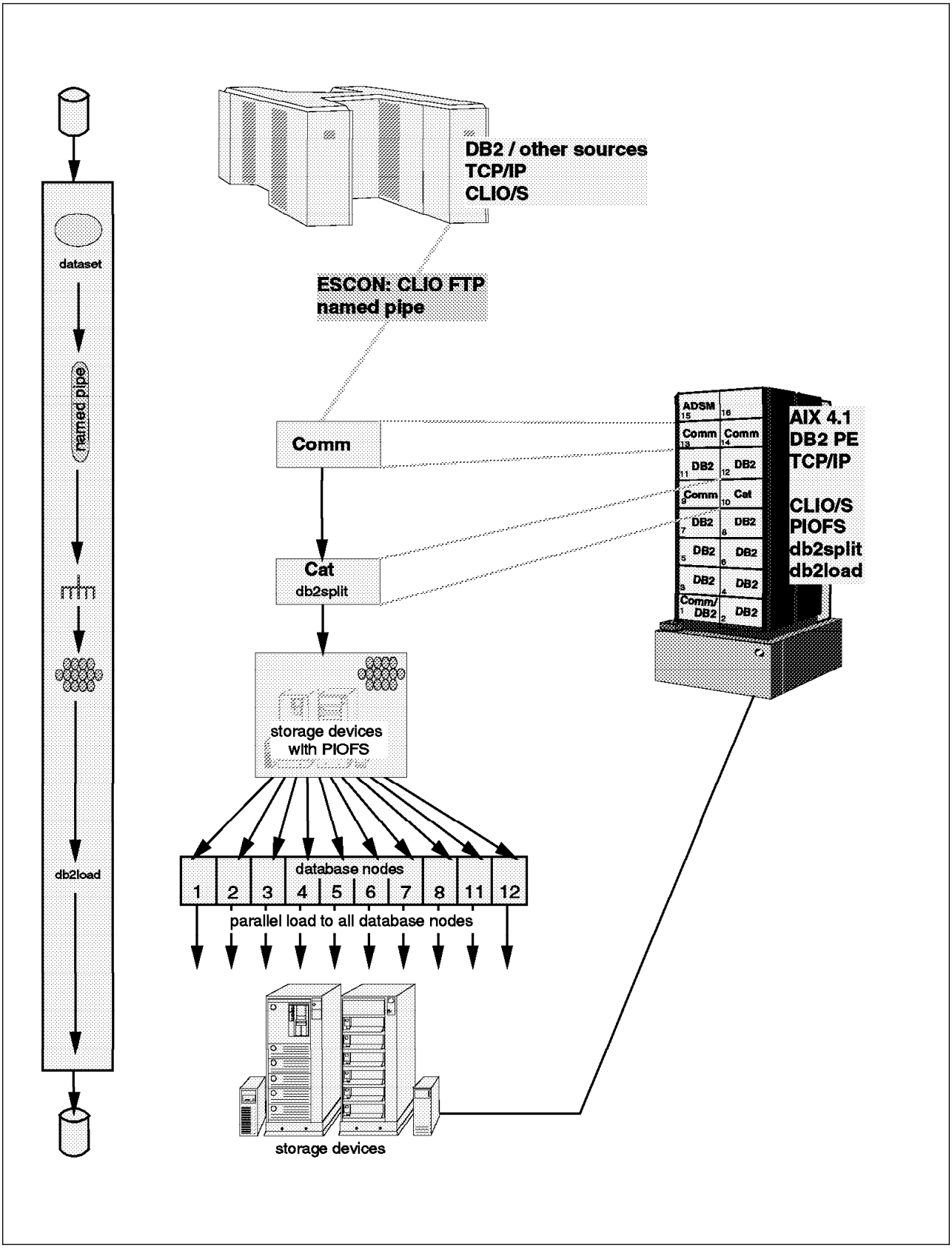


Figure 41. Variation 3 Overview

Summary: This is a good option for regular refreshes of a table on the RISC/6000 SP as it minimizes unload time on MVS and minimizes the outage on the DB2 Parallel Edition table, while reducing overall elapse and AIX disk space requirements in comparison to 6.2.1.1, “Variation 1: All Files” on page 110.

Description: This is the same as 6.2.1.1, “Variation 1: All Files” on page 110 except that the input to db2split is read directly from MVS.

Process: A clplink command transfers data from the unload data set on MVS into an AIX pipe. The pipe is read by db2split. Figure 42 shows the relevant commands.

```
clpmk -r frommvs
clplink -h db2mvs -x db2mvs.nation -w frommvs -U endy -P db2pevi
db2split -c /u/db2peusr/cfgfiles/nation.cfg
```

Figure 42. Shell Script for Piping Data from MVS to db2split

Note:

1. The clpmk command creates a CLIO/S pipe that can be read by an application (in this case db2split).
2. The clplink command defines the link that reads data from the file db2mvs.customer, on the remote host db2mvs, and writes it into the pipe.
3. The user ID and password are included here so that there is no need for operator intervention. If you use this technique you must ensure the security of the shell script.
4. The db2split command points to the config that points to the pipe.

Figure 43 shows the nation.cfg file.

```
Description=tpcd
InFile=/u/db2peusr/tools/autoloader/frommvs
RecLen=32000
Nodes=(11,12)
OutputNodes=(11,12)
DistFile=nation.distfile
LogFile=nation.log,w
OutFile=nation
CDelimiter=|
SDelimiter= '
RunType= PARTITION
Partition=N_NATIONKEY,1,,,NN,INTEGER
Trace=100
```

Figure 43. Splitter Configuration File for Loading NATION

The output files from db2split are input to the load in the same way as 6.2.1.1, “Variation 1: All Files” on page 110.

Benefits

- Fast unload from MVS.
- MVS unload independent of AIX platform.

- File transfer independent of database.
- No input file to db2split on AIX thus reducing disk requirements.
- PIOFS not needed.
- CLIO/S transfers data from MVS faster than db2split can process it, so db2split elapse will not be increased.
- Total elapse will be reduced because the file transfer is overlapped with db2split.
- Loads in parallel minimize outages on DB2 Parallel Edition table.
- The transfer can be automated as part of a script on AIX.

Limitations

- The unload must either create an EBCDIC file or an all ASCII file.
- The dataset on MVS will be required for longer because:
  - It must be kept until db2split is run.
  - As db2split is the bottleneck, the transfer rate will be a bit slower than file to file transfer.
- Space for temporary files is required on each node.

6.2.1.4 Variation 4: No Files on AIX

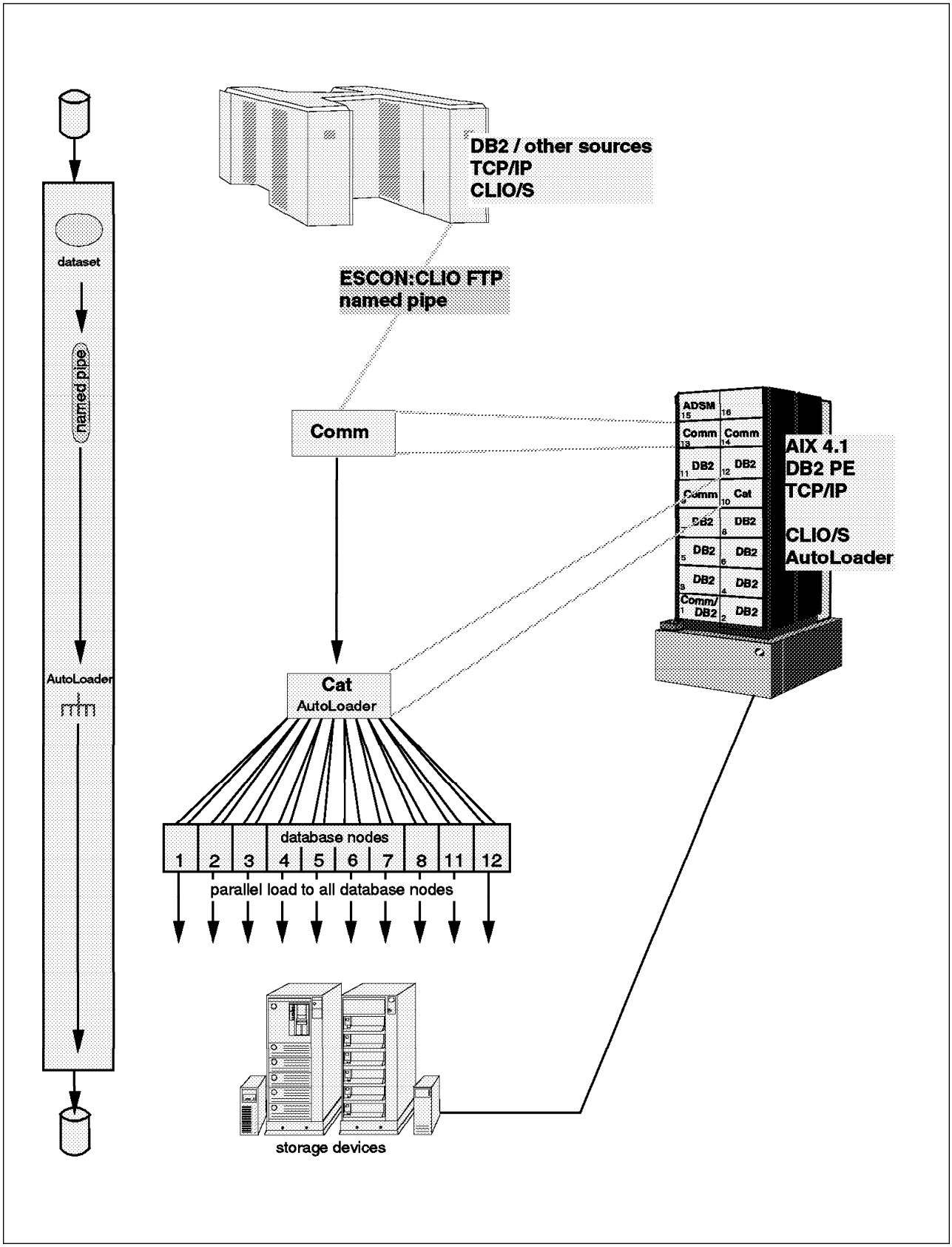


Figure 44. Variation 4 Overview

Figure 44 is an overview of variation 4.

Summary: This is a good solution for initial load, or table refresh where the refresh window is not critical.

It is better than 6.2.1.2, “Variation 2: No Load Files” on page 114 because it avoids temporary files on AIX without increasing the autoloader time on AIX.

It should be compared with:

- 6.2.1.2, “Variation 2: No Load Files” on page 114
- 6.2.1.3, “Variation 3: No Input Files to db2split” on page 117
- 6.2.2.4, “Variation 8: No Files on AIX” on page 133
- 6.2.3.4, “Variation 12: No Files” on page 148

Description

1. An unload process, such as DSNTIAUL, creates a sequential file on MVS in a suitable format for loading into DB2 Parallel Edition.
2. The file is transferred to an AIX pipe directly into autoloader using either TCP/IP FTP or CLIO/S CLPLINK.
3. Autoloader automatically splits the file and loads it onto the nodes.

As an alternative, autoloader can read direct from MVS using TCP/IP FTP. In this case the bottleneck will be the file transfer rather than split. It is an option if CLIO/S is not installed.

Process: A clplink command transfers data from the unload data set on MVS into an AIX pipe. The pipe is read by autoloader. Figure 45 shows the relevant commands.

```
clpmk -r frommvs
clplink -h db2mvs -x db2mvs.nation -w frommvs -U endy -P db2pevi
autoloader -d -s nation.spec tpcd
```

Figure 45. Shell Script for Piping Data from MVS to Autoloader

Note:

1. The clpmk command creates a CLIO/S pipe that can be read by an application (in this case autoloader).
2. The clplink command defines the link that reads data from the file db2mvs.customer, on the remote host db2mvs, and writes it into the pipe.
3. The user ID and password are included here so that there is no need for operator intervention. If you use this technique you must ensure the security of the shell script.
4. The autoloader command points to the config that points to the pipe.

Benefits

- Fast unload from MVS.
- MVS unload independent of AIX platform.

- No temporary files on AIX thus reducing disk requirements.
- PIOFS not needed.
- CLIO/S transfers data from MVS faster than autoloader can process it, so autoloader elapse will not be increased.
- Total elapse will be reduced because the file transfer is overlapped with split and load.
- The transfer can be automated as part of a script on AIX.

Limitations

- The unload must either create an EBCDIC file or an all ASCII file.
- The dataset on MVS will be required for longer because:
  - It must be kept until autoloader has run.
  - As the split processing is the bottleneck, the transfer rate will be a bit slower than file to file transfer.
- The outage on the DB2 Parallel Edition table will be longer, as split takes longer to run than the load on an individual table.

6.2.2 No Unload File on MVS, db2split on AIX

In this set of variations the first step unloads the data from the source into a batch pipe. This pipe is then connected to AIX either into a file or directly into the split process.

This set of variations:

- Requires that the RISC/6000 SP is connected and available during the unload step
- Minimizes the disk requirement in MVS

6.2.2.1 Variation 5: All Files

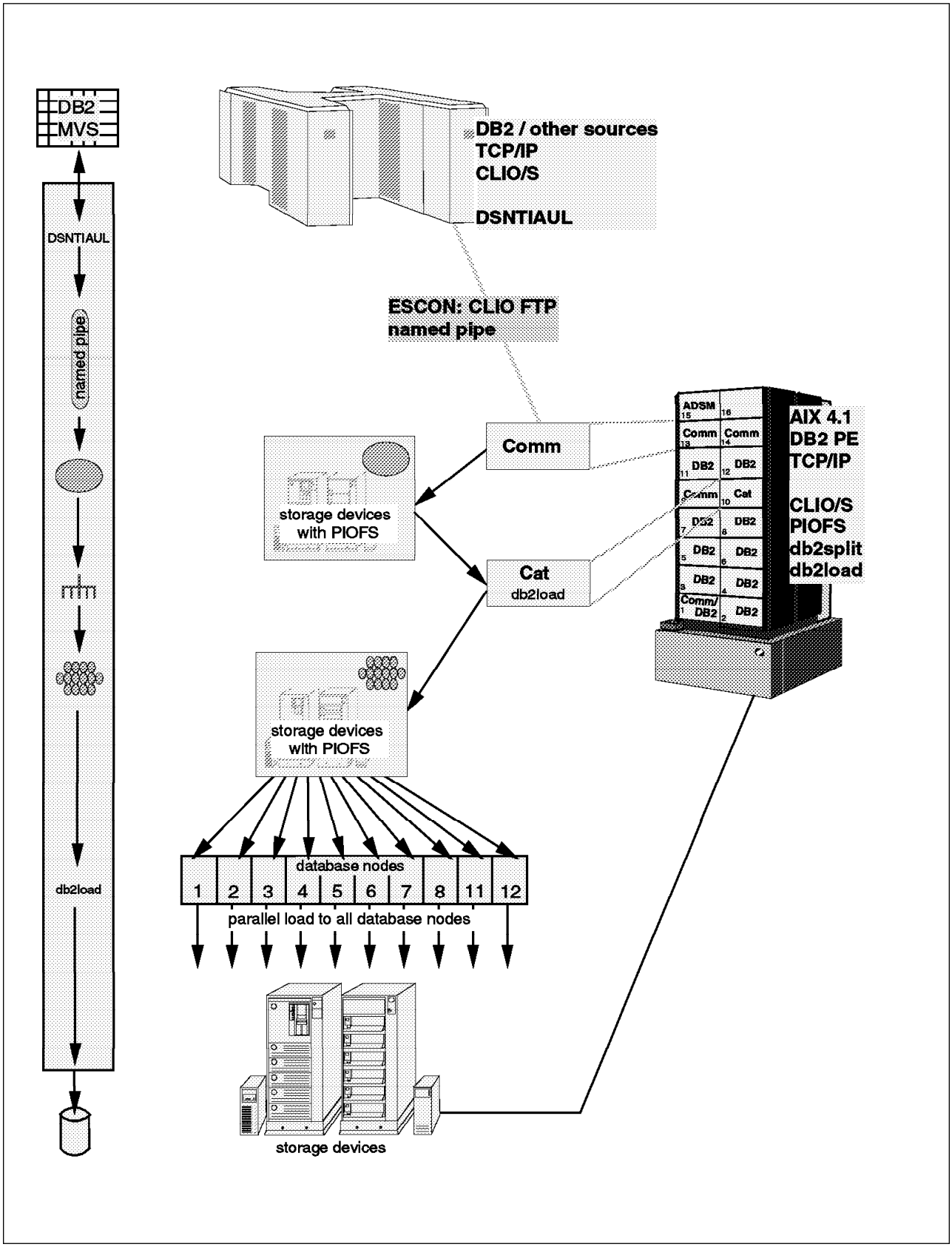


Figure 46. Variation 5 Overview

Figure 46 is an overview of variation 5.

Summary This is a good solution for regularly refreshes of DB2 Parallel Edition tables. It removes the requirement for temporary disk space on MVS and minimizes the outage on the table. It should be compared with the following:

- 6.2.1.1, “Variation 1: All Files” on page 110
- 6.2.2.2, “Variation 6: No Load Files” on page 127

Description:

1. An unload process, such as DSNTIAUL, writes to an MVS Batch Pipe.
2. A second job runs at the same time as the unload and reads the pipe and writes to a file on the RISC/6000 SP.
3. This file is then input to db2split and load exactly like 6.2.1.1, “Variation 1: All Files” on page 110.

Process: Two jobs must be submitted to MVS to run at the same time. A simple way to do this is to have them in the same file and submit the file from TSO. Examples of the jobs are shown in Figure 47

```
//ENDYUNC  JOB (999,POK),NOTIFY=&SYSUID,                              00001001
//         CLASS=A,MSGCLASS=T,MSGLEVEL=(1,1),TIME=1440                00002000
//UNLOAD  EXEC TIAUL                                                  01170000
//SYSIN    DD  DSN=DB2TOPE.UNLOAD(CUSTOMER)                           01340000
//SYSREC00 DD  DSN=BP01.DB2MVS.CUSTOMER,                              01250001
//             DCB=(RECFM=FB,LRECL=249,BUFNO=10),                     01251001
//             SUBSYS=BP01                      <=========PIPE        01270001
//*
//*
//ENDYLINK JOB (999,POK),'CBIPO INSTALL',NOTIFY=&SYSUID,REGION=20M,   01420001
//             CLASS=P,MSGCLASS=T,MSGLEVEL=(1,1)                      01430001
//FCFTP    EXEC CLLINKW,                                              01440001
//  NODE='sp2es01',PIPE='db2mvs.customer.aix'                         01450001
//IN       DD  DSN=BP01.DB2MVS.CUSTOMER,                              01451001
//             DCB=(RECFM=FB,LRECL=249,BUFNO=10),                     01452001
//             SUBSYS=BP01                      <=========PIPE        01453001
//
```

Figure 47. MVS JCL for Piping Data Directly to an AIX File

Note:

1. The SUBSYS=BP01 parameter defines the file as a pipe.
2. The DD card for the pipe must be identical in each job as the first job that starts will create the pipe with the required DCB parameters.
3. The first job to start will wait for the second so the order of the jobs is not important.
4. The ENDYLINK job has CLASS=P. It is suggested that special initiators are set up for pipes to avoid them from filling the normal job initiators.
5. The PIPE parameter defines the AIX file that is to be created.

The db2split and load phases are the same as in 6.2.1.1, “Variation 1: All Files” on page 110.

Benefits

- Fast unload from MVS. CLPLINK transfers data faster than a disk write, so the elapse will be dependent either on the read speed from MVS or the write to the AIX disk. The AIX disks should be able to perform at a similar rate to MVS disks so the elapse should be comparable to the unload in 6.2.1.1, “Variation 1: All Files” on page 110.
- Unload requires no scheduling of jobs on AIX.
- No MVS files required.
- File transfer runs concurrently with unload so the overall elapse is reduced.
- The transfer can be automated as part of a batch job on MVS.
- Loads in parallel minimize outages on the DB2 Parallel Edition table.

Limitations

- The unload must either create an EBCDIC file or an all ASCII file.
- Sufficient temporary space must be available on the RISC/6000 SP for the unloaded data and the split data.
- If the input file for splitter is more than 2 GB it must be set up using PIOFS.
- The unload process is dependent on the connection and the RISC/6000 SP being available.

6.2.2.2 Variation 6: No Load Files

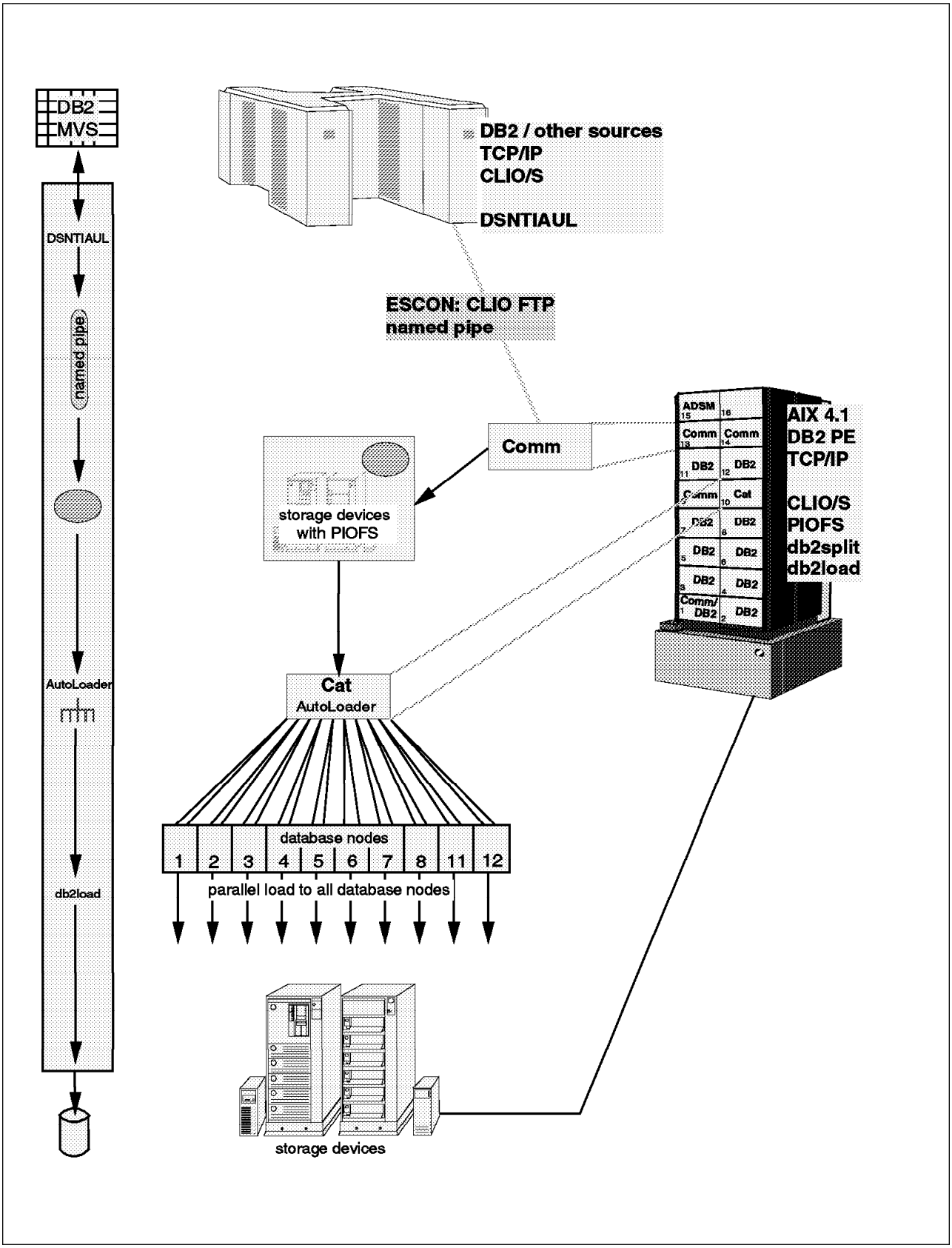


Figure 48. Variation 6 Overview

Figure 48 is an overview of variation 6.

Summary This is a good solution for initial loads or regularly refreshes of DB2 Parallel Edition tables where the outage on the table is not critical. It removes the requirement for temporary disk space on MVS and reduces the temporary AIX disk space requirements. It should be compared with:

- 6.2.1.2, “Variation 2: No Load Files” on page 114
- 6.2.1.4, “Variation 4: No Files on AIX” on page 120
- 6.2.2.1, “Variation 5: All Files” on page 124

Description:

1. An unload process, such as DSNTIAUL, writes to an MVS Batch Pipe.
2. A second job runs at the same time as the unload and reads the pipe and writes to a file on the RISC/6000 SP
3. This file is then input to autoloader and loaded exactly like 6.2.1.2, “Variation 2: No Load Files” on page 114.

Process: The unload is exactly the same as 6.2.2.1, “Variation 5: All Files” on page 124.

The autoloader is exactly the same as 6.2.1.2, “Variation 2: No Load Files” on page 114.

Benefits

- Fast unload from MVS. CLPLINK transfers data faster than a disk write, so the elapse will be dependent either on the read speed from MVS or the write to the AIX disk. The AIX disks should be able to perform at a similar rate to MVS disks so the elapse should be comparable to the unload in 6.2.1.2, “Variation 2: No Load Files” on page 114.
- Unload requires no scheduling of jobs on AIX.
- No MVS files required.
- File transfer runs concurrently with unload so that the overall elapse is reduced.
- The transfer can be automated as part of a batch job on MVS.
- The autoloader removes the need for temporary split files.
- The split and load run in parallel reducing the overall elapse time on the RISC/6000 SP.

Limitations

- The unload must either create an EBCDIC file or an all ASCII file.
- Sufficient temporary space must be available on the RISC/6000 SP for the unloaded data.
- The outage on the RISC/6000 SP table will be longer, as db2split takes longer to run than the load on an individual node.
- If the input file for splitter is more than 2 GB it must be set up using PIOFS.
- The unload process is dependent on the connection and the RISC/6000 SP being available.

6.2.2.3 Variation 7: No Input Files to db2split

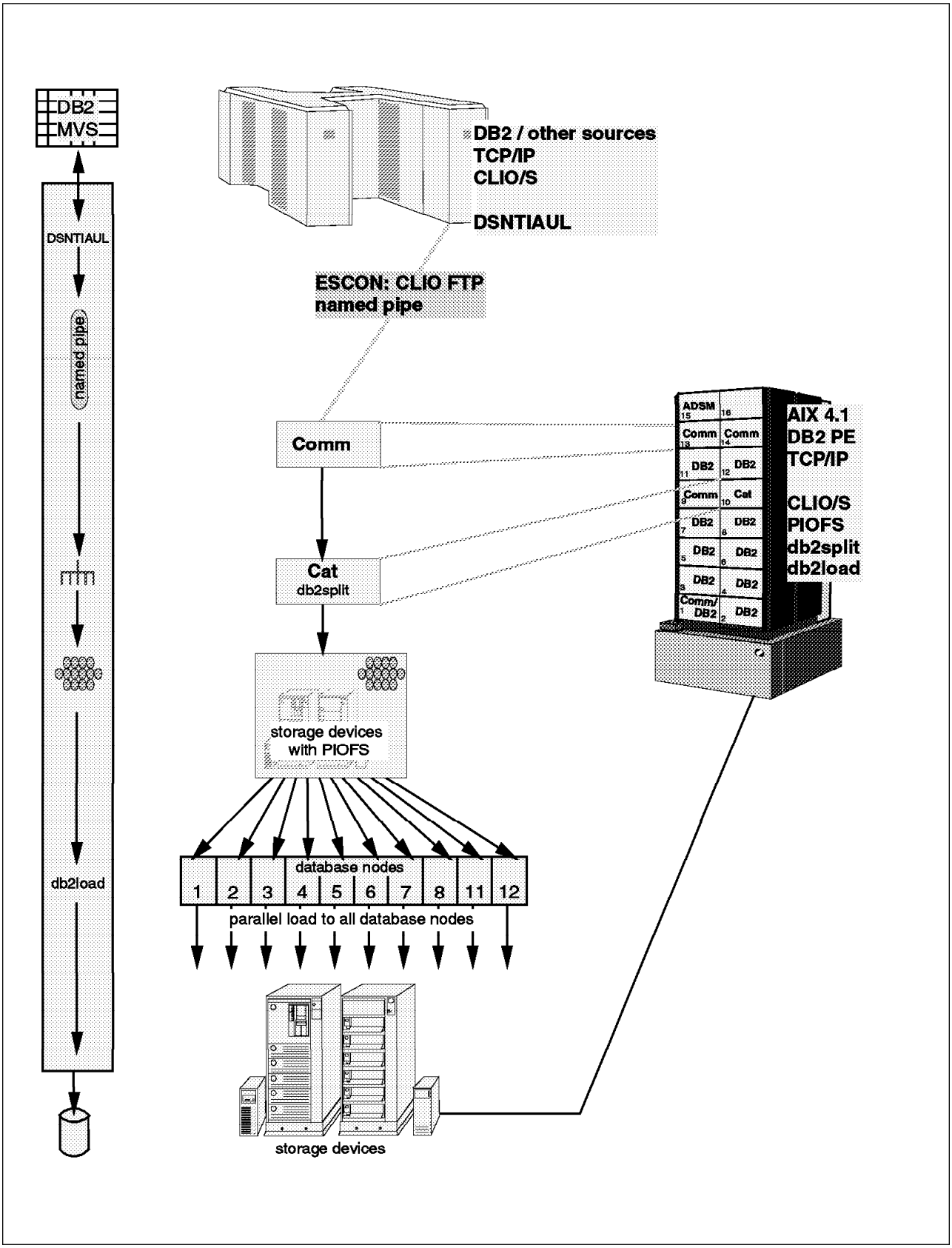


Figure 49. Variation 7 Overview

Figure 49 is an overview of variation 7.

Summary This is a good solution for regularly refreshes of the DB2 Parallel Edition tables. It removes the requirement for temporary disk space on MVS and minimizes the outage on the DB2 Parallel Edition table. However it does require coordination between MVS and AIX operations. It should be compared with:

- 6.2.1.3, “Variation 3: No Input Files to db2split” on page 117
- 6.2.2.1, “Variation 5: All Files” on page 124
- 6.2.2.4, “Variation 8: No Files on AIX” on page 133

Description:

1. An unload process, such as DSNTIAUL, writes to an MVS Batch Pipe.
2. A second job runs at the same time as the unload and reads the pipe and writes to a pipe on the RISC/6000 SP.
3. The AIX pipe is then input to db2split and load exactly like 6.2.1.3, “Variation 3: No Input Files to db2split” on page 117.

Process: The shell script for the db2split must be started first. This creates the pipe that MVS writes to. Then the MVS unload and pipe jobs can be started. The AIX catcher must be started first, otherwise MVS will create a file on the RISC/6000 SP rather than writing to the pipe.

Figure 50 is an example of the shell script for db2split.

```
clpmk -r frommvs
db2split -c /u/db2peusr/cfgfiles/nation.cfg
```

Figure 50. Shell Script for Piping Data from MVS to db2split

Note:

1. The clpmk command creates a CLIO/S pipe that can be read by an application (in this case db2split).
2. The db2split command points to the config that points to the pipe.

Two jobs must be submitted to MVS to run at the same time. A simple way to do this is to have them in the same file and submit the file from TSO. Examples of the jobs are shown in Figure 51 on page 131.

```
//ENDYUNC  JOB (999,POK),NOTIFY=&SYSUID,                                  00001001
//         CLASS=A,MSGCLASS=T,MSGLEVEL=(1,1),TIME=1440                    00002000
//UNLOAD  EXEC TIAUL                                                      01170000
//SYSIN    DD  DSN=DB2TOPE.UNLOAD(CUSTOMER)                               01340000
//SYSREC00 DD  DSN=BP01.DB2MVS.CUSTOMER,                                  01250001
//             DCB=(RECFM=FB,LRECL=249,BUFNO=10),                         01251001
//             SUBSYS=BP01                     <==========PIPE            01270001
//*
//*
//ENDYLINK JOB (999,POK),'CBIPO INSTALL',NOTIFY=&SYSUID,REGION=20M,       01420001
//             CLASS=A,MSGCLASS=T,MSGLEVEL=(1,1)                          01430001
//FCFTP    EXEC CLLINKW,                                                  01440001
//  NODE='sp2es01',PIPE='frommvs'                                         01450001
//IN       DD  DSN=BP01.DB2MVS.CUSTOMER,                                  01451001
//             DCB=(RECFM=FB,LRECL=249,BUFNO=10),                         01452001
//             SUBSYS=BP01                     <==========PIPE            01453001
//
```

Figure 51. MVS JCL for Piping Data Directly to an AIX File

Note:

1. The SUBSYS=BP01 parameter defines the file as a pipe.
2. The DD card for the pipe must be identical in each job because the first job that starts will create the pipe with the required DCB parameters.
3. The first job to start will wait for the second so that the order of the jobs is not important.
4. The PIPE parameter must be the same as the pipe created by the clpmk in Figure 50 on page 130.

The load is the same as in 6.2.1.1, “Variation 1: All Files” on page 110.

Benefits

- Fast unload from MVS. CLPLINK transfers data faster than a disk write, so the elapse will be dependent either on the read speed from DB2 on MVS or the CPU requirement of db2split. In tests using DSNTIAUL with complex SQL db2split was able to easily keep up with the unload. So, the elapse should be comparable to the unload in 6.2.1.1, “Variation 1: All Files” on page 110.
- No MVS files required.
- File transfer runs concurrently with unload and split, so the overall elapse is reduced.
- Loads in parallel minimize outages on the DB2 Parallel Edition table.

Limitations

- The unload must either create an EBCDIC file or an all ASCII file.
- Sufficient temporary space must be available on the RISC/6000 SP for the split data.
- The unload process is dependent on the connection and the RISC/6000 SP being available.

- This variation is dependent on coordination between AIX and MVS operations (to ensure that AIX starts first); this may limit the ability to automate the process.

6.2.2.4 Variation 8: No Files on AIX

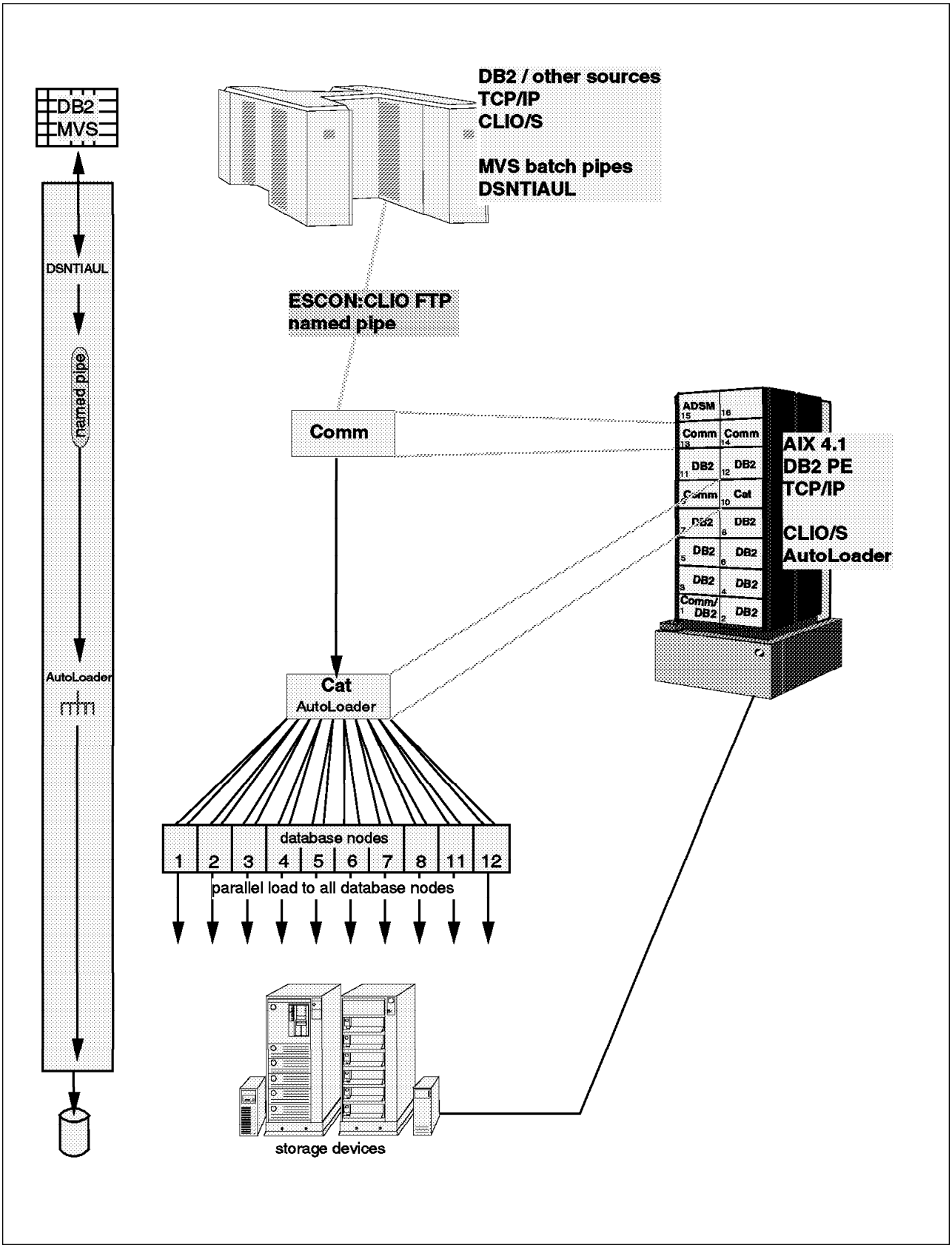


Figure 52. Variation 8 Overview

Figure 52 is an overview of variation 8.

Summary This is a good solution for initial loads and regularly refreshes of DB2 Parallel Edition tables, where the outage on the table is not critical. It removes the requirement for temporary disk space on MVS and RISC/6000 SP. However it does require coordination between MVS and AIX operations. It should be compared with:

- 6.2.1.4, “Variation 4: No Files on AIX” on page 120
- 6.2.2.2, “Variation 6: No Load Files” on page 127
- 6.2.2.3, “Variation 7: No Input Files to db2split” on page 129

Description:

1. An unload process, such as DSNTIAUL, writes to an MVS Batch Pipe.
2. A second job runs at the same time as the unload and reads the pipe and writes to a pipe on the RISC/6000 SP.
3. The AIX pipe is then input to autoloader exactly like 6.2.1.4, “Variation 4: No Files on AIX” on page 120.

Process: The shell script for the autoloader must be started first. This creates the pipe that MVS will write to. The MVS unload and pipe jobs can then be started. The AIX catcher must be started first, otherwise MVS will create a file on the RISC/6000 SP rather than writing to the pipe.

Figure 53 is an example of the shell script for autoloader.

```
clpmk -r frommvs
autoloader -d -s nation.spec tpcd
```

Figure 53. Shell Script for Piping Data from MVS to Autoloader

Note:

1. The clpmk command creates a CLIO/S pipe that can be read by an application (in this case autoloader).
2. The autoloader command points to the spec that points to the config that points to the pipe.

Two jobs must be submitted to MVS to run at the same time. A simple way to do this is to have them in the same file and submit the file from TSO. Examples of the jobs are shown in Figure 47 on page 125

Benefits

- Fast unload from MVS. CLPLINK transfers data faster than a disk write, so the elapse will be dependent either on the read speed from DB2 on MVS or the CPU requirement of db2split. In tests using DSNTIAUL with complex SQL db2split was able to easily keep up with the unload. So the elapse should be comparable to the unload in 6.2.1.4, “Variation 4: No Files on AIX” on page 120.
- No MVS files required.

- File transfer runs concurrently with unload, split and load so the overall elapse is minimized.

Limitations

- The unload must either create an EBCDIC file or an all ASCII file.
- The unload process is dependent on the connection and the RISC/6000 SP being available.
- The outage on the DB2 Parallel Edition table will be longer, as the splitting process takes longer to run than the load on an individual node. It may also be that the unload takes longer than the split so the outage on the DB2 Parallel Edition table is even longer.
- This variation is dependent on coordination between AIX and MVS operations (to ensure that AIX starts first), this may limit the ability to automate the process.
- There is no restart capability as the data moves directly from the source to the target without any intermediate files.

6.2.3 Unload File on MVS, db2split on MVS

In this set of variations, the first step unloads the data from the source into a file on MVS. This is then input to db2split on MVS. The output of the split is then transferred to the RISC/6000 SP for loading.

This set of variations does the following:

- It makes the unload from DB2 (or other sources) independent of the RISC/6000 SP.
- It reduces the CPU requirement on AIX.
- It reduces the disk requirement in AIX.
- It increases the operational complexity because multiple files have to be transferred from MVS to AIX.

6.2.3.1 Variation 9: All Files

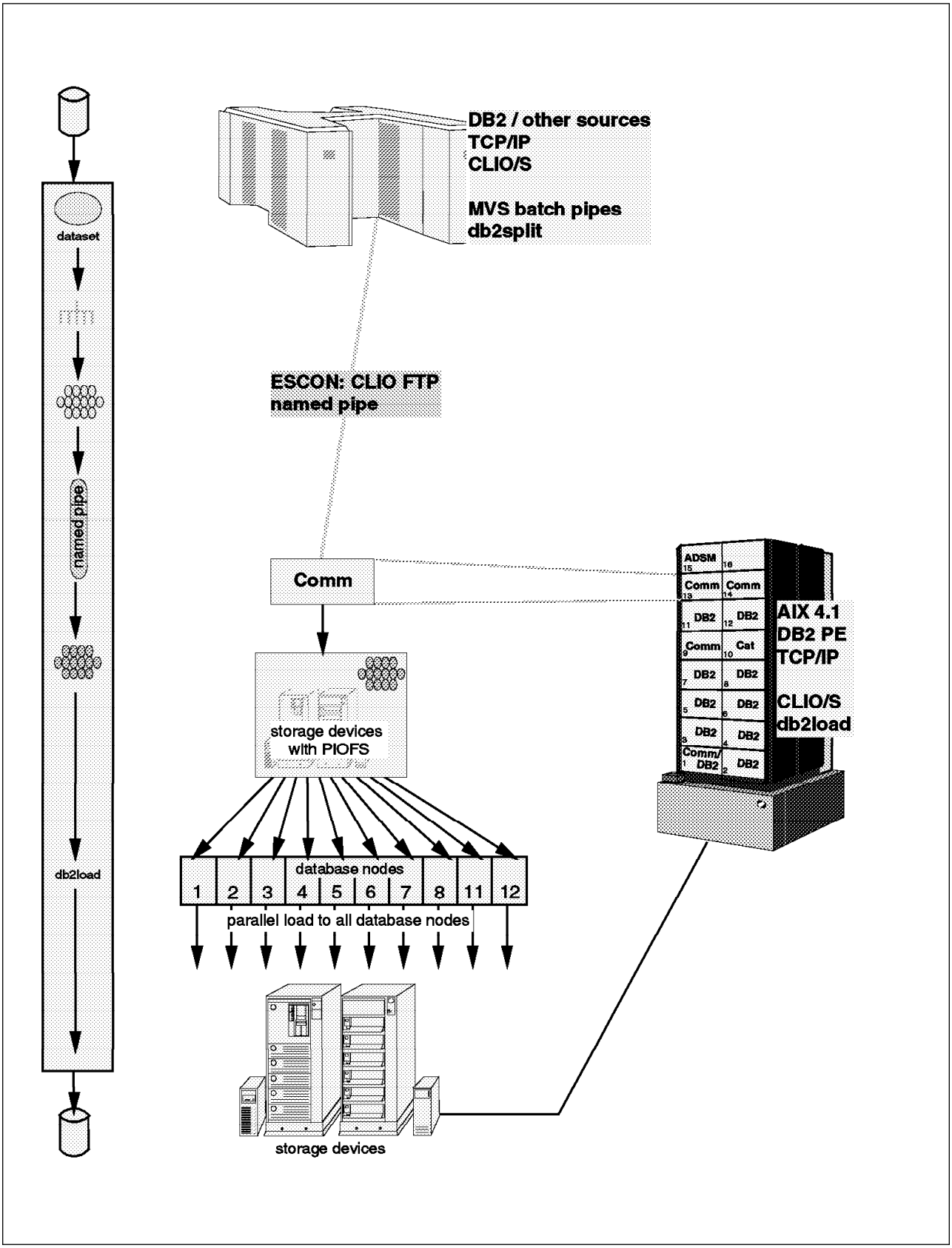


Figure 54. Variation 9 Overview

Summary: This is a base for comparison with other variations, where db2split is on MVS. The number of files that need to be transferred will require careful operations management.

It is a good solution for regular refresh; when you want to minimize the processing on the RISC/6000 SP and minimize the outage on the DB2 Parallel Edition table.

Description

1. An unload process, such as DSNTIAUL, creates a sequential file on MVS in a format suitable for loading into DB2 Parallel Edition.
2. db2split is run, on MVS, and creates an output file per node.
3. The files are transferred to AIX using the fastest transfer method available (CLIO/S is the preferred method). This transfer can be initiated as part of the MVS job stream or the AIX script depending on how quickly you want to release the space on MVS.
4. The load files are loaded on all nodes in parallel.

Process: The file transfer can be initiated from the following:

- MVS, as part of the db2split job
- AIX, as part of the load shell script

Once the file transfer is complete the file on MVS can be deleted. The decision as to which option to choose will depend on factors such as:

- Scheduling constraints on MVS
- Scheduling constraints on RISC/6000 SP
- Availability of disk space on each end

Examples of JCL and shell scripts for both options are included below. The CLPLINK step should only be included at one end.

Figure 55 on page 139 shows the JCL for running the unload and db2split. and the transfers using CLIO/S CLPLINK. The first job step unloads the data using DSNTIAUL. The second step runs db2split and creates a file for each node. Then there is an example of a job to transfer the file to the corresponding node.

```
//UNLOAD  EXEC TIAUL
//SYSIN    DD  DSN=DB2TOPE.UNLOAD(CUST)
//SYSREC00 DD  DSN=DB2MVS.CUSTOMER,DISP=(,CATLG),
//             DCB=(RECFM=FB,LRECL=249,BLKSIZE=27888),
//             UNIT=SYSDA,SPACE=(CYL,(358,0)),
//             VOL=SER=DBPE01
//DB2SPLIT PROC
//*--------------------------------------------------------------------
//SPLIT    EXEC PGM=DB2SPLIT
//STEPLIB  DD DSN=&USERNAME..&PNAME..MOD,DISP=SHR
//         DD DSN=&LIBPRFX..SCEERUN,DISP=SHR
//CONFIG   DD DSN=&USERNAME..&PNAME..RUN(CONFIG),DISP=SHR
//MYINPUT  DD DSN=DB2MVS.CUSTOMER,DISP=SHR
//OUTMAP   DD SYSOUT=*
//DISTFILE DD SYSOUT=*
//LOG      DD SYSOUT=*
//NOD00001 DD DSN=&USERNAME..&PNAME..NOD00001
//         DISP=(NEW,CATLG,DELETE),UNIT=SYSDA,VOL=SER=DBPE15   .,
//         DCB=(RECFM=VB,LRECL=376,BLKSIZSE=27998),
//         SPACE=(CYL,(256,1),RLSE)
//...      ...     ...    Similar attributes for NOD00002 to NOD00011
//NOD00012 DD DSN=&USERNAME..&PNAME..NOD00012
//         DISP=(NEW,CATLG,DELETE),UNIT=SYSDA,VOL=SER=DBPE16   .,
//         DCB=(RECFM=VB,LRECL=376,BLKSIZSE=27998),
//         SPACE=(CYL,(256,1),RLSE)
//         PEND
//*--------------------------------------------------------------------
//STEP1    EXEC DB2SPLIT,USERNAME=DB2MVS,PNAME=CUSTOMER,
//         LIBPRFX=SYS1.CEE.V1R5M0
//
//*  the job below is repeated for each node
//
//ENDYC9   JOB (999,POK),'CBIPO INSTALL',NOTIFY=&SYSUID,REGION=20M,
//             CLASS=P,MSGCLASS=T,MSGLEVEL=(1,1)
//FCFTP    EXEC CLLINKW,
//  NODE='sp2es01:sp2n08',PIPE='customer.node8'
//IN       DD  DSN=DB2MVS.CUSTOMER.NOD00008,
//             DISP=OLD
//
```

Figure 55. JCL for Variation 9

Note:

1. For more details on DB2SPLIT, see 5.5.3, “Job to Execute DB2SPLIT” on page 97.
2. There has to be a job like ENDYC9 for each node.
3. The procedure CLLINKW is described in Figure 15 on page 47.
4. The NODE parameter shows how to specify the jump from the ESCON attached node 1 (sp2es01) to the database node 8 (sp2n08).

Figure 56 on page 140 shows the AIX shell script for the file transfer and load for one node The clpmk and clplink commands get the data from MVS and put it into a file on AIX. The db2 commands run the load. A similar script is needed on each node that will be loaded. For more details on load, see Chapter 5, “Data

Extractions" on page 69. A similar script is needed on each node that will be loaded.

```
clpmk -r customer.node9
clplink -h db2mvs -x db2mvs.customer.nod00009 \
        -w customer.node9 \
        -U endy -P db2pevi
db2 connect to tpcd
db2 "load from customer.node9 of del modified by coldel| replace into customer"
db2 connect reset
```

Figure 56. Shell Script clload.customer01

Figure 57 is an example of a shell script that starts load script on each of the nodes.

```
for i in 01 02 03 04 05 06 07 08 11 12
do
rsh sp2n$i " . ./.profile  ; ./clload.customer$i "
done
```

Figure 57. Shell Script to Start Load on Remote Nodes

Note:

1. In this example, the table is being loaded on nodes 1-8 and 11 and 12.
2. The ./.profile command ensures that the command is run from the correct directory.

Benefits

- Fast unload from MVS.
- MVS unload independent of AIX platform.
- MVS dataset can be released as soon as the file transfer is complete
- File transfer independent of database.
- If file transfer is by tape, then no hardware connection is required.
- The transfer can be initiated either from MVS or AIX depending on where the disk space is more critical.
- The transfer can be automated as part of a batch job on MVS or a script on AIX.
- Loads in parallel minimize outages on the DB2 Parallel Edition table.

Limitations

- The unload must either create an EBCDIC file or an all ASCII file.
- The steps are sequential so that the total time will be long.
- There are temporary files at each step and space must be found for them.
- There are many small files that are being transferred between MVS and AIX and the automation of this process must be carefully planned.

6.2.3.2 Variation 10: No db2split Output

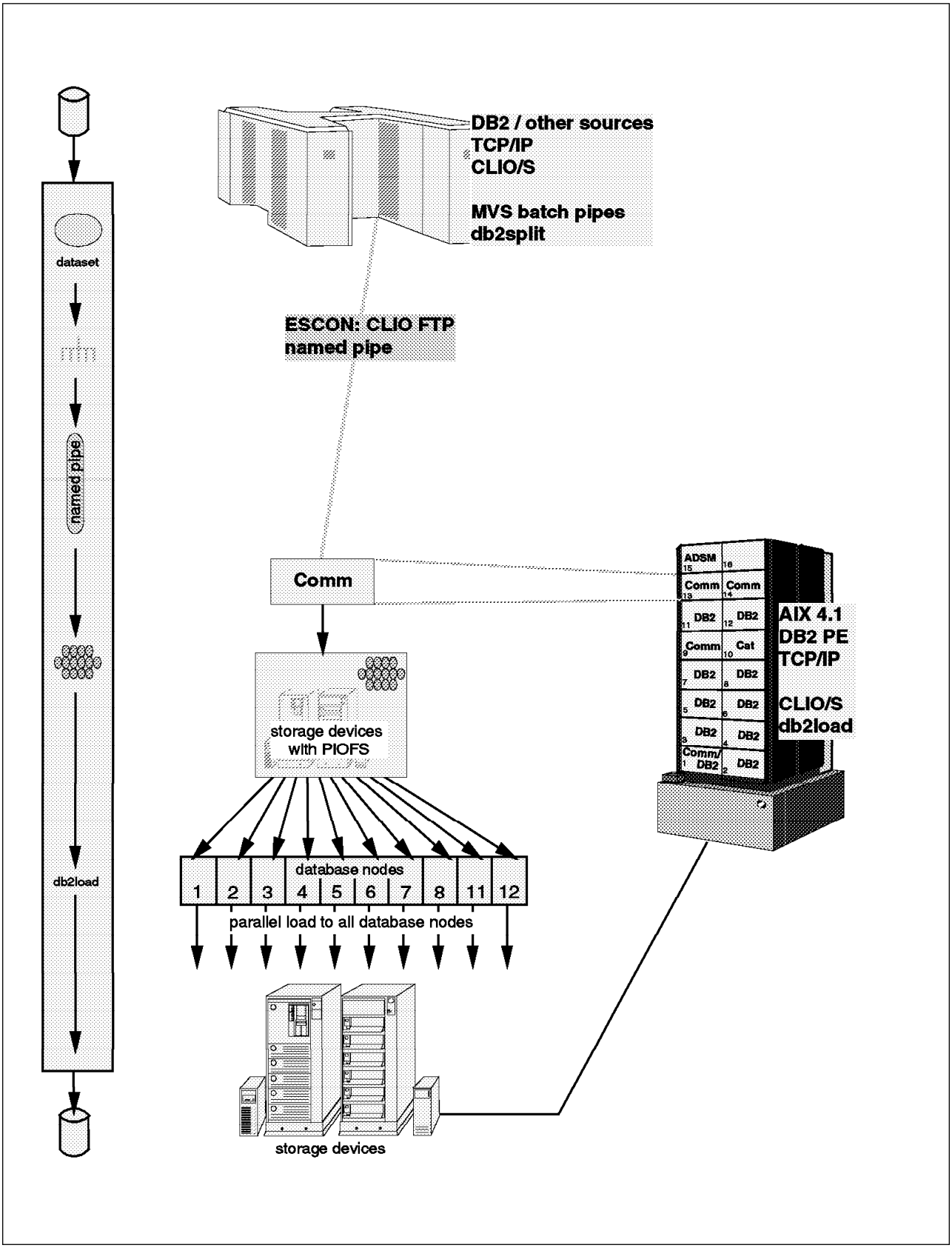


Figure 58. Variation 10 Overview

Summary: This is a solution for regular refresh when you want to minimize the processing on the RISC/6000 SP and minimize the outage on the DB2 Parallel Edition table. It reduces the need for temporary file on MVS, as compared to 6.2.3.1, “Variation 9: All Files” on page 137. However, to do this you need many CLPLINK jobs running in parallel. The extra complexity, and possibility of a single failure, probably will outweigh the benefit of the disk saving.

It should be compared with:

- 6.2.1.3, “Variation 3: No Input Files to db2split” on page 117
- 6.2.3.1, “Variation 9: All Files” on page 137

Description

1. An unload process, such as DSNTIAUL, creates a sequential file on MVS in a format suitable for loading into DB2 Parallel Edition.
2. db2split is run, on MVS, and creates an output file per node, which is fed directly into a batch pipe. The batch pipes are connected to multiple CLPLINK jobs which each write a file to a node.
3. The load files are loaded on all nodes in parallel

Process: Figure 59 on page 143 shows the JCL for running the unload and db2split, and the transfers using CLIO/S CLPLINK. The first job step unloads the data using DSNTIAUL. The second step runs db2split and creates a file for each node.

```
//UNLOAD  EXEC TIAUL
//SYSIN    DD  DSN=DB2TOPE.UNLOAD(ORDERS)
//SYSREC00 DD  DSN=DB2MVS.ORDERS,DISP=(,CATLG),
//             DCB=(RECFM=FB,LRECL=249,BLKSIZE=27888),
//             UNIT=SYSDA,SPACE=(CYL,(358,0)),
//             VOL=SER=DBPE01
//*-------------------------------------------------------------------
//SPLIT    EXEC PGM=DB2SPLIT
//CONFIG   DD DSN=DB2MVS.CONFIG(ORDERS),DISP=SHR
//MYINPUT  DD DSN=DB2MVS.ORDERS,DISP=SHR
//OUTMAP   DD SYSOUT=*
//DISTFILE DD SYSOUT=*
//LOG      DD SYSOUT=*
//NOD00001 DD DSN=DB2MVS.ODRERS.NOD00001
//         DCB=(RECFM=VB,LRECL=376,BUFNO=3),
//         SUBSYS=BP01
//...      ...     ...    Similar attributes for NOD00002 to NOD00011
//*-------------------------------------------------------------------
//
//
//ENDYC8   JOB (999,POK),'CBIPO INSTALL',NOTIFY=&SYSUID,REGION=20M,
//             CLASS=P,MSGCLASS=T,MSGLEVEL=(1,1)
//FCFTP    EXEC CLLINKW,
//  NODE='sp2es01:sp2n08',PIPE='customer.node1'
//IN       DD DSN=DB2MVS.ORDERS.NOD00008
//         DCB=(RECFM=VB,LRECL=376,BUFNO=3),
//         SUBSYS=BP01
//...      ...     ...    Similar jobs for other nodes
//
```

Figure 59. JCL for Variation 10

Note:

1. For more details on DB2SPLIT, see 5.5.3, “Job to Execute DB2SPLIT” on page 97.
2. There has to be a job like ENDYC8 for each node.
3. The procedure CLLINKW is described in Figure 15 on page 47.
4. The NODE parameter shows how to specify the jump from the ESCON attached node 1 (SP2ES01) to the database node 8 (SP2N08).
5. You have to have an initiator and a CLIO/S link for each node of the table.

The load process is the same as 6.2.1.1, “Variation 1: All Files” on page 110.

Benefits

- Fast unload from MVS.
- MVS unload independent of AIX platform.
- MVS dataset can be released as soon as db2split is complete.
- File transfer independent of database.
- The transfer can be automated as part of a batch job on MVS.
- Loads in parallel minimize outages on the DB2 Parallel Edition table.

- Transfer is done in parallel with db2split.

Limitations

- The unload must either create an EBCDIC file or an all ASCII file.
- There are many small pipes open at the same time between MVS and AIX and the automation of this process must be carefully planned.

6.2.3.3 Variation 11: No load input (Piped load input)

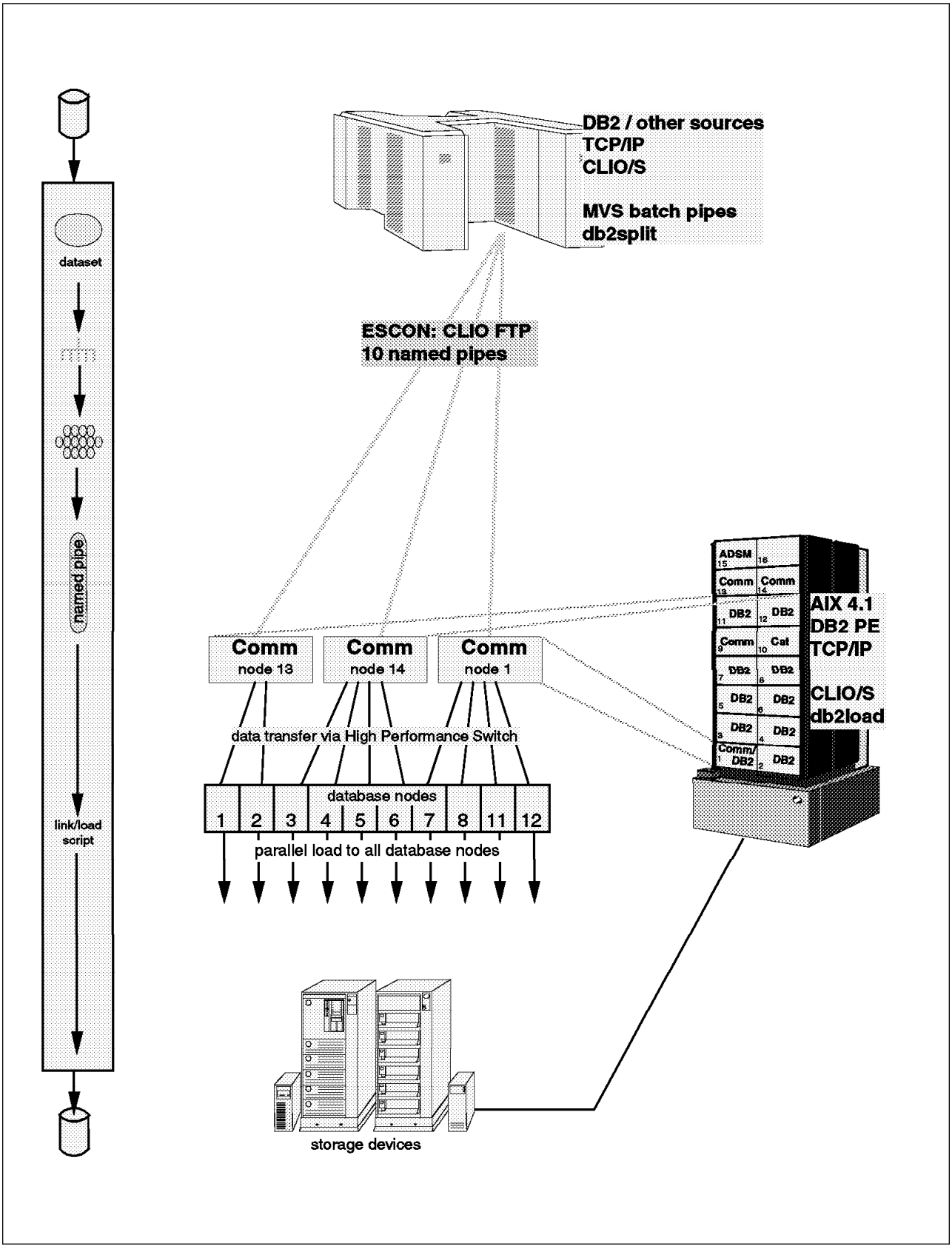


Figure 60. Variation 11 Overview

Summary: It is a good solution for regular refresh; when you want to minimize the processing on the RISC/6000 SP, minimize the outage on the DB2 Parallel Edition table, and remove the need for temporary files on the RISC/6000 SP.

In general, it is preferable to 6.2.3.1, “Variation 9: All Files” on page 137 because it reduces the elapse time, removes the need for AIX temporary files, and is less complex to run.

When compared to 6.2.1.3, “Variation 3: No Input Files to db2split” on page 117, it replaces AIX disk and CPU by MVS disk and CPU without increasing the operational complexity; it also reduces the elapse time because the transfer is done in multiple parallel streams.

Description

1. An unload process, such as DSNTIAUL, creates a sequential file on MVS in a format suitable for loading into DB2 Parallel Edition.
2. db2split is run, on MVS, and creates an output file per node.
3. The files are transferred to AIX using CLPLINK and piped straight into load on the nodes.
4. The loads can be run on all nodes in parallel.

Process: The unload and split are the same as 6.2.3.1, “Variation 9: All Files” on page 137.

The transfer and load are initiated from the node using a shell script, as shown in Figure 61.

```
clpmk -r frommvs
clplink -h db2mvs -x db2mvs.customer.nod00001
        -w frommvs -U endy -P db2pevi
db2 connect to tpcd
db2 "load from frommvs of del modified by coldel| replace into customer"
db2 connect reset
```

Figure 61. Shell Script for Piping Data from MVS to Load

Note:

1. The clpmk command creates a CLIO/S pipe that can be read by an application (in this case load).
2. The clplink command defines the link that reads data from the file db2mvs.customer, on the remote host db2mvs, and writes it into the pipe.
3. The user ID and password are included here so that there is no need for operator intervention. If you use this technique you must ensure the security of the shell script.
4. A script of this type must run on each of the nodes.

Benefits

- Fast unload from MVS
- MVS unload independent of AIX platform

- Loads in parallel with transfer, reducing elapse time
- Loads in parallel minimize outages on the DB2 Parallel Edition table
- No temporary files on AIX

Limitations

- The unload must either create an EBCDIC file or an all ASCII file.
- Each of the loads requires a CLIO/S pipe and an MVS initiator and this may limit the number of loads that can be done in parallel.

6.2.3.4 Variation 12: No Files

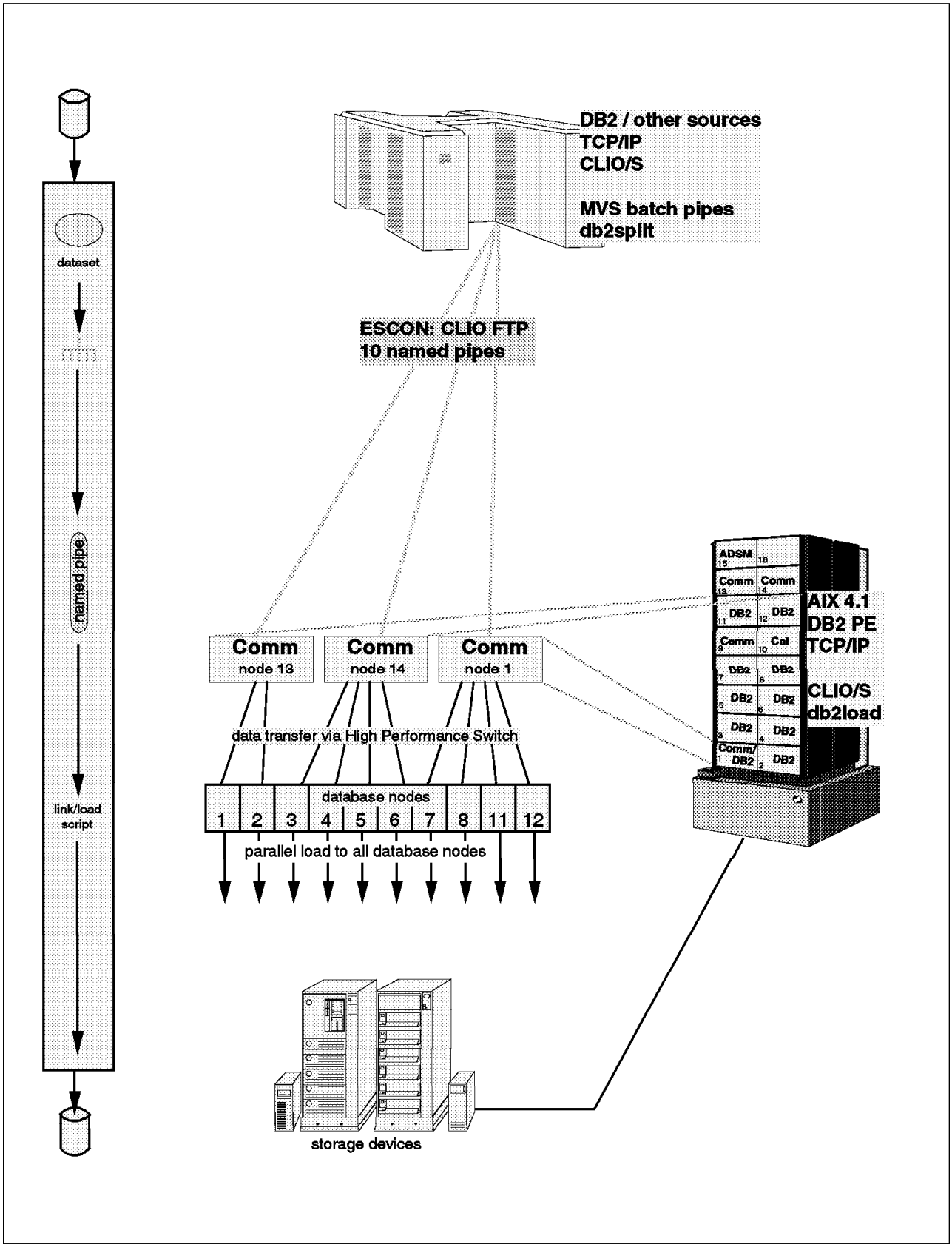


Figure 62. Variation 12 Overview

Summary: This is a possible solution for initial load, or table refresh where the refresh window is not critical.

The db2split is likely to be the bottleneck rather than the transfer. So this variation is operationally more complex than 6.2.1.4, “Variation 4: No Files on AIX” on page 120, which has one transfer and autoloader, without any real benefits.

It should be compared with:

- 6.2.1.4, “Variation 4: No Files on AIX” on page 120
- 6.2.3.3, “Variation 11: No load input (Piped load input)” on page 145
- 6.2.4.4, “Variation 16: No Files” on page 163

Description

1. An unload process, such as DSNTIAUL, creates a sequential file on MVS in a format suitable for loading into DB2 Parallel Edition.
2. db2split is run, on MVS, and creates an output file per node, that is piped straight into CLPLINK.
3. CLPLINK pipes the data straight into the load on the nodes.
4. The loads can be run on all nodes in parallel.

Process: The unload and split is identical to 6.2.3.2, “Variation 10: No db2split Output” on page 141.

Figure 63 on page 150 shows an example of an unload, db2split and a CLPLINK job for reading the pipe written by db2split and writing into a pipe for load.

```
//UNLOAD  EXEC TIAUL
//SYSIN    DD  DSN=DB2TOPE.UNLOAD(ORDERS)
//SYSREC00 DD  DSN=DB2MVS.CUSTOMER,DISP=(,CATLG),
//             DCB=(RECFM=FB,LRECL=249,BLKSIZE=27888),
//             UNIT=SYSDA,SPACE=(CYL,(358,0)),
//             VOL=SER=DBPE01
//*-------------------------------------------------------------------
//SPLIT    EXEC PGM=DB2SPLIT
//CONFIG   DD DSN=DB2MVS.CONFIG(ORDERS),DISP=SHR
//MYINPUT  DD DSN=DB2MVS.ORDERS,DISP=SHR
//OUTMAP   DD SYSOUT=*
//DISTFILE DD SYSOUT=*
//LOG      DD SYSOUT=*
//NOD00001 DD DSN=DB2MVS.ORDERS.NOD00001
//         DCB=(RECFM=VB,LRECL=376,BUFNO=3),
//         SUBSYS=BP01
//...      ...     ...    Similar statements for other nodes
//*-------------------------------------------------------------------
//
//*  the job below is repeated for each node
//
//ENDYC8   JOB (999,POK),'CBIPO INSTALL',NOTIFY=&SYSUID,REGION=20M,
//             CLASS=P,MSGCLASS=T,MSGLEVEL=(1,1)
//FCFTP    EXEC CLLINKW,
//  NODE='sp2es01:sp2n08',PIPE='orders8'
//IN       DD  DSN=DB2MVS.ORDERS.NOD00008,
//         DCB=(RECFM=VB,LRECL=376,BUFNO=3),
//         SUBSYS=BP01
//
```

Figure 63. JCL for Variation 12

Note:

1. For more details on DB2SPLIT, see 5.5.3, “Job to Execute DB2SPLIT” on page 97.
2. There has to be a job like ENDYC8 for each node.
3. The procedure CLLINKW is described in Figure 15 on page 47.
4. The NODE parameter shows how to specify the jump from the ESCON attached node 1 (SP2ES01) to the database node 8 (SP2N08).

Figure 64 shows a shell script that reads the pipe from CLPLINK and loads the data.

```
clpmk -r orders8
db2 connect to tpcd
db2 "load from orders8 of del modified by coldel| replace into orders    "
db2 connect reset
```

Figure 64. Shell Script for Loading Data from a Pipe

Note:

1. There has to a equivalent shell script on each node to be loaded.

2. All these shell scripts must be started before the CLPLINK jobs on MVS because otherwise CLPLINK will write a file rather than into the pipe.

Benefits

- Fast unload from MVS.
- MVS unload independent of AIX platform.
- No temporary files on AIX thus reducing disk requirements.
- PIOFS not needed.
- CLIO/S transfers data from MVS faster than db2split can process it, so db2split elapse will not be increased.
- Total elapse will be reduced because the file transfer is overlapped with split and load.

Limitations

- The unload must either create an EBCDIC file or an all ASCII file.
- The dataset on MVS will be required for longer because:
  - It must be kept until db2split is run.
  - As db2split is the bottleneck, the transfer rate will be a bit slower than file to file transfer.
- The outage on the DB2 Parallel Edition table will be longer, as db2split takes longer to run than the load on an individual table
- The complexity of ensuring that all the load pipes and process on the nodes are up before the CLPLINK jobs, makes this a complex operational scenario.

6.2.4 No Unload File on MVS, db2split on MVS

In this set of variations the first job unloads the data from the source into a BatchPipe and this is the input to db2split on MVS.

This set of variations does the following:

- It makes the unload from DB2 (or other sources) independent of the RISC/6000 SP.
- It reduces the CPU requirement on AIX.
- It reduces the disk requirement on AIX.
- It increases the operational complexity because multiple files have to be transferred from MVS to AIX.
- It reduces the temporary file requirement on MVS.

6.2.4.1 Variation 13: Both Files

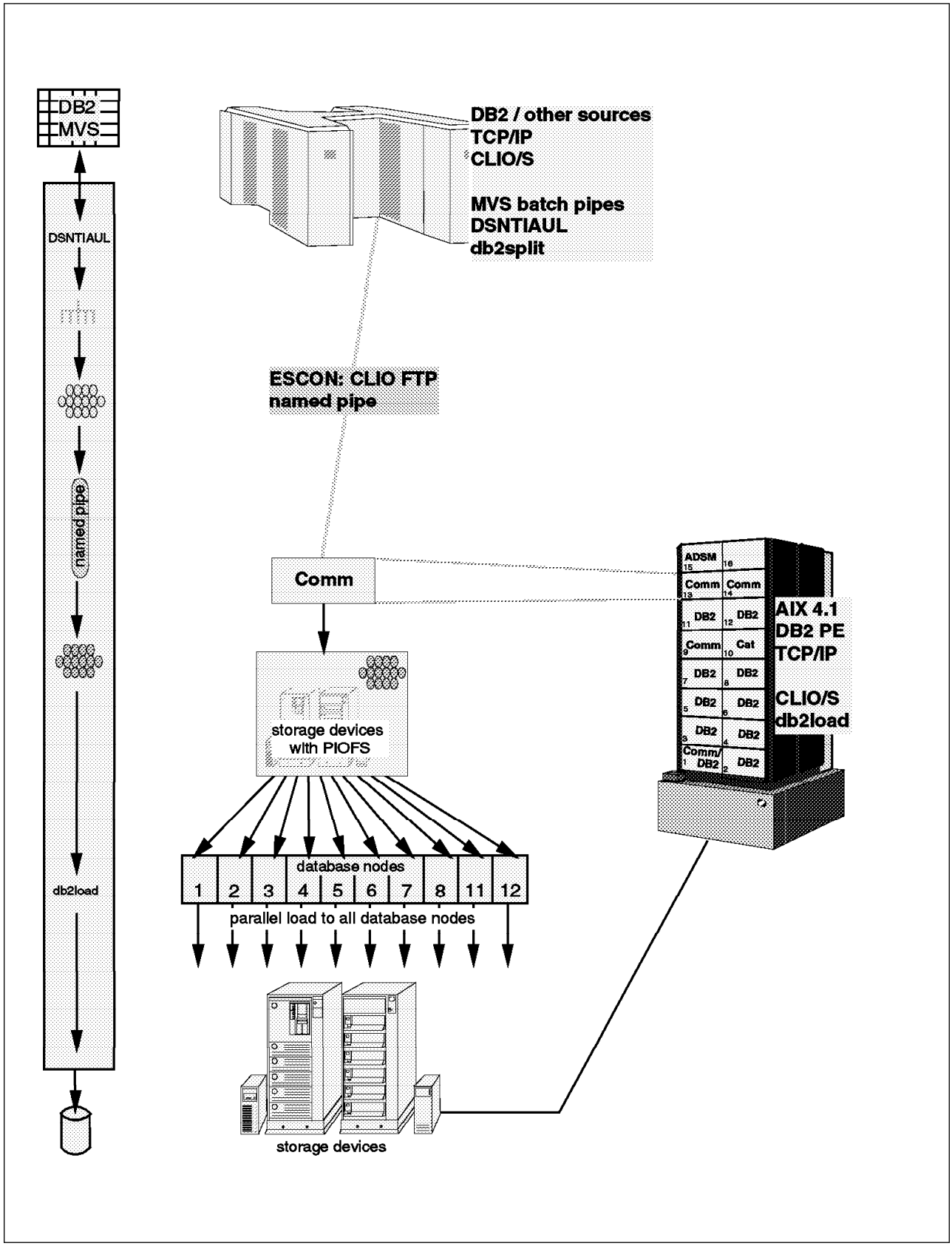


Figure 65. Variation 13 Overview

Summary: This is a good solution for regular refresh when you want to minimize the processing on the RISC/6000 SP and minimize the outage on the DB2 Parallel Edition table.

The number of files that need to be transferred will require careful operations management.

It is probably a better solution than 6.2.3.1, “Variation 9: All Files” on page 137, because it removes the need for one temporary file and reduces the elapsed time. However, the combined CPU requirement of the unload and db2split may marginally increase the unload elapse.

Description

1. An unload process, such as DSNTIAUL, creates a sequential file on MVS in a form suitable for loading into DB2 Parallel Edition. This file is not written to disk but straight into a BatchPipe.
2. db2split is run, on MVS, using the pipe as input and creates an output file per node.
3. The files are transferred to AIX files using the fastest transfer method available (CLIO/S is the preferred method). This transfer can be initiated as part of the MVS job stream or the AIX script depending on how quickly you want to release the space on MVS.
4. The load files are loaded on all nodes in parallel.

Process: The process is identical to 6.2.3.1, “Variation 9: All Files” on page 137, except that the output of the unload is piped directly into db2split. An example of the JCL for the unload and db2split is shown in Figure 66 on page 155

```
//ENDYUNC  JOB (999,POK),NOTIFY=&SYSUID,                                00001001
//         CLASS=A,MSGCLASS=T,MSGLEVEL=(1,1),TIME=1440                  00002000
//UNLOAD  EXEC TIAUL                                                    01170000
//SYSIN    DD  DSN=DB2TOPE.UNLOAD(CUSTOMER)                             01340000
//SYSREC00 DD  DSN=BP01.DB2MVS.CUSTOMER,                                01250001
//             DCB=(RECFM=FB,LRECL=249,BUFNO=10),                       01251001
//             SUBSYS=BP01                     <=========PIPE           01270001
//*
//*
//ENDYSPL  JOB (999,POK),NOTIFY=&SYSUID,                                00001001
//         CLASS=A,MSGCLASS=T,MSGLEVEL=(1,1),TIME=1440                  00002000
//*-------------------------------------------------------------------
//SPLIT    EXEC PGM=DB2SPLIT
//CONFIG   DD DSN=DB2MVS.CONFIG(CUSTOMER),DISP=SHR
//MYINPUT  DD  DSN=BP01.DB2MVS.CUSTOMER,                                01250001
//             DCB=(RECFM=FB,LRECL=249,BUFNO=10),                       01251001
//             SUBSYS=BP01                     <=========PIPE           01270001
//OUTMAP   DD SYSOUT=*
//DISTFILE DD SYSOUT=*
//LOG      DD SYSOUT=*
//NOD00001 DD DSN=DB2MVS.CUSTOMER.NOD00001,
//         DISP=(NEW,CATLG,DELETE),UNIT=SYSDA,VOL=SER=DBPE15,
//         DCB=(RECFM=VB,LRECL=376,BLKSIZSE=27998),
//         SPACE=(CYL,(256,1),RLSE)
//...      ...     ...    Similar statements for other nodes
//*-------------------------------------------------------------------
//
//*  the job below is repeated for each node
//
//ENDYC8   JOB (999,POK),'CBIPO INSTALL',NOTIFY=&SYSUID,REGION=20M,
//             CLASS=P,MSGCLASS=T,MSGLEVEL=(1,1)
//FCFTP    EXEC CLLINKW,
//  NODE='sp2es01:sp2n08',PIPE='customer.node8'
//IN       DD  DSN=DB2MVS.CUSTOMER.NOD00008,
//             DISP=OLD
//
```

Figure 66. JCL for Variation 13

Note:

1. For more details on DB2SPLIT, see 5.5.3, “Job to Execute DB2SPLIT” on page 97.
2. There has to be a job like ENDYC8 for each node.
3. The procedure CLLINKW is described in Figure 15 on page 47.
4. The NODE parameter shows how to specify the jump from the ESCON attached node 1 (sp2es01) to the database node 8 (sp2n08).

Benefits

- Reduced need for temporary files on MVS.
- MVS unload independent of AIX platform.
- MVS dataset can be released as soon as the file transfer is complete
- File transfer independent of database.

- If the file transfer is by tape, then no hardware connection is required.
- The transfer can be initiated either from MVS or AIX depending on where the disk space is more critical.
- The transfer can be automated as part of a batch job on MVS or a script on AIX.
- Loads in parallel minimize outages on the DB2 Parallel Edition table.

Limitations

- The unload must either create an EBCDIC file or an all ASCII file.
- The unload and db2split can both use significant CPU. In this scenario they run together and thus may make the unload run slower than if it writes to a file.
- There are many small files that are being transferred between MVS and AIX and the automation of this process must be carefully planned.

6.2.4.2 Variation 14: No Input Files to Load

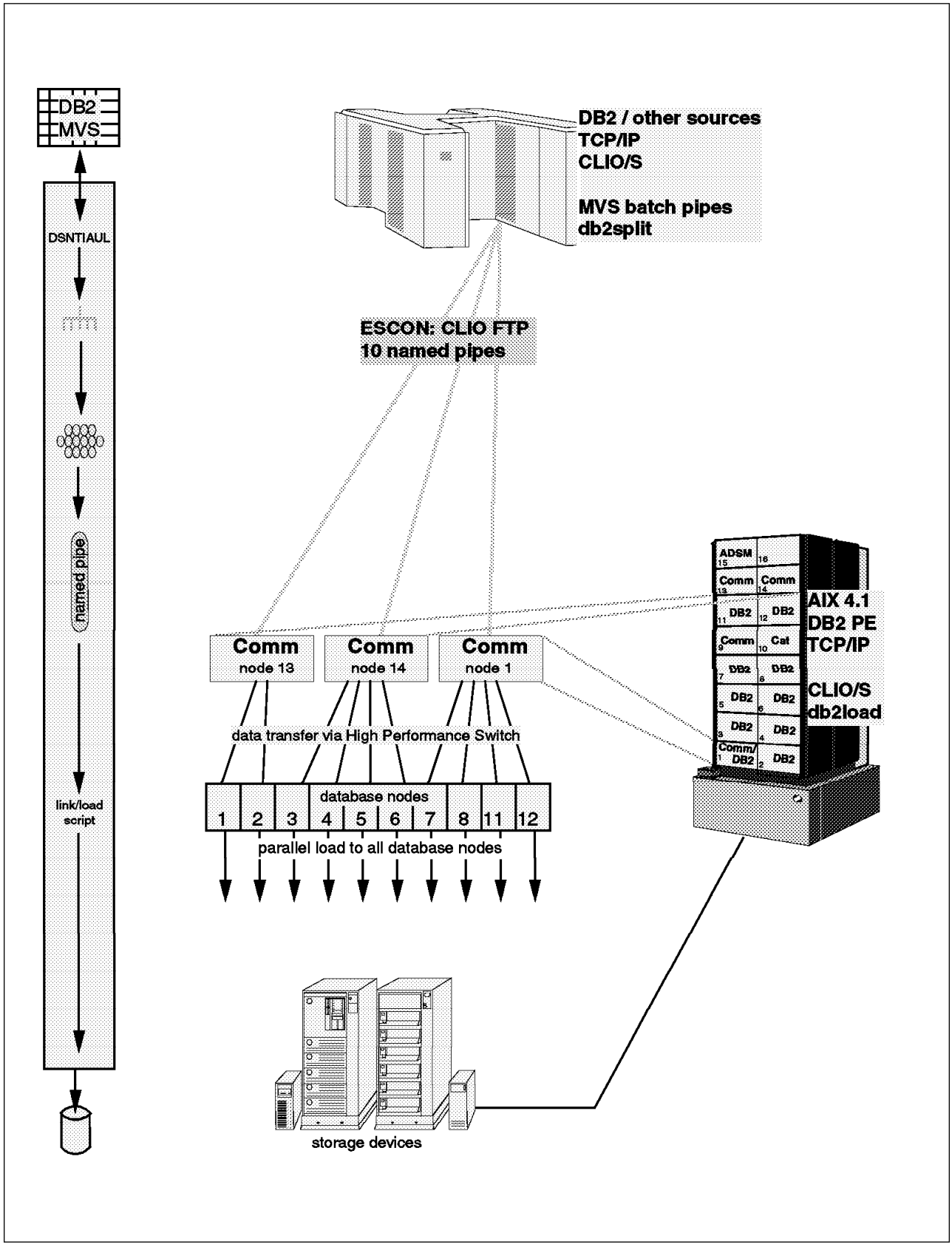


Figure 67. Variation 14 Overview

Summary: This is a good solution for regular refresh when you want to minimize the processing on the RISC/6000 SP and minimize the outage on the DB2 Parallel Edition table.

It is probably a better solution than 6.2.3.1, “Variation 9: All Files” on page 137 because it removes the need for one temporary file and reduces the elapse time. However, the combined CPU requirement of the unload and db2split may marginally increase the unload elapse.

It is a better solution than 6.2.4.1, “Variation 13: Both Files” on page 153 if there is sufficient bandwidth for the multiple file transfers for the parallel loads. It reduces elapse time, removes the need for temporary files on AIX and simplifies the operations.

It should also be compared with 6.2.2.3, “Variation 7: No Input Files to db2split” on page 129 and 6.2.3.3, “Variation 11: No load input (Piped load input)” on page 145.

Description

1. An unload process, such as DSNTIAUL, creates a sequential file on MVS in a form suitable for loading into DB2 Parallel Edition. This file is not written to disk but straight into a BatchPipe.
2. db2split is run, on MVS, using the pipe as input and creates an output file per node.
3. The files are transferred, in parallel, using CLPLINK, to pipes on the nodes.
4. The pipes are input to db2load which is run on all nodes in parallel.

Process: The process is identical to 6.2.3.3, “Variation 11: No load input (Piped load input)” on page 145 except that the output of the unload is piped directly into db2split, like 6.2.4.1, “Variation 13: Both Files” on page 153.

Benefits

- Reduced need for temporary files on MVS
- MVS unload independent of AIX platform
- No requirement for AIX temporary files
- File transfer in parallel with load reduces overall elapse
- Loads in parallel minimize outages on the DB2 Parallel Edition table

Limitations

- The unload must either create an EBCDIC file or an all ASCII file.
- The unload and db2split can both use significant CPU. In this scenario they run together and thus may make the unload run slower than if it writes to a file.
- The loads will initiate parallel file transfers and the speed of these may be limited by the bandwidth of the channel connection.

6.2.4.3 Variation 15: No Output from db2split

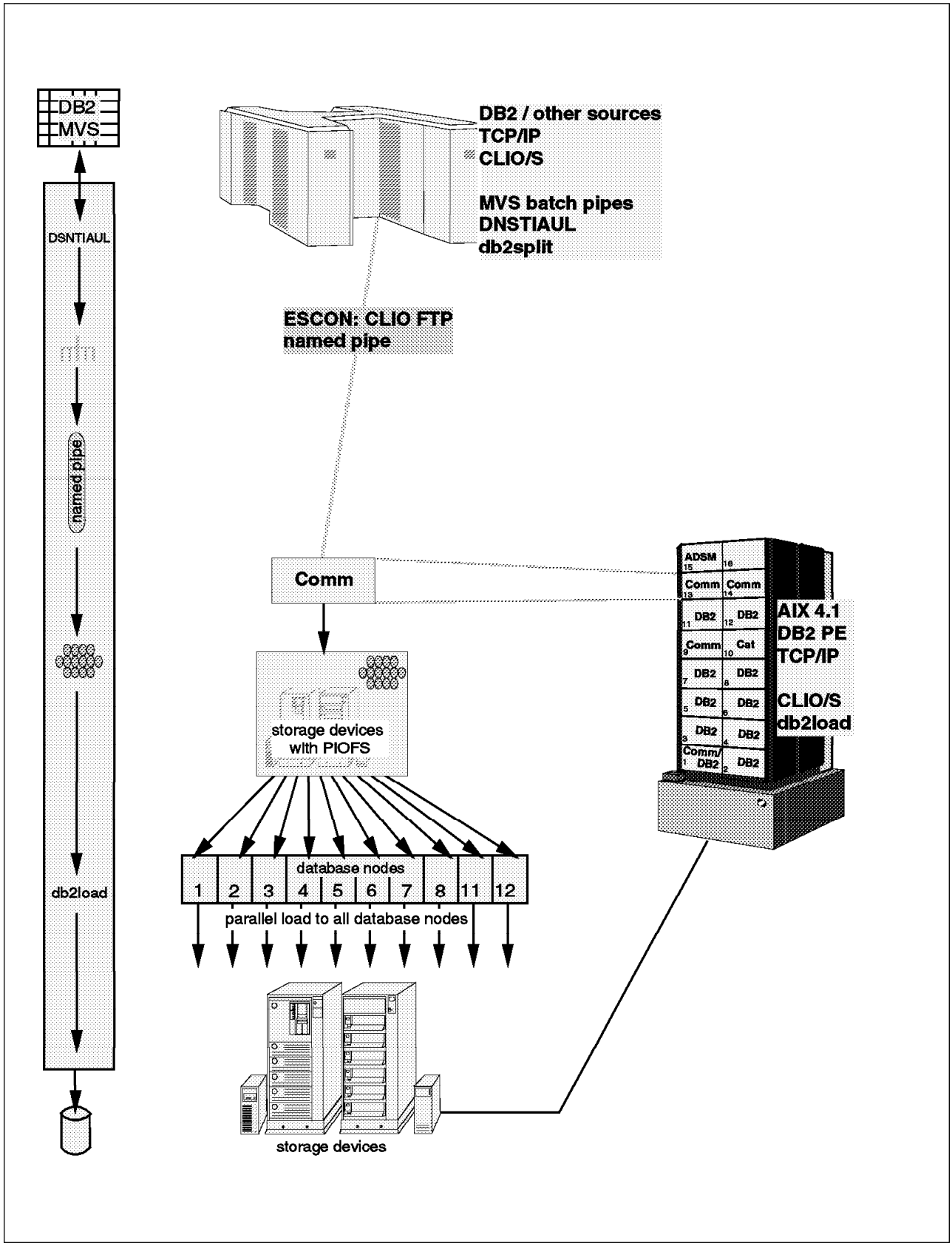


Figure 68. Variation 15 Overview

Summary: This is a solution for regular refresh; when you want to minimize the processing on the RISC/6000 SP and minimize the outage on the DB2 Parallel Edition table. It removes the need for temporary file on MVS. However to do this you need many CLPLINK jobs running in parallel. The extra complexity, and possibility of a single failure, probably will outweigh the benefit of the disk saving.

It should be compared with:

- 6.2.2.3, “Variation 7: No Input Files to db2split” on page 129
- 6.2.4.2, “Variation 14: No Input Files to Load” on page 157

Description

1. An unload process, such as DSNTIAUL, creates a sequential file on MVS in a suitable form for loading into DB2 Parallel Edition. This file is not written to disk but straight into a BatchPipe.
2. This pipe is input to db2split, on MVS, which creates an output file per node, which is fed directly into a batch pipe. The batch pipes are connected to multiple CLPLINK jobs which each write a file to a node.
3. The load files are loaded on all nodes in parallel

Process: Figure 69 on page 161 shows the JCL for running the unload and db2split, and the transfers using CLIO/S CLPLINK. The first job step unloads the data using DSNTIAUL. The second job runs db2split and creates a pipe for each node.

```
//UNLOAD  EXEC TIAUL
//SYSIN    DD  DSN=DB2TOPE.UNLOAD(CUST)
//SYSREC00 DD  DSN=DB2MVS.CUSTOMER,DISP=(,CATLG),
//             DCB=(RECFM=FB,LRECL=249,BLKSIZE=27888),
//             UNIT=SYSDA,SPACE=(CYL,(358,0)),
//             VOL=SER=DBPE01
//*-------------------------------------------------------------------
//SPLIT    EXEC PGM=DB2SPLIT
//CONFIG   DD DSN=DB2MVS.CONFIG(ORDERS),DISP=SHR
//MYINPUT  DD DSN=DB2MVS.ORDERS,DISP=SHR
//OUTMAP   DD SYSOUT=*
//DISTFILE DD SYSOUT=*
//LOG      DD SYSOUT=*
//NOD00001 DD DSN=DB2MVS.ORDERS.NOD00001
//         DCB=(RECFM=VB,LRECL=376,BUFNO=3),
//         SUBSYS=BP01
//...      ...     ...    Similar statements for other nodes
//
//*-------------------------------------------------------------------
//
//ENDYC8   JOB (999,POK),'CBIPO INSTALL',NOTIFY=&SYSUID,REGION=20M,
//             CLASS=P,MSGCLASS=T,MSGLEVEL=(1,1)
//FCFTP    EXEC CLLINKW,
//  NODE='sp2es01:sp2n08',PIPE='orders.node8'
//IN       DD DSN=db2mvs.orders.NOD00008,
//         DCB=(RECFM=VB,LRECL=376,BUFNO=3),
//         SUBSYS=BP01
//...      ...     ...    Similar jobs for other nodes
//
```

Figure 69. JCL for Variation 15

Note:

1. For more details on DB2SPLIT see 5.5.3, “Job to Execute DB2SPLIT” on page 97.
2. There has to be a job like ENDYC8 for each node.
3. The procedure CLLINKW is described in Figure 15 on page 47.
4. The NODE parameter shows how to specify the jump from the ESCON attached node 1 (SP2ES01) to the database node 8 (SP2N08).
5. You have to have an initiator and a CLIO/S link for each node.
6. The PIPE parameter creates a file on the relevant node table.

The transfer should not be a bottleneck as it should be able to keep up with the unload and db2split processes.

The load process is the same as 6.2.1.1, “Variation 1: All Files” on page 110.

Benefits

- Fast unload from MVS.
- The transfer can be automated as part of a batch job on MVS.
- Loads in parallel minimize outages on the DB2 Parallel Edition table.

- Transfer is done in parallel with db2split and unload.

Limitations

- The unload must either create an EBCDIC file or an all ASCII file.
- There are many small pipes open at the same time between MVS and AIX and the automation of this process must be carefully planned.

6.2.4.4 Variation 16: No Files

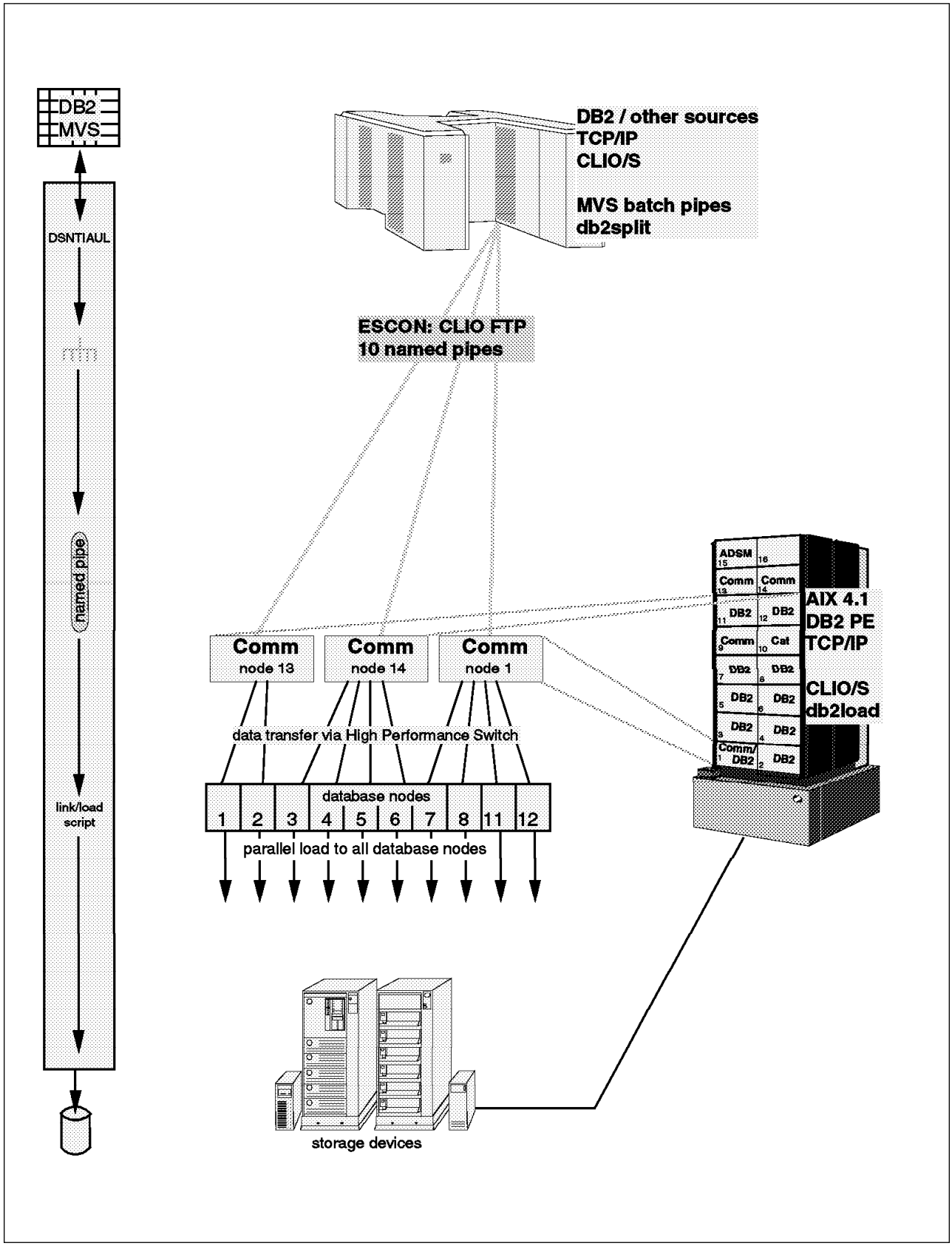


Figure 70. Variation 16 Overview

Summary: This is a possible solution for initial load, or table refresh where the refresh window is not critical.

The db2split is likely to be the bottleneck rather than the transfer. So this variation is operationally more complex than 6.2.2.4, "Variation 8: No Files on AIX" on page 133, which has one transfer and autoloader, without any real benefits.

It should be compared with:

- 6.2.2.4, "Variation 8: No Files on AIX" on page 133
- 6.2.3.4, "Variation 12: No Files" on page 148
- 6.2.4.4, "Variation 16: No Files" on page 163

This is a macho variation. It is fun to try and watch, but is not recommended for operational environments. If you need the minimum elapse time use, 6.2.2.4, "Variation 8: No Files on AIX" on page 133, which should provide the same performance but is operationally simpler.

Description

1. An unload process, such as DSNTIAUL, creates a sequential file on MVS in a form suitable for loading into DB2 Parallel Edition. This file is not written to disk but straight into a BatchPipe.
2. db2split is run, on MVS, and creates an output file per node, that is piped straight into CLPLINK.
3. CLPLINK pipes the data straight into load on the nodes.
4. The loads can be run on all nodes in parallel.

Process: The unload and split is identical to 6.2.4.3, "Variation 15: No Output from db2split" on page 159.

The load is identical to 6.2.3.4, "Variation 12: No Files" on page 148.

Benefits

- Fast unload from MVS.
- No temporary files on AIX thus reducing disk requirements.
- No temporary files on MVS thus reducing disk requirements.
- PIOFS not needed.
- CLIO/S transfers data from MVS faster than db2split can process it, so the db2split elapse will not be increased.
- Total elapse will be reduced because the file transfer is overlapped with split and load.

Limitations

- The unload must either create an EBCDIC file or an all ASCII file.
- The outage on the DB2 Parallel Edition table will be longer, because db2split takes longer to run than the load on an individual table
- The complexity of ensuring that all the load pipes and process on the nodes are up before the CLPLINK jobs makes this a complex operational scenario.

6.2.4.5 Variation 17: DataPropagator Relational

DataPropagator Relational is a set of easy-to-use, automated copy tools. These tools can be used to extract data from DB2 for MVS and put it on DB2 Parallel Edition

Figure 71 on page 166 and Figure 72 on page 168 show the functional DataPropagator Relational components needed to copy data from DB2 for MVS to DB2 Parallel Edition:

- DataPropagator Relational for OS/2 (Administration Control Point)

  This component is the graphical user interface running on DataHub for OS/2 to manage the replication solution. The administration control point must be connected to DB2 for MVS via DDCS and to DB2 Parallel Edition

  The DataHub Workstation is using TCP/IP communication protocol to connect to DDCS and to DB2 Parallel Edition

  DDCS uses APPC to get to DB2 for MVS via an SNA link.

  DB2 for OS/2 V2.1 is a prerequisite product of DataHub for OS/2.

- Capture for MVS

  This component enables update propagation, capturing the changes made to the data in the registered source tables by reading the database log and placing them on change data tables.

- Apply for AIX

  Apply reads the changed data previously captured and stored in a change data table and applies it to the target tables. Apply can also read data directly from source tables, for example in full refresh situation.

  Apply also tailors the data to your specifications when it copies the data to the targets.

  The apply component must be installed on one of the DB2 Parallel Edition nodes and that node must be connected as a DDCS client.

Each Relational Database Management System (RDBMS) involved in the replication solution using DataPropagator Relational acts as a server.

- DB2 for MVS is the data server

  This is the database where the source data is stored and where the Capture component runs.

- DB2 Parallel Edition is the copy server

  This is the database where the target copy of the data is stored and where the apply component executes.

- Control server is defined on DB2 Parallel Edition

  This is the database where the DataPropagator Relational subscription control tables are located. This server can be collocated at either the data or copy server.

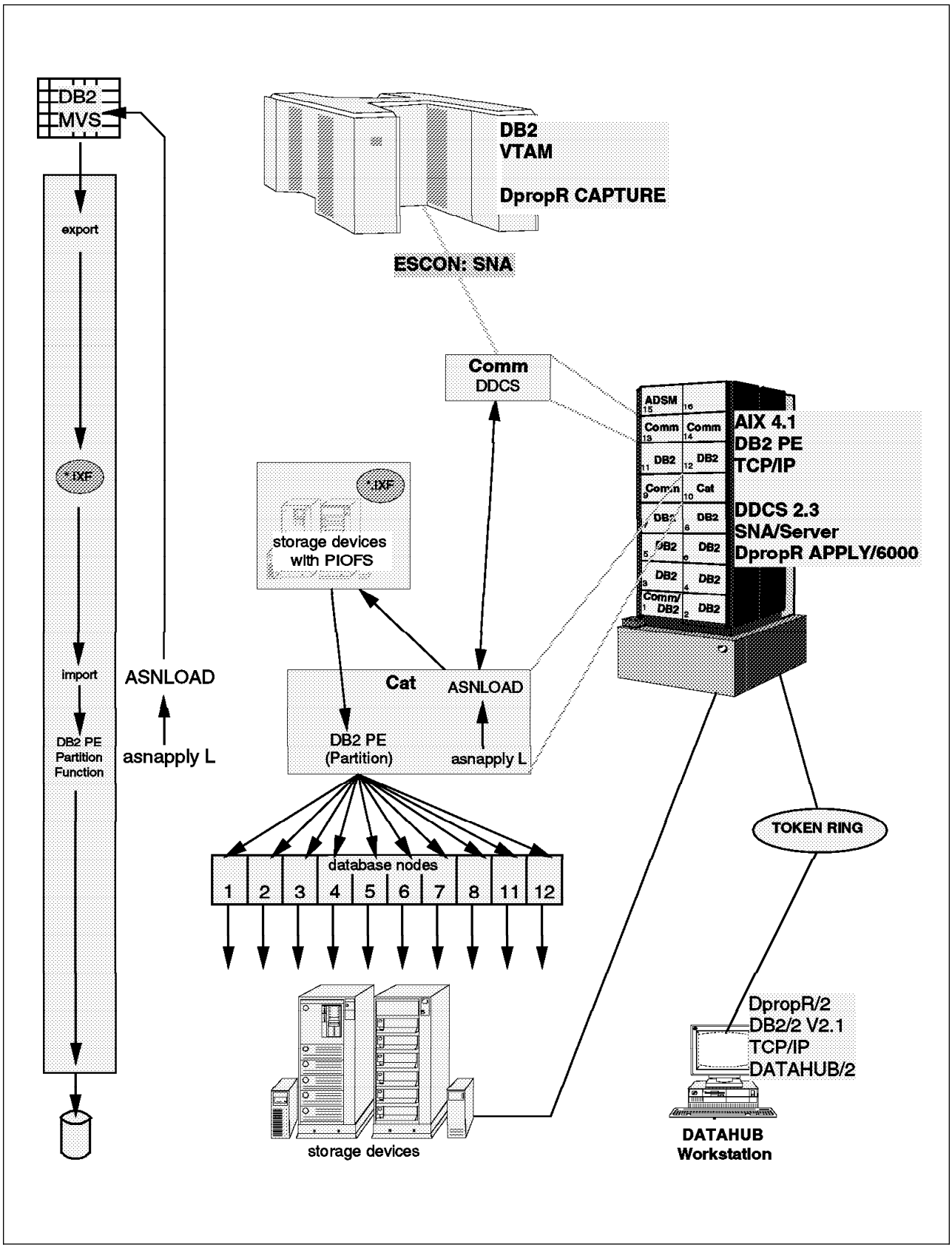


Figure 71. Apply Process for a Full Refresh Via ASNLOAD Exit Routine

The apply component is designed for automated operation. That is, it selects the copy mode, the most appropriate source between change data tables (all captured changes), consistent change data table (only committed changes) and the original source tables. It also automatically selects between refresh or update, and replace or append.

When apply is started, it can be instructed to use an exit routine called ASNLOAD whenever it has to do a full refresh operation (asnapply L).

The exit routine ASNLOAD is provided as an example program. It uses the EXPORT and IMPORT utilities of DB2 Parallel Edition as shown in Figure 71 on page 166. The EXPORT utility extracts data from MVS and creates a PC/IXF file that is later on read by the IMPORT utility to insert the data in the DB2 Parallel Edition target table.

Loading large copies using apply with the exit routine ASNLOAD as provided, EXPORT/IMPORT, is not very efficient. You can perform your own load using other methods, as described in this chapter, and notify the apply component that the initial full refresh has been done.

To apply changes on the target tables, apply gets the data from the appropriate source and uses inserts or updates statements on the target tables, See Figure 72 on page 168.

If apply is invoked without any invocation parameters, it uses inserts to do full refresh operations.

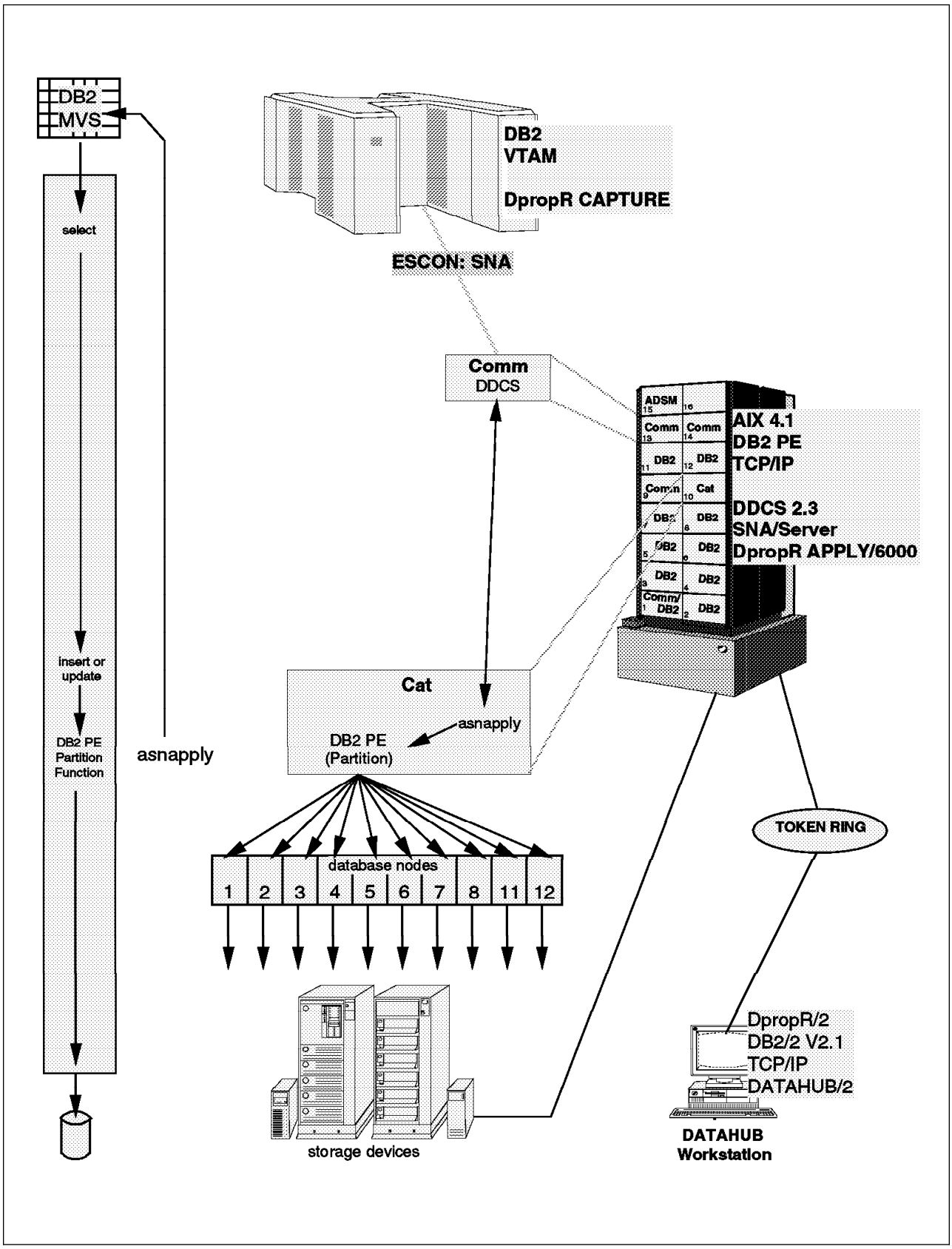


Figure 72. Apply Process to Apply Changes to the Target Table

Summary: DataPropagator relational can be used to create informational data in DB2 Parallel Edition from operational data residing on DB2 for MVS tables.

Description

1. The capture component monitors the DB2 for MVS log to detect change records from source tables that both have the DATA CAPTURE CHANGES attribute and are registered as refresh and update copies.
2. The apply component gets the data from most appropriate source and place it into the target tables.

   Before apply begins to apply changes to a point-in-time target table, it must first initialize the table with a full refresh.

   Apply will use the ASNLOAD program if it was started with the L parameter

   Point-in-time or complete condensed consistent change data (CCD) copies are updated with the information from the change data table, after the first initialization is completed.

   Aggregate and non-condensed copies are just appended from the captured changes or from the new calculations performed at the apply time in the source table.

Benefits

- Automatic data copying
- Automatic update of DB2 Parallel Edition tables with changes captured from tables in DB2 for MVS
- Data enhancement with full SQL support
- Data conversion are made during the copy process

Limitations

- It is not suitable for doing large copies since the export utility is slower than other loading processes.
- If the source columns that form the partitioning key of the target table are modified by the user transactions, then point-in-time or condensed copy tables should be created in a single node nodegroup.

6.3 Summary

All of the variations have their benefits and limitations, and there are no clear winners or losers.

The following questions should help to narrow down the options suitable for a specific environment.

- Is the file transfer going to be via tape? If so, then only variations 1, 2, 9 and 13 are relevant.
- Is CLIO/S being used? If so, file to file transfer is overkill so variations 1, 2, 9 and 13 are irrelevant.
- Do you need to minimize the outage on the DB2 Parallel Edition table? If so, then load files are needed and variations 1, 3, 5, 7, 9, 10, 11, 13 and 15 are relevant.

- Do you have sufficient CPU resource on the RISC/6000 SP for db2split? If so it is simpler to run db2split on RISC/6000 SP and variations 1-8 are preferred.
- Where do you have spare disk space for temporary files? Choose the relevant variations.

For example, if we have the following:

- CLIO/S is installed.
- Outage of the DB2 Parallel Edition table is not critical.
- MVS has spare disk capacity.

Then variations 4 and 8 are the most suitable.

Chapter 7. Implementing Data Propagation from DB2 for MVS to DB2 Parallel Edition

This chapter provides the steps to implement data propagation between DB2 for MVS and DB2 Parallel Edition using DataPropagator Relational components.

This is not a complete usage guide of the products and it only details the basic steps to install, set up and use of this data replication solution.

7.1 Introduction

DataPropagator Relational is a set of easy-to-use, automated copy tools. These tools can be used to extract data from DB2 for MVS and put it on DB2 Parallel Edition.

The following are the functional DataPropagator Relational components needed to copy data from DB2 for MVS to DB2 Parallel Edition:

- DataPropagator Relational for OS/2 (Administration Control Point)

  This component is the graphical user interface running on DataHub for OS/2 to manage the replication solution.

- Capture for MVS

  This component enables update propagation, capturing the changes made to the data in the registered source tables by reading the database log and placing them on change data tables.

- Apply for AIX

  Apply reads the changed data previously captured and stored in a change data table and applies it to the target tables. Apply can also read data directly from source tables, such as for a full refresh.

  Apply also tailors the data to your specifications when it copies the data to the targets.

Each Relational Database Management System (RDBMS) involved in a replication solution using DataPropagator Relational acts as a server. There are the following three types of servers:

1. Data server

   The database where the source data is stored and where the Capture component runs.

2. Copy server

   The database where the target copy of the data is stored and where the Apply component executes.

3. Control server

   The database where the DataPropagator Relational subscription control tables are located. This server can be co-located at either the data or copy server.

In our scenario:

DB2 for MVS is the data server.

DB2 Parallel Edition is the copy server.

The control server is co-located with the copy server (DB2 Parallel Edition).

DataPropagator Relational user roles include administrator, registrar, and subscriber.

Administrators

Database administrators will likely be the DataPropagator Relational administrators who perform the following tasks:

- Install and tune the DataPropagator Relational components
- Start or automate the operation of Capture products
- Create control tables
- Grant registrars access to control tables
- Work with subscribers to tune Capture and Apply interoperations

  Only one administrator can exist for each data server or copy server.

Registrars

Registrars obtain their authority from the DataPropagator Relational administrator.

Database administrators can also be registrars who perform the following tasks:

- Identify and register DataPropagator Relational source tables
- Create registration specifications
- Define copy sources outside of DataPropagator Relational, such as tables created by DataPropagator NonRelational, DataRefresher or other programs
- Grant subscribers authority to copy from registered tables
- Create and register SQL views for joins of source tables

Subscribers

Subscribers obtain their authority from DataPropagator Relational registrars.

LAN administrators, application database administrators, or others with SQL knowledge can be subscribers, who perform the following tasks:

- Create and maintain copy subscriptions on systems under their authority
- Select source tables or views from previously registered tables or views
- Select copy frequency
- Provide SQL to enhance data as it is copied
- Run Apply or set up automation of Apply
- Work with the administrator to tune the Capture and Apply interoperations

Figure 73 on page 173 shows the complete scenario covered in this section. It describes overview of all the products, components and network links used to set up the scenario for copying data from DB2 for MVS to DB2 Parallel Edition.

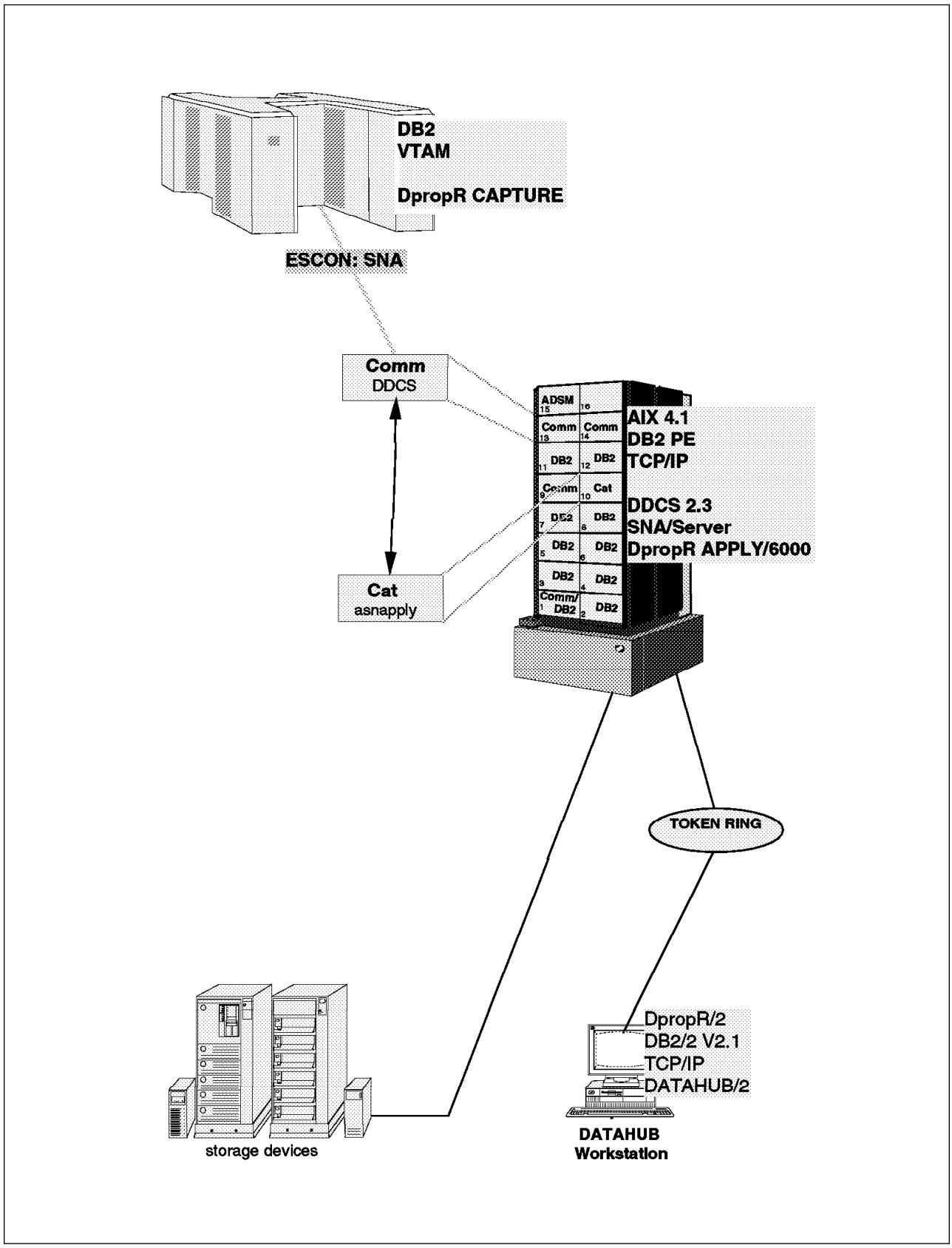


Figure 73. Overview of the System Environment

As you can see in the Figure 73:

The OS/2 machine is on a token-ring Local Area Network (LAN) with the RISC/6000 SP machine. One of the RISC/6000 SP nodes is attached via an ESCON Channel to the MVS system.

The Capture component is installed on DB2 for MVS V4 to collect the changes of the registered tables.

The Apply component is installed on one of the DB2 Parallel Edition nodes and it goes through DDCS to extract the data from DB2 for MVS tables.

The administration control point gets to DB2 for MVS via DDCS and to DB2 Parallel Edition using TCP/IP.

To extract data from a relational database management system and copy it into another, the proper network communications should be established. Also, the basic configuration for the communications products, such as TCP/IP and SNA, should be done before installing the replication solution.

7.1.1 Setting Up DataHub for OS/2

The following list provides the sequence of installation and configuration tasks that need to be done as part of the setting of DataHub for OS/2 product. It only presents the tasks needed to support the propagation between DB2 for MVS and DB2 Parallel Edition.

1. Enable the database support by installing DB2 for OS/2 Version 2.1. For details on doing this, see the *DB2 for OS/2 Installation and Operation Guide*, S62G-3664.
2. Install the DataHub database and the DataHub workstation. Follow these steps:
   a. Make sure DB2 for OS/2 is started and that you are logged on with an administrator user ID with authorization to create a database.
   b. Insert Diskette 1 in the drive and from an OS/2 window type `a:` and then type `install`.
   c. Check the Update CONFIG.SYS box and click on the **OK** button.
   d. From the list of components to install, select **DataHub database** and **DataHub Workstation** components.
   e. Enter the name of the directory where you want DataHub files to be stored.
   f. Click on the **Install...** button to begin the installation.

      During DataHub database install, you are prompted to answer a short series of questions as follows:

      1) `Create a new database? [y/n]`

         Type `y` and press Enter to create a new database.

         Having a separate database for DataHub isolates it from other user tables.

      2) `Enter the name of a DataHub database (suggested value EMQDB)`

         If you are creating a new database EMQDB is the default name, but the name can be any 8-byte name you choose as long as it is unique.

      3) If you are creating a new database you are prompted to:

         `Enter the drive letter where DataHub database resides`

The drive letter is the hard disk or partition where you want to store the DataHub database.

g. Check the screen for a return code of 0.

h. Reboot the machine.

3. Connect the DataHub workstation to the managed Relational Database Management Systems.

   - DB2 Parallel Edition

     Figure 74 on page 176 shows the relationships that must be established to connect the DataHub workstation as a database client of DB2 Parallel Edition.

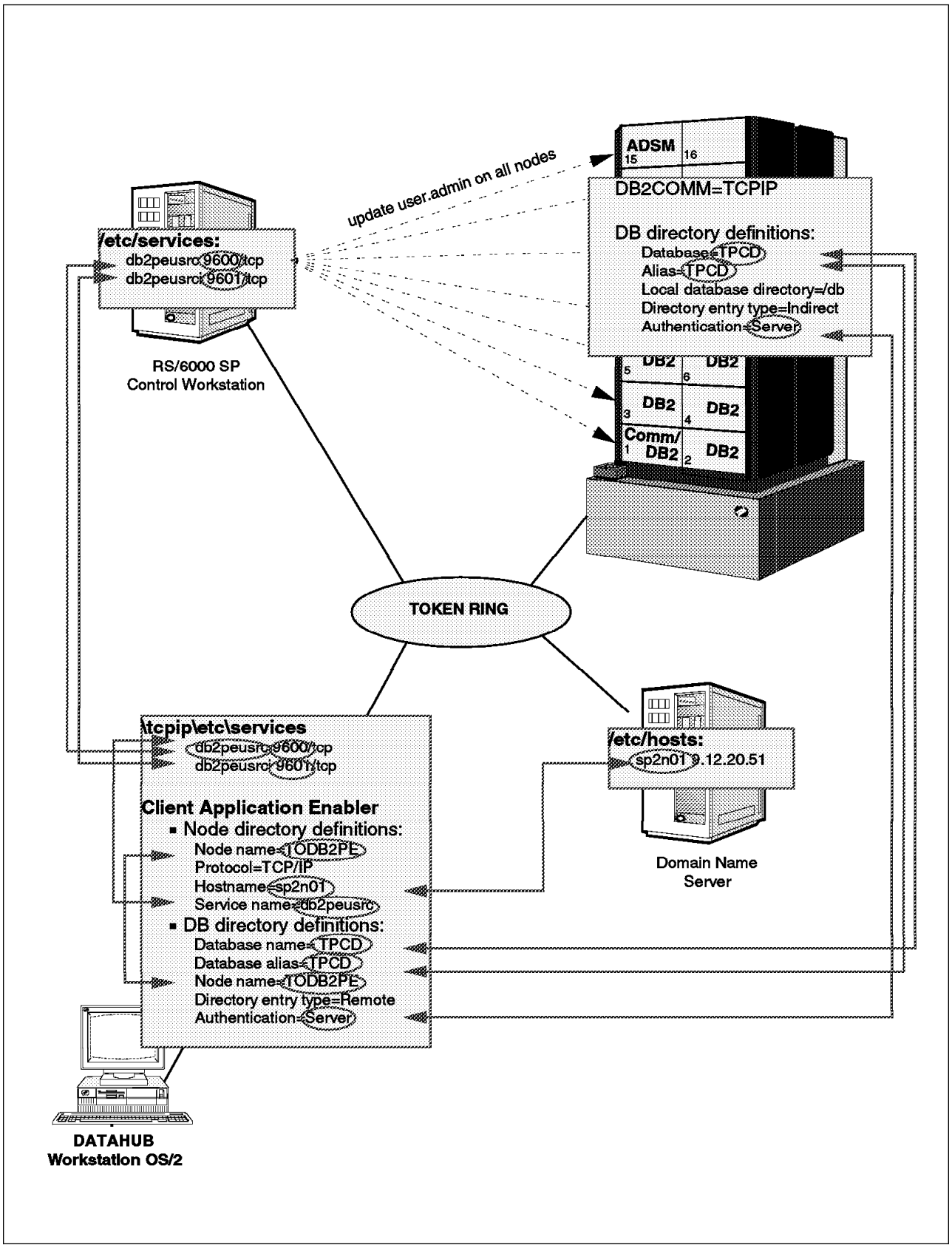


Figure 74. DataHub Workstation As a DB2 Parallel Edition Client

To do this setup you have to:

a. Establish DB2 Parallel Edition as a server for remote clients:

   1) Log on as root on the Control Workstation.

   2) Modify the /etc/services file to include two port numbers to establish DB2 Parallel Edition as a server.

      Add the following two lines:

      ```
      db2peusrc   9600/tcp
      db2peusrci  9601/tcp
      ```

   3) Propagate the /etc/services file to all RISC/6000 SP nodes.

      ```
      # dsh -a /var/sysman/supper upgrade user.admin
      ```

   4) Log on to one of the DB2 Parallel Edition nodes as the instance owner.

   5) Modify the database manager configuration file using the service name previously defined on the /etc/services file in step 3a2.

      ```
      db2 update database manager configuration using svcename db2peusrc
      ```

   6) Stop and start DB2.

b. Configure the DataHub workstation as a client of DB2 Parallel Edition.

   For this purpose DB2 Client Application Enabler for OS/2 is used. Follow these steps:

   1) Update or create a services file in the etc directory where TCP/IP was installed, for example C:\TCPIP\ETC, and add two port names with their corresponding port numbers:

      ```
      db2peusrc  9600/tcp
      db2peusrci 9601/tcp
      ```

      The port numbers must match those defined for the DB2 Parallel Edition server on step 3a2.

   2) Click on the **IBM Database 2** icon.

   3) Click on the **Client Setup** icon. If the assistance Client Setup screen appears, click on the **Cancel** button.

   4) Create a new node as shown in Figure 75 on page 178.

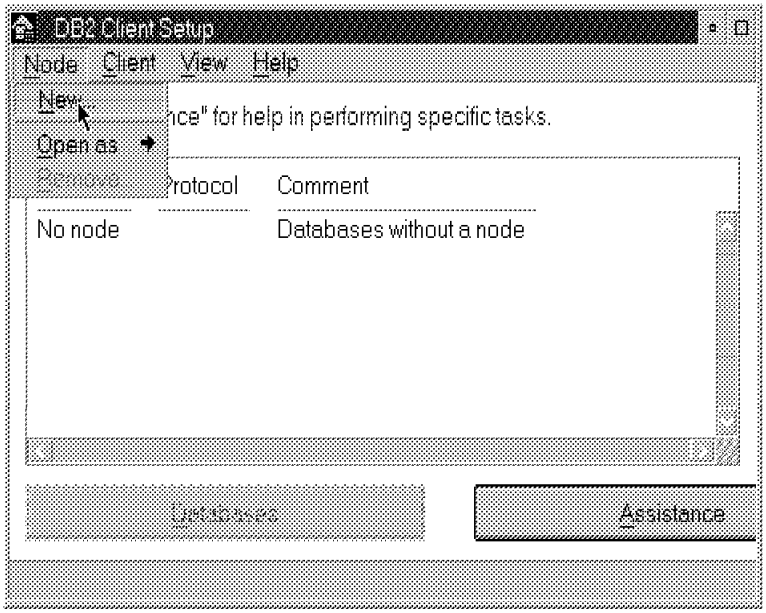


Figure 75. Creating a New Node

5) Catalog a node using the host name of one of the DB2 Parallel Edition nodes and the service name for the first port you have used in step 3b1 on page 177. This is shown in Figure 76.

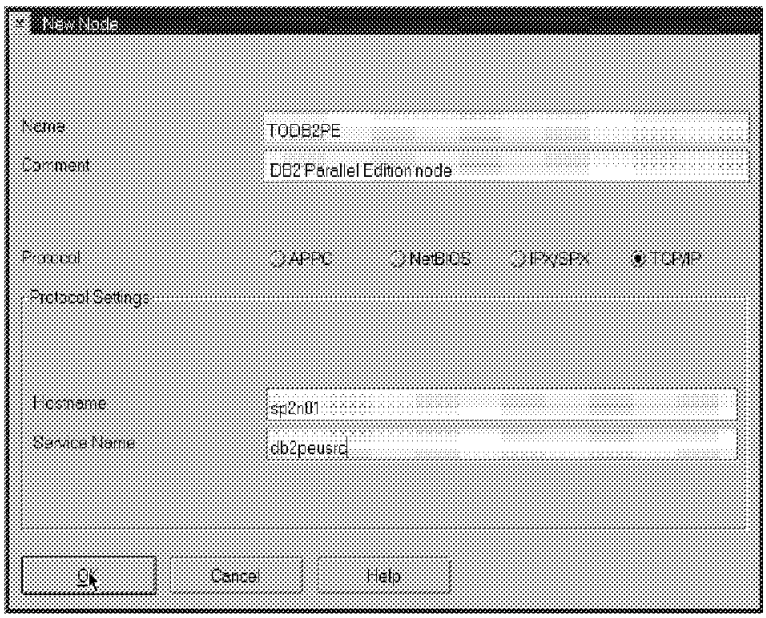


Figure 76. Catalog a Node

6) Select the node created and click on the **Databases** button as shown in Figure 77 on page 179.

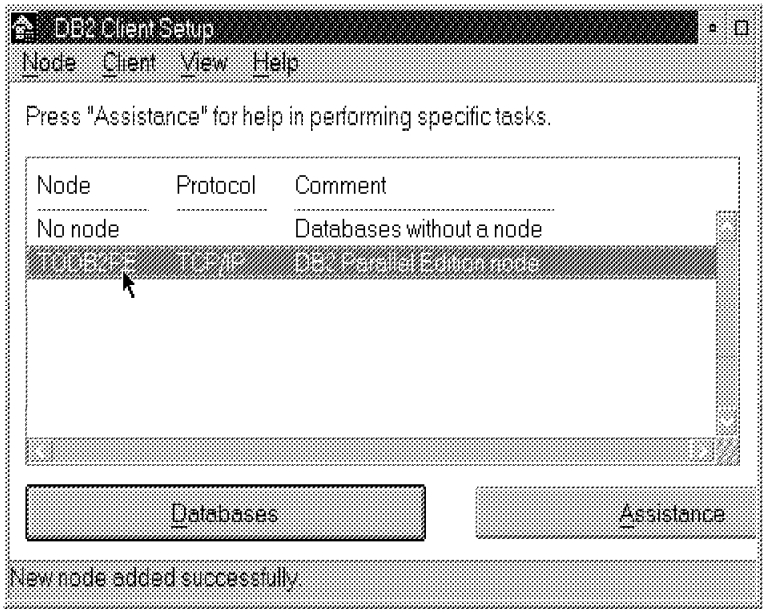


Figure 77. DB2 Client Setup Added Successfully

7) Create an entry for the DB2 Parallel Edition database. Select **New** on the database menu as shown in Figure 78.

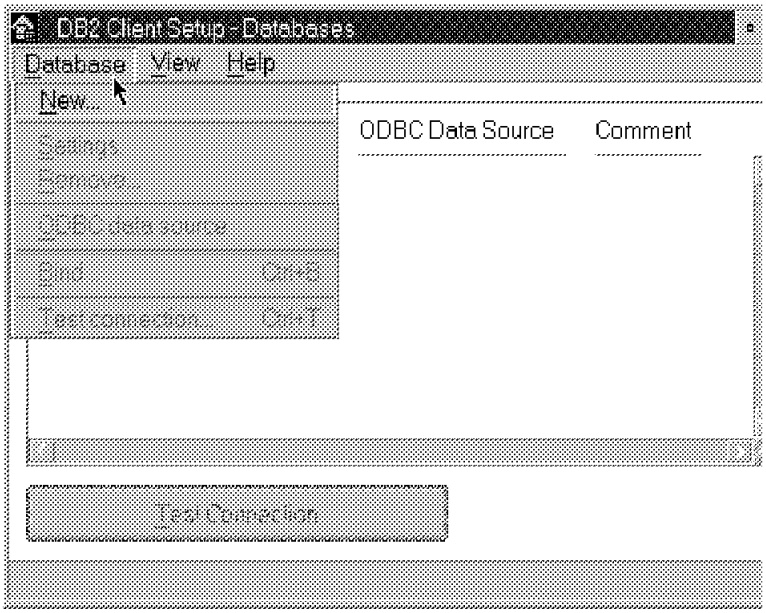


Figure 78. DB2 Client Database Setup

8) Catalog the DB2 Parallel Edition database as shown in Figure 79 on page 180 and fill in the values as follows:
   - The name must match the alias value used on DB2 Parallel Edition.
   - The alias for DataPropagator Relational must be the same for all systems involved in the replication process.
   - The at node value is already set. Do not change it.

Test the connection (click on the **Test Connection** button)

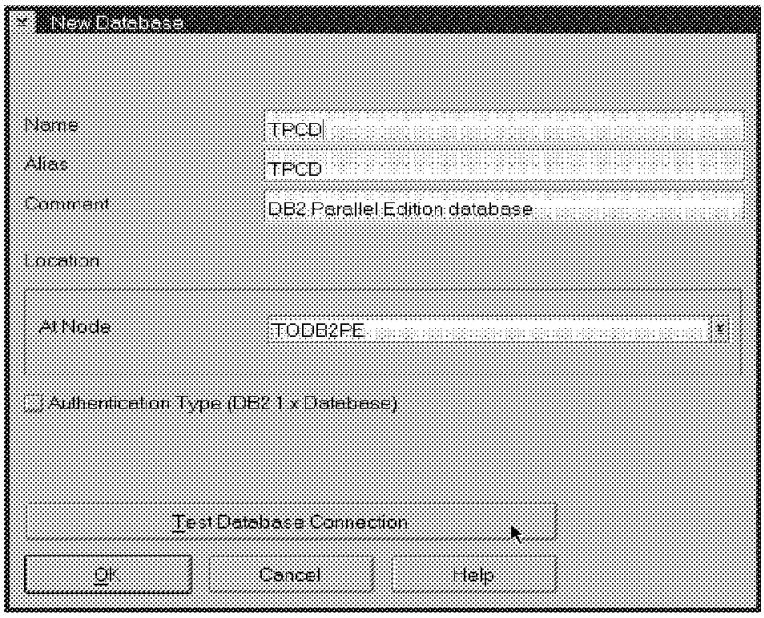


Figure 79. New Database

9) Provide a valid user ID and password for connecting to the DB2 Parallel Edition database and click on **OK** as shown in Figure 80.

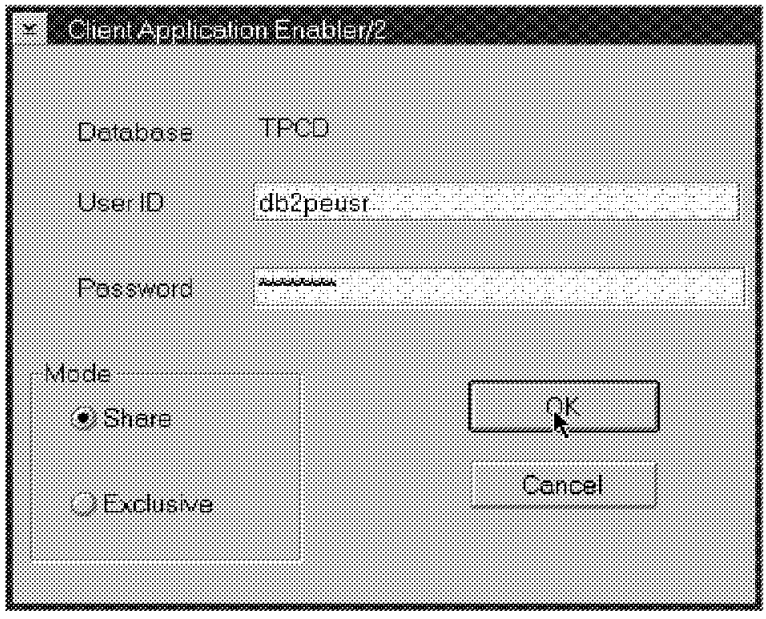


Figure 80. Client Application Enabler/2

If the connection is successful, you get a message stating the database connection was validated successfully.

Click on **OK** and at the bottom of the DB2 Client Setup-Databases screen you will see a message that says: `New database added successfully`.

- DB2 for MVS

  Connecting the DataHub workstation to DB2 for MVS is a little more complex than connecting to DB2 Parallel Edition.

  This connection requires installing and configuring the Distributed Database Connection Services product (DDCS). DDCS enables an OS/2 application, such as DataHub, to access DB2 for MVS databases. This product can be installed on OS/2 or AIX. In our scenario we chose the RISC/6000 SP node to act as the gateway server for the DataHub workstation.

  a. DDCS for AIX installation:

     1) Log on to the control workstation and create the administration group for DDCS.

     2) Create a 9076 user that will be the DDCS instance owner.

     3) Assign a password to the DDCS instance owner user.

     4) Propagate the changes to all RISC/6000 SP nodes as shown below

        ```
        # dsh -a /var/sysman/supper upgrade user.admin
        ```

     5) Log on to the node where DDCS is installed as root.

     6) Install DDCS using the smit or the installp command.

     7) Create the DDCS instance:

        ```
        # cd /usr/lpp/db2_02_01/instance
        # ./db2icrt ddcsusr
        ```

        Where ddcsusr is the user ID created to be the DDCS instance owner.

     8) Log on as the DDCS user and modify the .profile file to include the following line:

        ```
        . $HOME/sqllib/db2profile
        ```

     9) Execute the .profile file

     For more details of the installation process, refer to the *Distributed Database Connection Services for AIX Installation and Configuration Guide*, S20H-4794.

  b. DDCS configuration of DB2 for MVS databases

     Figure 81 on page 182 shows relationship between the definitions on DB2 for MVS, SNA Server for AIX and the DDCS directories that have to be done in order to connect DDCS to DB2 for MVS.

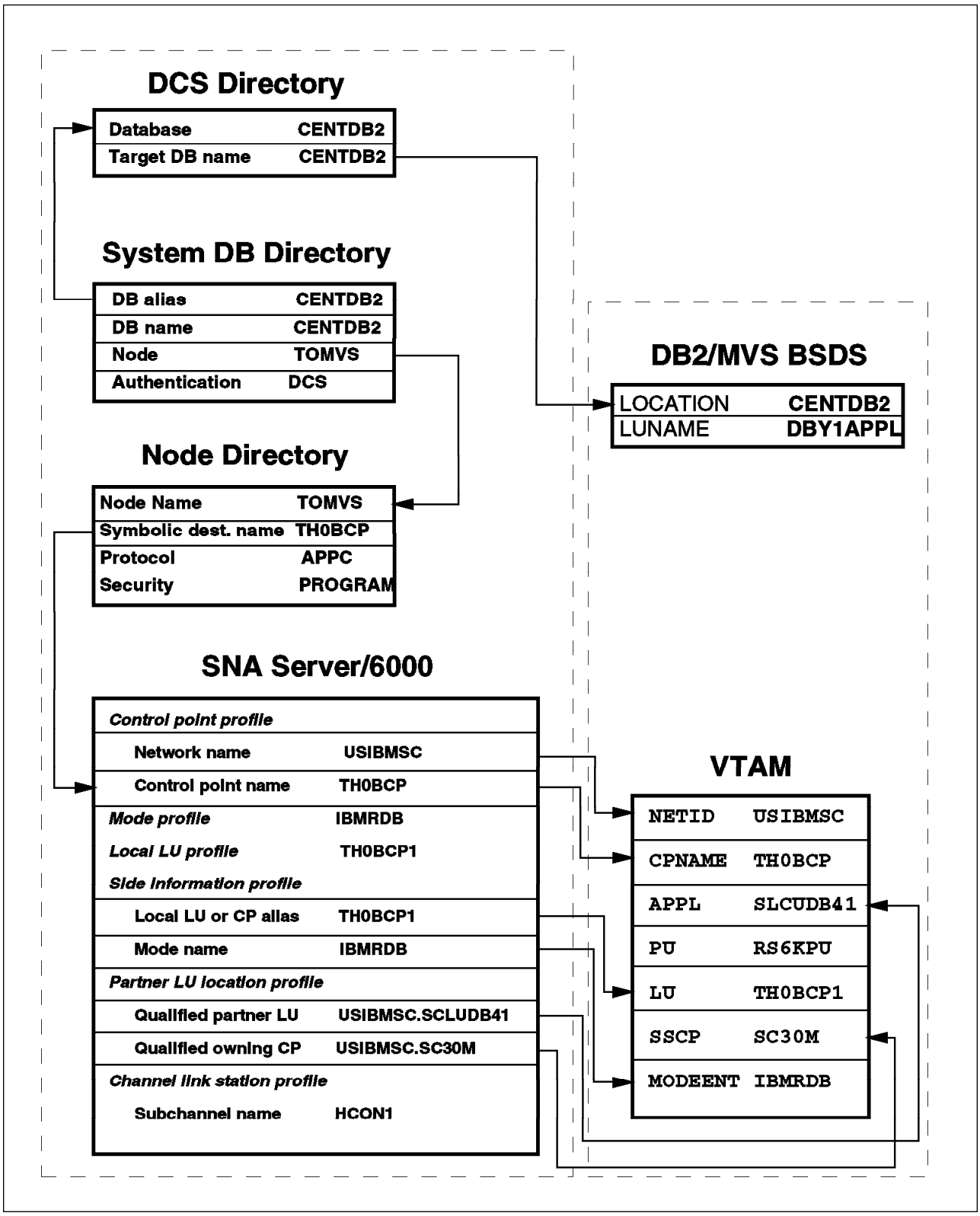


Figure 81. Configuration of DDCS to DB2 for MVS

The detailed steps are:

1) As root, create the SNA profiles to have LU6.2 sessions with the MVS system. See Appendix B, “SNA Profiles” on page 253 for the SNA profiles used in our scenario.

2) Start SNA and the link station defined for MVS.

3) Log on as the DDCS instance owner.

4) Catalog an APPC node using the LU 6.2 Side Information Profile name and security program.

```
db2 catalog appc node TOMVS remote TH0BCP1 security program
```

5) Catalog the DB2 for MVS database at the node you created in the previous step.

```
db2 "catalog database CENTDB2 as CENTDB2 at node TOMVS
authentication DCS"
```

6) Catalog the DCS database.

The target database name, the value following the *as* keyword of the catalog dcs command, must match the LOCATION parameter defined on the DB2 for MVS BootStrap Data Set (BSDS).

The database name must match the one defined as the database name on the catalog database command entered in the previous step.

```
db2 catalog dcs database CENTDB2 as CENTDB2
```

c. DDCS configuration to support remote clients

Follow these steps:

1) On the Control Workstation as root user, edit the /etc/services file and add two port numbers:

```
ddcs_cs         9510/tcp  # DB2 DDCS Client Support
ddcs_int        9511/tcp  # DB2 DDCS Client Support Interruption port
```

2) Propagate the /etc/services file to all RISC/6000 SP nodes by issuing the following command as root:

```
dsh -a /var/sysman/supper update user.admin
```

3) Log on to the node where DDCS is installed as the DDCS instance owner. The DDCS instance owner in our example environment is ddcsusr, and DDCS was installed on node nine (sp2n09).

```
tn sp2n09
login: ddcsusr
ddcsusr's Password
```

4) Edit the $HOME/sqllib/db2profile file to establish the communications environment variable for TCPIP communications.

```
# Set the DB2COMM environment variable.
#------------------------------------------------------------------------
DB2COMM=TCPIP
export DB2COMM
```

5) Execute the .profile file.

6) Update the database manager configuration with the service name defined in step 3c1.

```
db2 update dbm configuration using svcename ddcs_cs
```

7) Start and stop DDCS using the db2stop and db2start commands.

d. DataHub workstation configuration as a DDCS remote client

1) Assign two port numbers for the DDCS communication in the c:\tcpip\etc\services file. These numbers have to match the port numbers used on the DDCS server for remote clients. See step 3c1 on page 183.

```
ddcs_cs         9510/tcp  # DB2/CAE port to get to DDCS server
ddcs_int        9511/tcp  # DB2/CAE interruption port for DDCS Server
```

2) Click on the **IBM Database 2** icon.

3) Click on the **Client Setup** icon and the screen in Figure 82 will appear.

4) On the Node menu select **New** and the screen in Figure 83 on page 185 will appear.

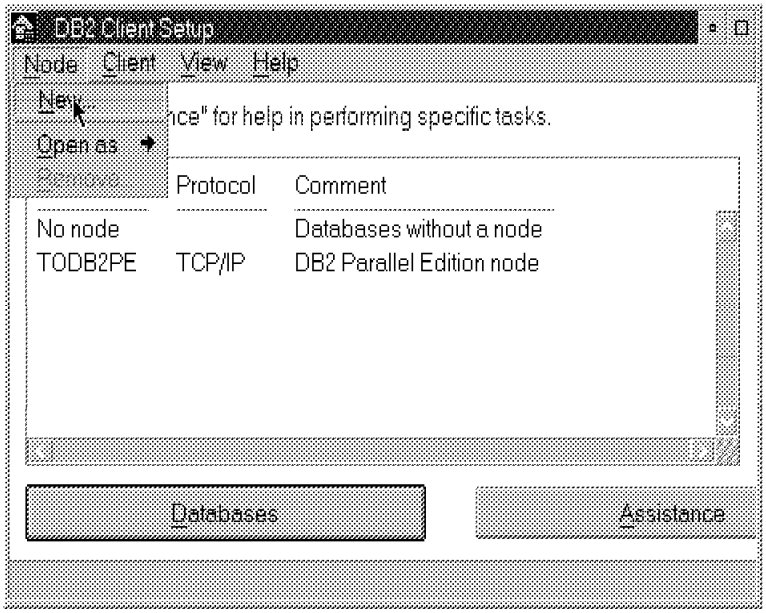


Figure 82. Creating a New Node

5) Give a name to the node and a comment. Select TCPIP Protocol.

The host name parameter must match the host name where DDCS is installed (in our system this corresponds to sp2n09). The service name is the one you have defined in step 3d1.

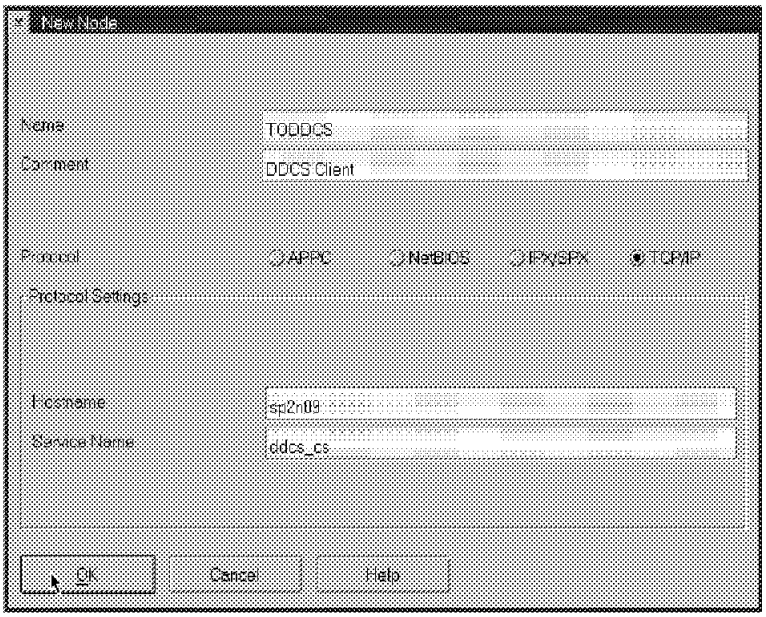


Figure 83. Display of New Node Screen

6) Select the node you have just created and click on **Databases** to define the DDCS database as shown in Figure 84.

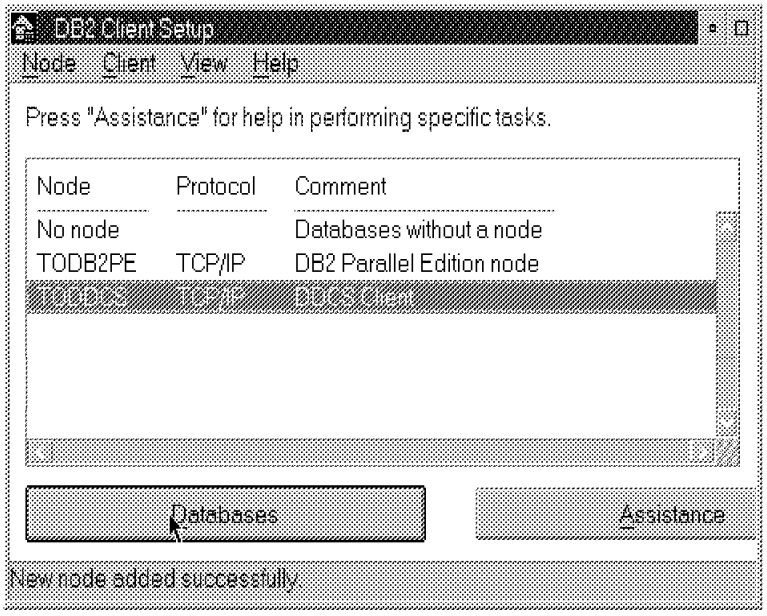


Figure 84. DB2 Client Setup

7) Select **New** from the database menu as shown in Figure 85 on page 186.

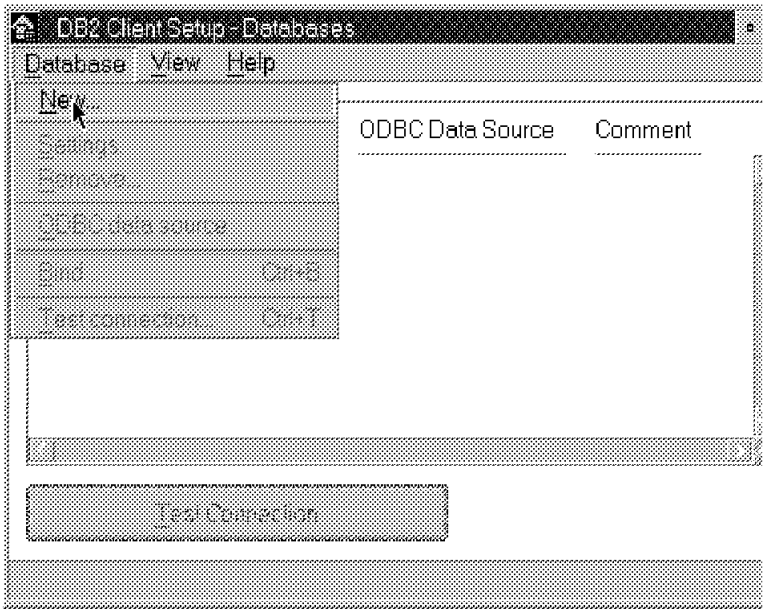


Figure 85. DB2 Client Database Setup

8) Provide the DB2 for MVS database name, an alias and a description.

   The Name entry must match the alias used to catalog the DB2 for MVS database on DDCS. See step 3b5 on page 183.

   For replication purposes, the Alias entry must match the alias used to catalog the DB2 for MVS database on DDCS. See step 3b5 on page 183.

   Try to connect; click on **Test Connection** as shown in Figure 86.

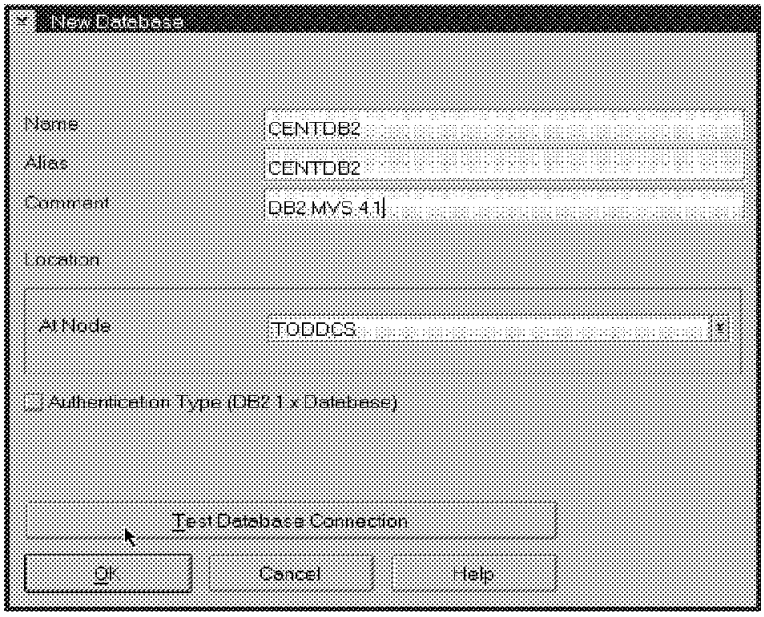


Figure 86. New Database

The connection process will prompt you for a user ID and password. Use a valid MVS user ID. This is shown in Figure 87 on page 187.

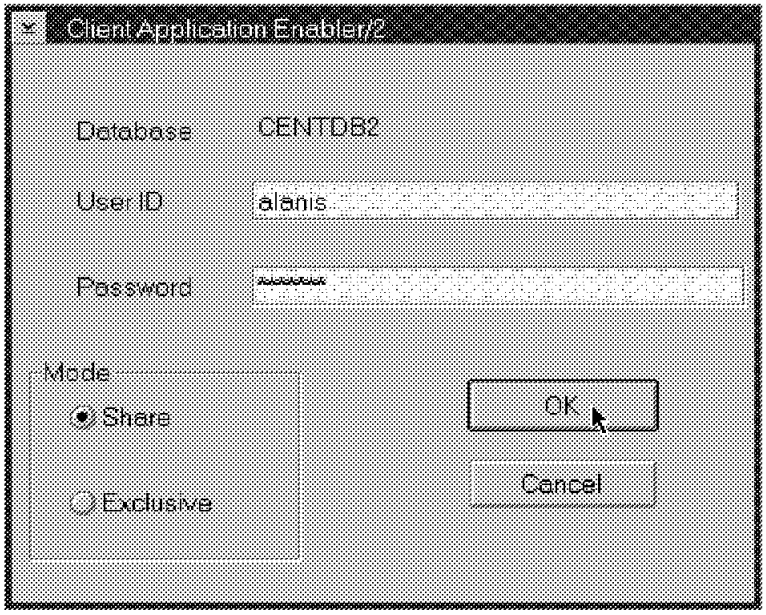


Figure 87. Client Application Enabler/2

If the connection is successful you will get a message saying: `Database connection validated successfully`.

Click on the **OK** button and at the bottom of the DB2 Client Setup-Databases screen you will see a message that says: `New database added successfully`.

Figure 88 on page 188 summarizes how the values used on DDCS must match the values on the DataHub workstation to set up client/server communication between them.

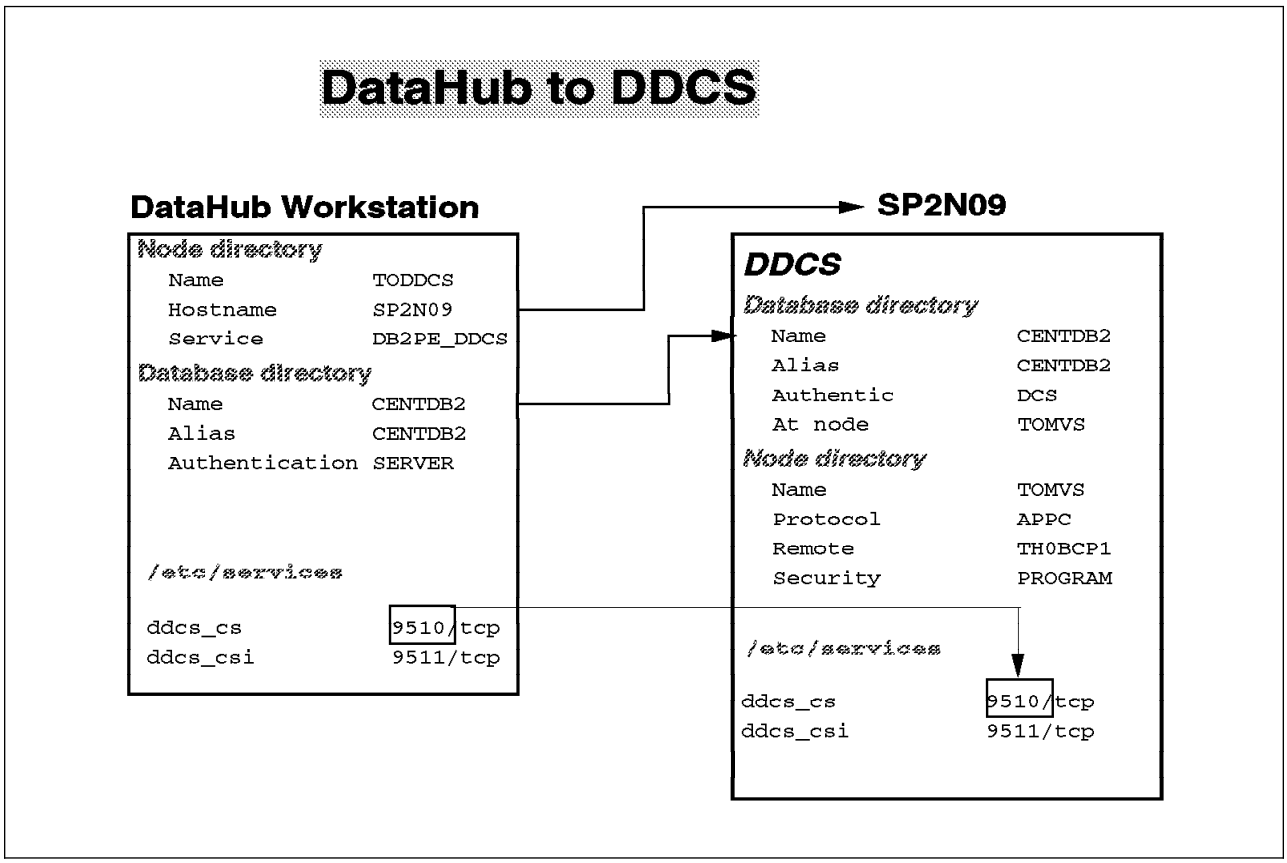


Figure 88. DDCS As a Server and DataHub Workstation As a Client

4. Register the resources on the DataHub Database.

   If you are not using the DataHub Support components the registering process of the resources should be done manually.

   In our scenario we are just using DataHub as prerequisite product for DataPropagator Relational, so registration of Hosts and Relational Database Systems (RDBs) must be done using the DHHOST and DHRDB commands.

   The DHHOST ADD command registers a managed host in the DataHub database so that the host name is displayed on the DataHub window.

   The DHRDB ADD command registers a managed RDB in the DataHub database so that the name of the RDB is displayed in the DataHub window.

   a. Register the MVS host in the DataHub Database. From an OS/2 window in the directory where DataHub for OS/2 was installed issue:

      ```
      C:\EMQDIR> DHHOST ADD mvs mvs
      ```

      The first parameter following the add keyword indicates the host name as you want it to appear on the DataHub window. The second parameter is the operating system type.

   b. Register the RISC/6000 SP host in the DataHub Database. From an OS/2 window issue:

```
C:\EMQDIR> DHHOST ADD sp2 aix
```

The first parameter following the add keyword indicates the host name as you want it to appear on the DataHub window. The second parameter is the operating system type.

c. Register the RDB MVS in the DataHub Database:

```
C:\EMQDIR> DHRDB ADD centdb2 mvs /Sdb41
```

The first parameter following the add keyword corresponds to the alias that is used to connect to DB2 for MVS. The second parameter is the host name you defined for the operating system. The /S option only applies for DB2 for MVS to specify the subsystem ID.

d. Register the RDB RISC/6000 SP in the DataHub Database:

```
C:\EMQDIR> DHRDB ADD tpcd sp2
```

The first parameter following the add keyword corresponds to the alias that is used to connect to DB2 Parallel Edition. The second parameter is the host name you defined for the operating system.

7.1.2 Setting Up DataPropagator Relational for OS/2

1. Installation.

   a. Insert the DataPropagator Relational for OS/2 installation diskette/CDROM into the appropriate drive and from an OS/2 window type:

   ```
   <path>:\install
   ```

   where <path> is the drive and directory identifier.

   b. Press Enter.

   c. In the installation window click on **Continue** and respond to the prompts.

   d. Shut down and reboot the system.

   In order to access the DataPropagator Relational objects and actions from the DataHub for OS/2 window, you must create at least one subscriber profile.

2. Select a copy server user ID to be the subscriber user ID.

   The user ID selected will be running the Apply program, must have access privileges on the source tables of the data server and must be able to create tables in the copy and control servers.

3. Create a local user ID in the DataHub workstation for the subscriber user ID selected.

4. Create the DataPropagator Relational profile as shown in Figure 89 on page 190.

The ASNPROF program creates on the designated DataHub for OS/2 database the subscriber profile table, and it also updates it with the new DataPropagator Relational objects and actions.

a. Log on locally with the subscriber user ID.

b. From an OS/2 Window, type:

```
asnprof create
```

The DataPropagator Relational for OS/2 Profile Installation Status window appears.

The DataPropagator Relational for OS/2 Profile Parameter panel appears. Fill in the entry fields as follows:

- Provide the Apply user ID of the user who will run the Apply program.
- In the copy server database name entry put the DB2 for OS/2 alias for the copy server.
- Choose the option control server located with copy server.

Note: The input to these fields is case sensitive.

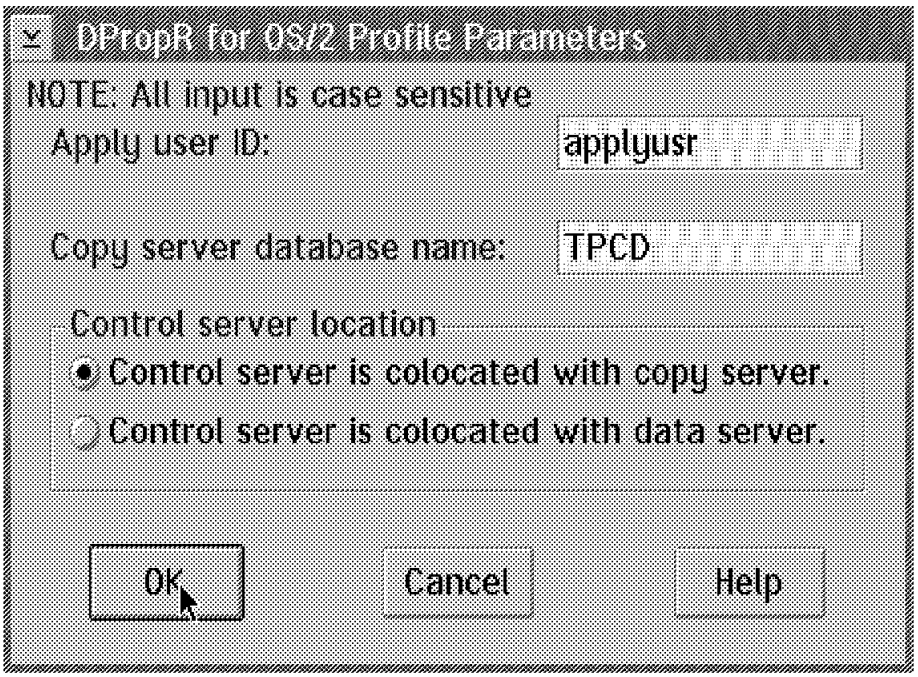


Figure 89. DPropR Profile Menu

The asnprof program creates a trace file (asninst.trc) under the directory where it is executed, and it logs all the messages displayed in the DataPropagator Relational profile installation status window. Figure 90 on page 191 shows the display of the trace file.

```
Installing the DataPropagator Relational Profile.
The Apply program user ID is "applyusr".
The copy server database is "TPCD".
The control server is co-located with copy server.
Reading the EMQCNFG.INI file.
The DataHub default database is "EMQDB".
Connecting to the DataHub default database "EMQDB".
Creating a DataPropagator Relational profile for the user ID "APPLYUSR ".
Reading the EMQCNFG.INI file.
Reading the EMQCNFG.INI file.
Inserting the product information into the DataHub database.
Installation is now complete.
Waiting for child to end
Ending PM thread...
```

Figure 90. Trace File Data

5. Select a user ID for the DataPropagator Relational administrator.

   This user ID must exist in each data server and copy server with the same password.

   This user ID must have privileges for creating and granting access to control tables in each data and copy server.

   This user ID cannot be the same as a DB2 for MVS installation SYSADM user ID.

6. Create the selected administrator user ID as a local user ID in the DataHub workstation with the same password.

7. Create the global control tables.

   You must create a set of global control tables at each data server and copy server using the DataPropagator Relational administrator user ID.

   a. Log on locally as the DataPropagator Relational administrator user ID, DPROPADM in our example.

   b. Edit the ASNCCP.SQL file to customize the SQL sentences according to the DDL syntax of the data server with the Capture component installed, DB2 for MVS in our scenario.

```
-- IBM DataPropagator Relational/2 Version 1 Release 2
-- ------------------------------------------------------------------
--
-- ASNCCP.SQL - An SQL file to create DataPropagator Relational global control
--              tables required on data server(s) where the DataPropagator
--              Relational Capture program will run.
--
-- Create the DPROPR DB2 database and tablespace
CREATE DATABASE DPROPR STOGROUP SYSDEFLT;
CREATE TABLESPACE DPROPTS
 IN DPROPR SEGSIZE 4 LOCKSIZE TABLE CLOSE NO;

-- Check whether there's a previous install or not by creating a view
-- with a name equal to ASN.IBMSNAP_CD_CNTL table.
CREATE VIEW ASN.IBMSNAP_CD_CNTL( VCOUNT, VUSER ) AS
   SELECT COUNT(*), USER FROM SYSIBM.SYSTABLES;

-- then attempt to DROP VIEW - will fail if table already exists.
DROP VIEW ASN.IBMSNAP_CD_CNTL;

-- Create the ASN.IBMSNAP_CD_CNTL table
CREATE TABLE ASN.IBMSNAP_CD_CNTL (
   BASE_OWNER         CHAR(18) NOT NULL,
   BASE_TABLE         CHAR(18) NOT NULL,
   CRIT_SECTION_OWNER CHAR(18) NOT NULL,
   CRIT_SECTION_TABLE CHAR(18) NOT NULL,
   CD_OWNER           CHAR(18),
   CD_TABLE           CHAR(18),
   PRUNE_CNTL_OWNER   CHAR(18) NOT NULL,
   PRUNE_CNTL_TABLE   CHAR(18) NOT NULL,
   CD_OLDEST_CHANGE   CHAR(10) FOR BIT DATA,
   FULLREFRESHENABLE  SMALLINT NOT NULL,
   CCD_OWNER          CHAR(18),
   CCD_TABLE          CHAR(18),
   CCD_OLDEST_CHANGE  CHAR(10) FOR BIT DATA,
   CD_CONDENSED       CHAR(1),
   CD_CONSISTENT      CHAR(1),
   CD_COMPLETE        CHAR(1),
   CCD_CONDENSED      CHAR(1),
   CCD_CONSISTENT     CHAR(1),
   CCD_COMPLETE       CHAR(1),
   BASE_STRUCTURE     SMALLINT NOT NULL,
   BASE_CONDENSED     CHAR(1)  NOT NULL,
   BASE_CONSISTENT    CHAR(1)  NOT NULL,
   BASE_COMPLETE      CHAR(1)  NOT NULL,
   REGISTRAR          CHAR(18) NOT NULL,
   ARCH_LEVEL         CHAR(4)  NOT NULL,
   DESCRIPTION        CHAR(254),
   PREFIX_CHAR        VARCHAR(4) )  IN DPROPR.DPROPTS;
```

Figure 91 (Part 1 of 3). SQL for the Creation of Control Tables on DB2 for MVS V4

```
-- Create an UNIQUE INDEX on ASN.IBMSNAP_CD_CNTL
CREATE TYPE 1 UNIQUE INDEX ASN.ZIBMSNAP_CD_CNTL
   ON ASN.IBMSNAP_CD_CNTL
   ( BASE_OWNER, BASE_TABLE )
    SUBPAGES 1;

-- Create the ASN.IBMSNAP_UOW table
CREATE TABLE ASN.IBMSNAP_UOW (
   IBMSNAP_UOWID     CHAR(10)  FOR BIT DATA NOT NULL,
   IBMSNAP_COMMITSEQ CHAR(10)  FOR BIT DATA NOT NULL,
   IBMSNAP_LOGMARKER TIMESTAMP NOT NULL,
   IBMSNAP_AUTHTKN   CHAR(12)  NOT NULL,
   IBMSNAP_AUTHID    CHAR(18)  NOT NULL )
IN DPROPR.DPROPTS;

-- Create an UNIQUE INDEX on ASN.IBMSNAP_UOW
CREATE TYPE 1 UNIQUE INDEX ASN.ZIBMSNAP_UOW
   ON ASN.IBMSNAP_UOW
   ( IBMSNAP_COMMITSEQ DESC )
    SUBPAGES 1
    CLOSE NO;

-- Create an INDEX on ASN.IBMSNAP_UOW
CREATE TYPE 1 INDEX ASN.ZIBMSNAP_UOW2
   ON ASN.IBMSNAP_UOW
   ( IBMSNAP_UOWID DESC )
    SUBPAGES 1
    CLOSE NO;

-- Create the ASN.IBMSNAP_CRITSEC table
CREATE TABLE ASN.IBMSNAP_CRITSEC (
   NUMBER_OF_LOGS SMALLINT )
IN DPROPR.DPROPTS;

-- Grant authority on ASN.IBMSNAP_CRITSEC to PUBLIC
GRANT SELECT, UPDATE, INSERT, DELETE
   ON TABLE ASN.IBMSNAP_CRITSEC TO PUBLIC;

-- Create the ASN.IBMSNAP_AUTH table
CREATE TABLE ASN.IBMSNAP_AUTH (
   GRANTOR        CHAR(18) NOT NULL,
   GRANTEE        CHAR(18) NOT NULL,
   GRANTOR_HIGH   CHAR(1)  NOT NULL,
   GRANTEE_LEVEL  CHAR(1)  NOT NULL,
   GRANTEE_STATUS CHAR(1)  NOT NULL )
IN DPROPR.DPROPTS;
```

Figure 91 (Part 2 of 3). SQL for the Creation of Control Tables on DB2 for MVS V4

```
-- Create an UNIQUE INDEX on ASN.IBMSNAP_AUTH
CREATE TYPE 1 UNIQUE INDEX ASN.ZIBMSNAP_AUTH
   ON ASN.IBMSNAP_AUTH
   ( GRANTOR, GRANTEE, GRANTEE_LEVEL )
   SUBPAGES 1;

-- Insert a row into ASN.IBMSNAP_AUTH
INSERT INTO ASN.IBMSNAP_AUTH
   ( GRANTOR, GRANTEE, GRANTOR_HIGH, GRANTEE_LEVEL, GRANTEE_STATUS )
   VALUES ( 'SYSIBM', USER, 'S', 'A', 'A' );

-- Create the ASN.IBMSNAP_TRACE table
CREATE TABLE ASN.IBMSNAP_TRACE (
  OPERATION    CHAR(8)      NOT NULL,
  TRACE_TIME   TIMESTAMP    NOT NULL,
  DESCRIPTION  VARCHAR(254) NOT NULL )
IN DPROPR.DPROPTS;

-- Create an INDEX on ASN.IBMSNAP_TRACE
CREATE TYPE 1 INDEX ASN.ZIBMSNAP_TRACE
   ON ASN.IBMSNAP_TRACE
   ( TRACE_TIME ASC )
   SUBPAGES 1
   CLOSE NO;

-- Create the ASN.IBMSNAP_CCPPARMS table
CREATE TABLE ASN.IBMSNAP_CCPPARMS (
  RETENTION_LIMIT    INT,
  LAG_LIMIT          INT,
  COMMIT_INTERVAL    INT )
IN DPROPR.DPROPTS;

-- Delete rows from ASN.IBMSNAP_CCPPARMS table
DELETE FROM ASN.IBMSNAP_CCPPARMS;

-- Insert a row into ASN.IBMSNAP_CCPPARMS table
INSERT INTO ASN.IBMSNAP_CCPPARMS
  ( RETENTION_LIMIT, LAG_LIMIT, COMMIT_INTERVAL )
  VALUES ( 10800, 10800, 30 );

-- Commit changes
COMMIT;
```

Figure 91 (Part 3 of 3). SQL for the Creation of Control Tables on DB2 for MVS V4

c. Run the ASNCNTL program to create the control tables as follows:

```
ASNCNTL /C centdb2
```

You will see the following messages shown in Figure 92 on page 195.

```
Installation of DataPropagator Relational control tables in progress...
CONNECT TO CENTDB2
Installation of control tables completed successfully!
```

Figure 92. Messages from DataPropagator Install

d. Terminate the connection to the database by typing `db2 terminate`.

e. Edit the ASNNOCCP.SQL file to customize the SQL sentences according to the DDL syntax of the copy server, DB2 Parallel Edition in our scenario.

   Include the creation of a single node group where the control tables will be created.

```
-- IBM DataPropagator Relational/2
-- Version 1 Release 2
-- ------------------------------------------------------------------
--
-- ASNNOCCP.SQL - An SQL script file to create DataPropagator Relational
--                global tables for databases without the DataPropagator
--                Relational Capture program installed.
-- '--' denotes start of comment for that line
-- ';'  denotes the end of an SQL statement
-- An SQL statement can span multiple lines
-- Blank lines are ignored
--
-- Create a single node nodegroup
CREATE NODEGROUP DPROPRNG ON NODE (3);

-- Create the ASN.IBMSNAP_CD_CNTL table
CREATE TABLE ASN.IBMSNAP_CD_CNTL (
   BASE_OWNER         CHAR(18) NOT NULL,
   BASE_TABLE         CHAR(18) NOT NULL,
   CRIT_SECTION_OWNER CHAR(18) NOT NULL,
   CRIT_SECTION_TABLE CHAR(18) NOT NULL,
   CD_OWNER           CHAR(18),
   CD_TABLE           CHAR(18),
   PRUNE_CNTL_OWNER   CHAR(18) NOT NULL,
   PRUNE_CNTL_TABLE   CHAR(18) NOT NULL,
   CD_OLDEST_CHANGE   CHAR(10) FOR BIT DATA,
   FULLREFRESHENABLE  SMALLINT NOT NULL,
   CCD_OWNER          CHAR(18),
   CCD_TABLE          CHAR(18),
   CCD_OLDEST_CHANGE  CHAR(10) FOR BIT DATA,
   CD_CONDENSED       CHAR(1),
   CD_CONSISTENT      CHAR(1),
   CD_COMPLETE        CHAR(1),
   CCD_CONDENSED      CHAR(1),
   CCD_CONSISTENT     CHAR(1),
   CCD_COMPLETE       CHAR(1),
   BASE_STRUCTURE     SMALLINT NOT NULL,
   BASE_CONDENSED     CHAR(1)  NOT NULL,
   BASE_CONSISTENT    CHAR(1)  NOT NULL,
   BASE_COMPLETE      CHAR(1)  NOT NULL,
   REGISTRAR          CHAR(18) NOT NULL,
   ARCH_LEVEL         CHAR(4)  NOT NULL,
   DESCRIPTION        CHAR(254),
   PREFIX_CHAR        VARCHAR(4) )
IN DPROPRNG;
```

Figure 93 (Part 1 of 2). Global Control Tables for the Apply Component in DB2 Parallel Edition

```
-- Create an UNIQUE INDEX on ASN.IBMSNAP_CD_CNTL
CREATE UNIQUE INDEX ASN.ZIBMSNAP_CD_CNTL
   ON ASN.IBMSNAP_CD_CNTL
   ( BASE_OWNER, BASE_TABLE );


-- Create the ASN.IBMSNAP_AUTH table
CREATE TABLE ASN.IBMSNAP_AUTH (
   GRANTOR        CHAR(18) NOT NULL,
   GRANTEE        CHAR(18) NOT NULL,
   GRANTOR_HIGH   CHAR(1)  NOT NULL,
   GRANTEE_LEVEL  CHAR(1)  NOT NULL,
   GRANTEE_STATUS CHAR(1)  NOT NULL )
IN DPROPRNG;

-- Create an UNIQUE INDEX on ASN.IBMSNAP_AUTH
CREATE UNIQUE INDEX ASN.ZIBMSNAP_AUTH
   ON ASN.IBMSNAP_AUTH
   ( GRANTOR, GRANTEE, GRANTEE_LEVEL );

-- Insert a row into ASN.IBMSNAP_AUTH
INSERT INTO ASN.IBMSNAP_AUTH
   ( GRANTOR, GRANTEE, GRANTOR_HIGH, GRANTEE_LEVEL, GRANTEE_STATUS )
   VALUES ( 'SYSIBM', USER, 'S', 'A', 'A' );

-- Commit changes
COMMIT;
```

Figure 93 (Part 2 of 2). Global Control Tables for the Apply Component in DB2 Parallel Edition

f. Run the ASNCNTL program to create the control tables as follows:

```
ASNCNTL /NC tpcd
```

You will see the following messages shown in Figure 94.

```
Installation of DataPropagator Relational control tables in progress...
CONNECT TO TPCD
Installation of control tables completed successfully!
```

Figure 94. Control Table Messages

g. Terminate the connection to the database by typing `db2 terminate`.

8. Bind the DataHub package to enable DataPropagator Relational for OS/2 to access the data server under DataHub for OS/2.

   From an OS/2 window in the directory where DataHub was installed (EMQDIR is the default value) type:

   ```
   SQLBIND @EMQS.LST centdb2 /K=ALL /G=PUBLIC
   ```

   Figure 95 on page 198 shows the messages that will appear.

```
   Database Connection Information

 Database product       = DB2 MVS 4.1.0
 SQL authorization ID   = DPROPADM
 Local database alias   = CENTDB2

LINE    MESSAGES FOR emqs.lst
------  ------------------------------------------------------------------
        SQL0061W  The binder is in progress.
LINE    MESSAGES FOR emqp2010.bnd
------  ------------------------------------------------------------------
        SQL0020W  Bind or precompile option(s) "LANGLEVEL" are not
                  supported by the target database and will be ignored.

LINE    MESSAGES FOR emq2cs20.bnd
------  ------------------------------------------------------------------
        SQL0020W  Bind or precompile option(s) "LANGLEVEL" are not
                  supported by the target database and will be ignored.
        SQL0091N  Binding was ended with "0" errors and "2" warnings.

DB20000I The SQL command completed successfully.
DB20000I The TERMINATE command completed successfully.
```

Figure 95. Database Connection Information

9. Bind the DataHub package to enable DataPropagator Relational for OS/2 to access the copy server under DataHub for OS/2.

   From an OS/2 window in the directory where DataHub was installed (EMQDIR is the default value) type:

   ```
   SQLBIND @EMQS.LST tpcd /K=ALL /G=PUBLIC
   ```

   The following messages should appear as shown in Figure 96.

```
   Database Connection Information

 Database product       = DB2/6000 PE 1.1.0
 SQL authorization ID   = DPROPADM
 Local database alias   = TPCD

LINE    MESSAGES FOR emqs.lst
------  ------------------------------------------------------------------
        SQL0061W  The binder is in progress.
        SQL0091N  Binding was ended with "0" errors and "0" warnings.

DB20000I The SQL command completed successfully.
DB20000I The TERMINATE command completed successfully.
```

Figure 96. Database Connection Messages

7.1.3 Installing Apply for AIX

Choose one DB2 Parallel Edition node where the Apply component will run.

1. Install the Apply component on that node using smit as shown in Figure 97 on page 199.

   In our environment we chose node one sp2n01 to install the Apply component.

Log on to the DB2 Parallel Edition on which you want the Apply component to be installed.

```
smitty install
  Install and Update software
    Install/ Update Selectable Software (Custom Install)
      Install Software Products at Latest Level
```

```
                  Install Software Products at Latest Level

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                  [Entry Fields ]
* INPUT device / directory for software             /mnt
* SOFTWARE to install                          [1.2.0.0  asnapply     > +
  PREVIEW only? (install operation will NOT occur)   no                      +
  COMMIT software updates?                           yes                     +
  SAVE replaced files?                               no                      +
  ALTERNATE save directory                           []
  AUTOMATICALLY install requisite software?          yes                     +
  EXTEND file systems if space needed?               yes                     +
  OVERWRITE same or newer versions?                  no                      +
  VERIFY install and check file sizes?               no                      +
  Include corresponding LANGUAGE filesets?           yes                     +
  DETAILED output?                                   no                      +



F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 97. Smit Menu Display

2. Create the links needed for the Apply files as shown in Figure 98.

```
ln -s /usr/lpp/asnapply/asnapply /usr/bin
ln -s /usr/lpp/asnapply/ASNLOAD /usr/bin
ln -s /usr/lpp/asnapply/asnapvrfy /usr/bin
ln -s /usr/lpp/asnapply/asnapply.bnd /usr/lpp/db2pe_01_01/bnd
ln -s /usr/lpp/asnapply/msg/En_US/asnapply.cat /usr/lib/nls/msg/En_US
ln -s /usr/lpp/asnapply/ASNLOAD.bnd /usr/lpp/db2pe_01_01/bnd
```

Figure 98. Symbolic Links to Files

3. The Apply program must have communication with the data server to extract the data from it, so you must define DB2 Parallel Edition as a DDCS client following the steps presented below.

   Figure 99 on page 200 shows relationships between the definitions that have to be done on DB2 Parallel Edition to act as a DDCS client.

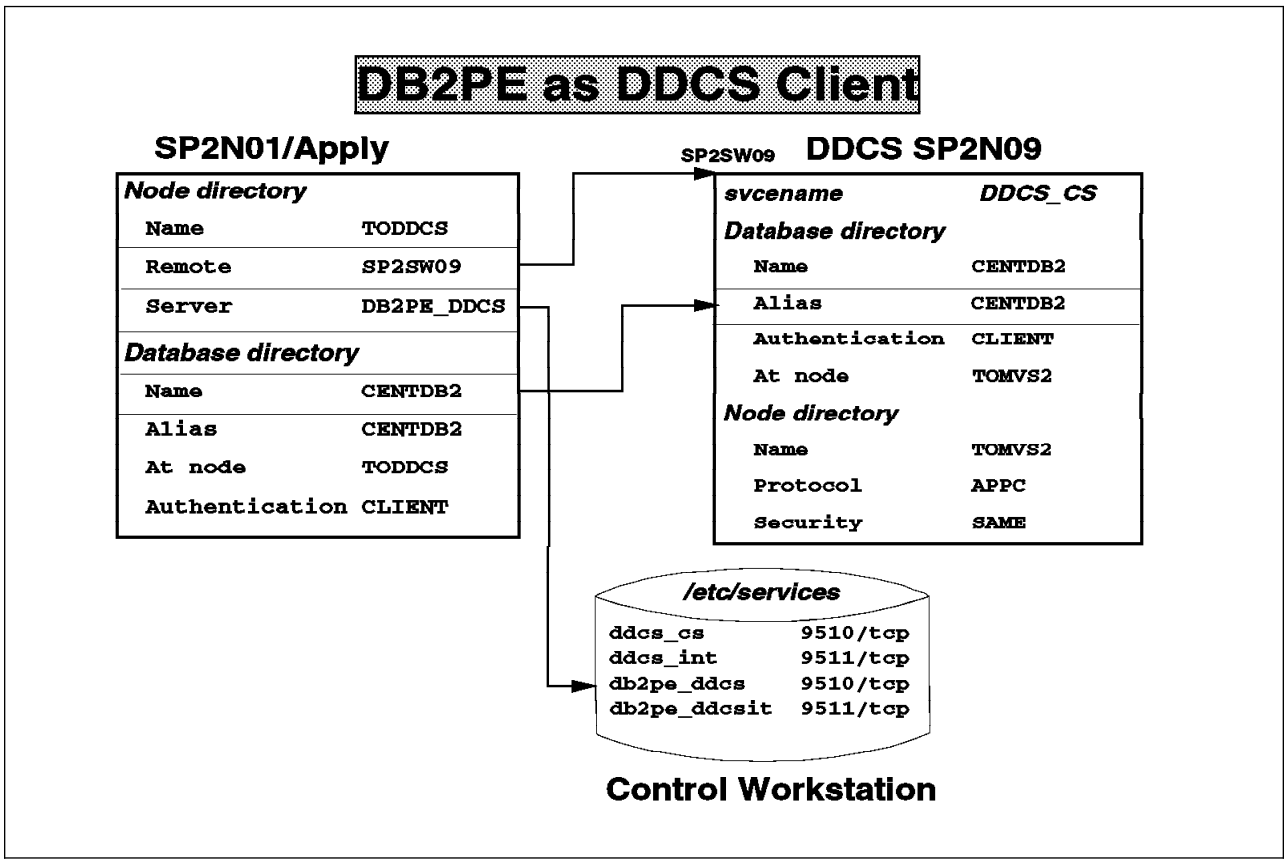


Figure 99. DDCS As a Server and DB2 Parallel Edition As a Client

a. Log on to the Control Workstation as root.

b. Edit the /etc/services file and add the following lines:

```
db2pe_ddcs      9510/tcp  # DB2PE client port for DDCS
db2pe_ddcsit    9511/tcp  # DB2PE client Interruption port for DDCS
```

Remember the port numbers defined must match the port numbers defined when you configured DDCS for remote clients. See step 3c1 on page 183.

c. Propagate the /etc/services updates to all RISC/6000 SP nodes:

```
dsh -a /var/sysman/supper upgrade user.admin
```

The Apply component requires access to DB2 for MVS without providing a password. This implies that the DB2 for MVS database must be cataloged using client authentication by completing the following steps:

1) Log on to the node where DDCS is installed as the DDCS instance owner.

2) Catalog an APPC node using the LU 6.2 Side Information Profile name and security same.

```
db2 catalog appc node TOMVS2 remote THBCP1 security same
```

3) Catalog the DB2 for MVS database at the node you created on the previous step.

```
db2 "catalog database CENTDB2 as CENTDB2C at node TOMVS2
authentication client"
```

d. Log on to the DB2 Parallel Edition node where the Apply component was installed as the DB2 Parallel Edition instance owner.

e. Catalog the DDCS Server on the DB2 Parallel Edition node directory.

```
db2 catalog tcpip node toddcs remote sp2sw09 server db2pe_ddcs
```

Note that the host name used in the catalog node command (sp2sw09) is the switch name to provide faster communications.

f. Catalog the data server database (DB2 for MVS) using the TCP/IP node created in the previous step.

```
db2 "catalog database centdb2c as centdb2c at node toddcs
authentication client"
```

The database name and the alias must match the alias defined for DB2 for MVS database in the DDCS Server.

4. Connect to the data server database (DB2 for MVS in our case).

```
db2 connect to centdb2c user DPROPADM
```

Provide the password and if the connection is successful, the following messages shown in Figure 100 will appear.

```
   Database Connection Information

 Database product       = DB2 MVS 4.1.0
 SQL authorization ID   = DPROPADM
 Local database alias   = CENTDB2C
```

Figure 100. Database Information

5. Issue the bind command for the Apply files against the data server (DB2 for MVS).

```
db2 bind /usr/lpp/asnapply/asnapply.bnd blocking all isolation cs
```

The resulting messages should look like the one in Figure 101.

```
LINE    MESSAGES FOR asnapply.bnd
------  ------------------------------------------------------------------------
        SQL0061W  The binder is in progress.
        SQL0091N  Binding was ended with "0" errors and "0" warnings.
```

Figure 101. asnapply messages

6. Terminate the connection to the data server; execute `db2 terminate`.

7. Connect to the copy server (DB2 Parallel Edition).

```
db2 connect to tpcd
```

When the connection is established, the following messages should appear as shown in Figure 102 on page 202.

```
   Database Connection Information

 Database product       = DB2/6000 PE 1.1.0
 SQL authorization ID   = DB2PEUSR
 Local database alias   = TPCD
```

Figure 102. Connection Messages

8. Bind the Apply program to the DB2 Parallel Edition database.

```
db2 bind /usr/lpp/asnapply/asnapply.bnd blocking all isolation cs
```

The resulting messages should look Figure 103.

```
LINE    MESSAGES FOR asnapply.bnd
------  ------------------------------------------------------------------------
        SQL0061W  The binder is in progress.
        SQL0091N  Binding was ended with "0" errors and "0" warnings.
```

Figure 103. asnapply.bnd Messages

7.1.4 Capture for MVS

The installation steps are clearly detailed in the *Program Directory for Capture MVS*. You should follow the steps presented there except for the last one that starts the Capture component and assumes that at least one registration and subscription has been done. This step is done later.

7.1.5 Creating DataPropagator Relational Users, Registrations and Subscriptions

To use DataPropagator Relational, you must have authorization privileges, which vary based on three user roles:

- Administrator
- Registrar
- Subscriber

Some privileges are granted from DataPropagator Relational for OS/2; others however, must be granted on the relational database management systems acting as copy, control or data servers.

Tables must be registered to DataPropagator Relational before they are eligible to be copied.

The subscription process to a base table implies automatic, periodic delivery of current data from the base table.

In this section we will discuss how to use the DataPropagator Relational for OS/2 for granting the privileges, how to register source tables on DB2 for MVS and making the subscriptions on DB2 Parallel Edition.

1. Authorize the registrar

a. On the DataHub workstation, make sure DB2 for OS/2 is started.

b. Log on with the DataPropagator Relational administrator's user ID, DPROPADM in our example.

c. Click on the **IBM DataHub** icon.

d. The managed hosts are presented, open the MVS host by double-clicking on it.

e. From the Display Related Objects window select **RDB** and click on **Display**.

f. Select the managed database alias for the DB2 for MVS system, CENTDB2 in our example, and double-click on it.

g. From the list presented on the Display Related Objects window, select **(DPropR) Registrars to be authorized** and click on **Display**.

h. From the list presented of user IDs under the title (DPropR) Registrars to be authorized, highlight the one you will authorize as the registrar, REGISTRA in our example.

i. On the DataPropagator Relational cascade menu select Grant Authority.

   Go to the Actions menu and select Copy, DPropR and Grant Authority.

j. Verify the operation by displaying the (DPropR) Registrars - authorized.

   Double-click on **CENTDB2** and from the Display Related Objects window select **(DPropR) Registrars - authorized** and click on **Display**. You should see the user ID selected under the (DPropR) Registrars - authorized.

   Also, you can do a refresh of the list of the (DPropR) Registrars to be authorized by highlighting the title and using the Refresh option of the Tree menu.

2. Authorize the subscriber

   a. Log on locally as the registrar user ID you have just authorized.

   b. Select the managed database **CENTDB2**, by double-clicking on it.

   c. From the list presented select **(DPropR) Subscribers to be authorized** and click on **Display**.

   d. Select the user ID of the user for which the subscription is created.

   e. On the DataPropagator Relational cascade menu, select **Authorize**.

      Go to the Actions menu and select Copy, DPropR and Grant Authority.

   f. Verify the operation by displaying the (DPropR) Subscribers - authorized.

      Double-click on **CENTDB2** and from the Display Related Objects window select **(DPropR) Subscribers - authorized** and click on **Display**. You should see the user ID selected under the (DPropR) Subscribers - authorized.

      Also you can do a refresh of the list of the (DPropR) Subscribers to be authorized, by highlighting the title and using the Refresh option of the Tree menu.

3. Register the source table

   DB2 for MVS Version 4 type 2 indexes are not supported registration process and therefore, must be done manually.

The steps required to register a user table are:

a. Choose the source table to be registered.

b. Insert a row in the change data control table, IBMSNAP_CD_CNTL, registering the source table as a base table.

   You can use the following statement in Figure 104 as an example.

```
INSERT INTO ASN.IBMSNAP_CD_CNTL(
  BASE_OWNER,  BASE_TABLE,
  CRIT_SECTION_OWNER, CRIT_SECTION_TABLE,
  CD_OWNER, CD_TABLE,
  PRUNE_CNTL_OWNER,  PRUNE_CNTL_TABLE,
  CD_OLDEST_CHANGE,  FULLREFRESHENABLE,
  CCD_OWNER,  CCD_TABLE,
  CCD_OLDEST_CHANGE,  CD_CONDENSED,
  CD_CONSISTENT,  CD_COMPLETE,
  CCD_CONDENSED,  CCD_CONSISTENT,
  CCD_COMPLETE,  BASE_STRUCTURE,
  BASE_CONDENSED,  BASE_CONSISTENT,
  BASE_COMPLETE,  REGISTRAR,
  ARCH_LEVEL,   DESCRIPTION,
  PREFIX_CHAR)
  VALUES
 ('DB2MVS', 'PART',
  'ASN', 'IBMSNAP_CRITSEC',
  'DB2MVS',  'CD01',
  'DB2MVS',  'PC01',
  X'00000000000000000000',  1,
  NULL,  NULL,
  X'00000000000000000000', 'N',
  'T', 'N',
   '',   '',
   '',  1,
   'Y',  'T',
   'Y',  'REGISTRA',
   '0101',  NULL,
   'X');
```

Figure 104. Change Data Control Table

Figure 104 is an example of the information needed to register on the change data control table the DB2MVS.PART table. The owner and name of the change data and pruning control tables are also provided (DB2MVS.CD01 and DB2MVS.PC01). For a complete description of each of the columns of the change data control table see *DataPropagator Relational Guide Release 2*, SC26-3399.

c. Create a segmented tablespace for the change data (CD) table.

d. Create the change data table.

e. Create a tablespace for the pruning control (PC) table.

f. Create the pruning control table.

g. Create an index on the pruning control table.

All the SQL required to complete the five previous steps can be put on a text file. Use the DataHub window to execute the statement in DB2 for MVS database. Follow these steps:

a. Create a file such as the one shown in Figure 105 on page 205.

```
CREATE  TABLESPACE CDTS01 IN DPROPR1
USING STOGROUP SYSDEFLT
PRIQTY 12
SECQTY 12
ERASE NO
SEGSIZE 4
LOCKSIZE TABLE
BUFFERPOOL BP0
CLOSE NO;

CREATE TABLE DB2MVS.CD01
( IBMSNAP_UOWID CHAR(10) FOR BIT DATA NOT NULL,
  IBMSNAP_INTENTSEQ CHAR(10) FOR BIT DATA NOT NULL,
  IBMSNAP_OPERATION CHAR(1) NOT NULL,
  P_PARTKEY INTEGER NOT NULL,
  P_NAME VARCHAR(55) NOT NULL,
  P_SIZE INTEGER NOT NULL)
IN DPROPR1.CDTS01;

CREATE  INDEX DB2MVS.CD01X ON
DB2MVS.CD01
(IBMSNAP_UOWID DESC)
BUFFERPOOL BP0
CLOSE NO
USING STOGROUP SYSDEFLT
PRIQTY 12
SECQTY 12;

CREATE  TABLESPACE PCTS01 IN DPROPR1
USING STOGROUP SYSDEFLT
PRIQTY 2
SECQTY 1
ERASE NO
LOCKSIZE PAGE
BUFFERPOOL BP0
CLOSE NO;

CREATE  TABLESPACE PCTS01 IN DPROPR1
USING STOGROUP SYSDEFLT
PRIQTY 2
SECQTY 1
ERASE NO
LOCKSIZE PAGE
BUFFERPOOL BP0
CLOSE NO;

CREATE UNIQUE INDEX DB2MVS.PC01X ON
DB2MVS.PC01
(COPY_OWNER, COPY_TABLE)
BUFFERPOOL BP0
CLOSE NO
USING STOGROUP SYSDEFLT;
```

Figure 105. Table Space Example

b. Select the DB2 for MVS database alias, CENTDB2 in our scenario.

c. From the Actions menu choose the **Run** option.

d. On the DataHub - Run window click on the **Input file..**

e. Locate the file you created and click on **OK**.

f. Run the commands on the input file and click on **Run**.

g. Verify that there were no errors by a return code of 0.

4. Create the subscription

   The subscription process includes the following tasks:

   - Defining the target table name and location
   - Defining the target table columns
   - Defining the target table predicate
   - Setting the copy frequency and the number of times to copy the source table
   - Specifying the copy mode
   - Defining SQL to execute before or after the table is copied
   - Auto-registration

     Auto-registration consists of registering the target table at the copy server and creating the pruning control table at the copy server.

   When the first subscription is made, the control tables on the control server are created. The following are the control tables:

   - The refresh control table, subscriber.IBMSNAP_REF_CNTL
   - The refresh columns table, subscriber.IBMSNAP_REF_COLS
   - The routing table, subscriber.IBMSNAP_ROUTING
   - The audit trail table, subscriber.IBMSNAP_TRAIL

   Following subscriptions under the same subscriber only updates these control tables.

   The following are the copy tables that can be defined during a subscription are:

   Point-in-time

   A complete, condensed copy of the source table, which must have a primary key.

   Base aggregate

   A history table in which a new row is appended for each changed row in the source table with the result of an SQL column function calculation against the existing user table.

   Change aggregate

   A history table in which a new row is appended for each changed row in the source table with the result of an SQL column function calculation against only recent changed data.

   Condensed, noncomplete Consistent Change Data (CCD)

   A small staging table that, when defined locally to the source, is useful for providing a stable source for effecting synchronization of fan out copies.

   Condensed, complete CCD

   A full-size staging table useful for efficient remote staging, which allows for remote copies to be both initialized and maintained without needing to access the original source each time it is updated.

   Noncondensed, noncomplete CCD

A table that starts out empty and is appended by each insert, update, and delete. Useful for auditing purposes.

Noncondensed, complete CCD

A table that starts out as a copy of the full-sized source table and is appended by each insert, update and delete.

The SQL data definition statements of DB2 Parallel Edition are not directly supported via the DataPropagator Relational for OS/2 graphical interface. The subscription process to the DB2 for MVS tables must be done manually.

A single node node group on DB2 Parallel Edition must be used to create the control tables.

Point-in-time copies require the definition of a primary key to apply the updates. In DB2 Parallel Edition, the primary key of a table must be a subset of the partitioning key. The columns conforming the partitioning key cannot be updated.

You must keep this consideration in mind before creating your target copy table. If the user transactions update the partitioning key of the target table, define it on a single-node node group.

The steps described below are an example of the SQL statements needed to define a point-in-time copy from a DB2 for MVS table on DB2 Parallel Edition. The partitioning key of the target table will not be updated by the user transactions.

a. Select one of the registered source tables.
b. Create the refresh control table.
c. Create an index on the refresh control table.
d. Insert global information into the refresh control table.
e. Create the refresh columns table.
f. Create an index on the refresh columns table.
g. Insert subscription information in the refresh columns table.
h. Create a routing table.
i. Create the copy table.

You can create a file containing the proper SQL statements and execute it directly on one of the DB2 Parallel Edition nodes or use the Run option on the Actions menu from DataHub as explained previously.

The file will look like the one presented in Figure 106 on page 208.

```
-- Create a nodegroup for the control tables on one node
create nodegroup dproprng on nodes (0);

-- Create the refresh control table
create table ddcsusr.ibmsnap_ref_cntl
 (enable smallint not null,
  data_server char(18) not null,
  base_owner char(18) not null,
  base_table char(18) not null,
  stmt_number smallint not null,
  priority smallint not null,
  copy_owner char(18) not null,
  copy_table char(18) not null,
  last_diff_sequence char(10) for bit data,
  runmode char(2) not null,
  status smallint not null,
  ibmsnap_logmarker timestamp,
  lastrun timestamp,
  interval_minutes int not null,
  copy_condensed char(1) not null,
  copy_complete char(1) not null,
  copy_consistent char(1) not null,
  copy_structure smallint not null,
  arch_level char(4) not null,
  copy_server char(18) not null,
  expires date,
  predicates varchar(512),
  sql_stmt varchar(1024)) in dproprng;

-- Create an index on the refresh control table
create unique index ddcsusr.cntlx
  on ddcsusr.ibmsnap_ref_cntl
  (copy_server,copy_owner,
   copy_table, stmt_number);

-- Insert global information into the refresh control table.
insert into ddcsusr.ibmsnap_ref_cntl
values (
         1,'',
         '','',
         0,0,
         '','',
         null,
         '',0,
         null,null,
         0,'',
         '','',
         32,
         '0101','',
         null,null,
         null);
```

Figure 106 (Part 1 of 4). SQL Statements of Manual Subscription for a Point-in-Time Copy

```
-- Create the refresh columns table
create table ddcsusr.ibmsnap_ref_cols
 (copy_server char(18) not null,
  copy_owner char(18) not null,
  copy_table char(18) not null,
  stmt_number smallint not null,
  col_type char(1) not null,
  target_name char(18) not null,
  is_key char(1) not null,
  colno smallint not null,
  expression varchar(254) not null) in dproprng;

-- Create an index on the refresh columns table
create index ddcsusr.colsx
on ddcsusr.ibmsnap_ref_cols
(copy_server, copy_owner, copy_table, stmt_number);

-- Create an the audit trail table
create table ddcsusr.ibmsnap_trail
( enable smallint not null,
  data_server char(18) not null,
  base_owner char(18) not null,
  base_table char(18) not null,
  stmt_number smallint not null,
  copy_owner char(18) not null,
  copy_table char(18) not null,
  last_diff_sequence char(10) for bit data,
  runmode char(2) not null,
  status smallint not null,
  ibmsnap_logmarker timestamp,
  lastrun timestamp,
  fetched int not null,
  fullrefreshed int not null,
  inserted int not null,
  deleted int not null,
  updated int not null,
  reworked int not null,
  interval_minutes int not null,
  sqlstate char(5),
  sqlcode int,
  sqlerrp char(8),
  apperrn char(8),
  copy_server char(18) not null,
  cntl_server char(18) not null,
  sqlerrm varchar(70),
  apperrm varchar(760))
in dproprng;
```

Figure 106 (Part 2 of 4). SQL Statements of Manual Subscription for a Point-in-Time Copy

```
-- Insert subscription information in the refresh control table
insert into ddcsusr.ibmsnap_ref_cntl values
(1,'CENTDB2',
  'DB2MVS','PART',
  100,10,
  'DDCSUSR','PART',
  null,'AO',
  0,null,
  null,1,
  'Y','Y',
  'T',4,
  '0101','TPCD',
  null,null,
  null);

-- Insert subscription information in the refresh columns table
insert into ddcsusr.ibmsnap_ref_cols values
( 'TPCD','DDCSUSR',
  'PART',100,
  'A','P_PARTKEY',
  'Y',0,
  'P_PARTKEY');

insert into ddcsusr.ibmsnap_ref_cols values
( 'TPCD','DDCSUSR',
  'PART',100,
  'A','P_NAME',
  'N',1,
  'P_NAME');

insert into ddcsusr.ibmsnap_ref_cols values
( 'TPCD','DDCSUSR',
  'PART',100,
  'A','P_SIZE',
  'N',2,
  'P_SIZE');

-- Create a routing table
create table ddcsusr.ibmsnap_routing
( cntl_server char(18) not null,
  try_connect_time timestamp) in dproprng;

create unique index ddcsusr.routingx
on ddcsusr.ibmsnap_routing (cntl_server);

insert into ddcsusr.ibmsnap_routing
values ( 'TPCD',null);
```

Figure 106 (Part 3 of 4). SQL Statements of Manual Subscription for a Point-in-Time Copy

```
-- Create the node group for the table
 create nodegroup dproprng2 on all nodes;

-- Create the copy table
 create table DDCSUSR.PART
 (
  P_Partkey integer not null,
  p_name varchar(55) not null,
  p_size integer not null,
  ibmsnap_logmarker timestamp
 )
  in dproprng2;

 create unique index DDCSUSR.PARTX on DDCSUSR.PART
 (p_partkey);
```

Figure 106 (Part 4 of 4). SQL Statements of Manual Subscription for a Point-in-Time Copy

7.1.6 Starting the Components

Run the job ASNLRN41 to start DataPropagator Relational Capture for MVS. Ensure that the user ID used to invoke Capture for MVS has sufficient privileges to run it.

Start the Apply for AIX component by completing the following steps:

1. Log on to the DB2 Parallel Edition node where the Apply component was installed with the subscriber user ID.
2. Make sure that the default database is the copy server name. If not, you can set it with the export command:

   `export DB2DBDFT=<target database name>`
3. Ensure that the locale is not set to C.
4. Make sure the subscriber user ID has write permission on the directory and that it belongs to the system group.

   The user ID running the Apply component is the user ID that will be verified at the client and passed to DB2 for MVS.
5. Enter `asnapply 1`

7.2 Considerations

Loading large copies using Apply is not very efficient. You can perform your own load and fool the Apply component into thinking it has initialized your large point-in-time copies.

One possible scenario is to define a subscription but include a dummy predicate that restricts the copy to few or no rows. The Apply component will be updating the pruning control table and you will now have a synchronization time stamp. After creating the subscription, extract the table and load it using the Replace option. You may leave the IBMSNAP_LOGMARKER time stamp column null as this is a

full refresh. Then remove the dummy predicate from the subscription definition. Now the apply component will maintain the copy without having initialized it.

The Apply component will perform a differential refresh thereafter. As an extra measure, you will likely want to turn off the FULLREFRESHENABLE flag in the change data control table (CD_CNTL) at the data server. This will prevent the Apply component from trying to initialize new copies via the normal online initialization process.

The following is another scenario:

Define subscriptions and set activate = no.

1. Update the pruning control table associated with the base table.

   Set copy_server, copy_owner and copy table to the fully qualified name of your copy, ibmsnap_logonmarker to a time stamp value approximating the current time, prune_limit_seq to x'00000000000000000000', cntl_server to the name of the control server and copy_structure matching the value of the same column in the ibmsnap_ref_cntl table. In the IBMSNAP_REF_CNTL definitions, set the value of lastrun to the same time stamp value set in the pruning control table row.

2. Unload the data from DB2 for MVS

3. Load the data, assuming the copy is a poin-in-time structure use a default constant for the value of ibmsnap_logonmarker column. This timestamp to be meaningful to those looking at the data should approximate the time of the unload.

4. Set the enable column for subscriptions in the IBMSNAP_REF_CNTL table to +1.

   As long as Capture/MVS and Apply/6000 are running, the copies should now only be differentially updated as the full refresh process has completed.

When you start Apply for AIX you can specify a parameter to call ASNLOAD, an exit routine that can call any IBM or vendor utility to efficiently perform this initial refresh. The EXPORT/IMPORT utility is called by default to perform the full refresh operation.

It is recommended to start the Apply component with the L parameter since for initial loading of tables this can be quicker than simple inserts.

Chapter 8. DB2 Parallel Edition AutoLoader Tool

This chapter describes another method of extracting and loading data to DB2 Parallel Edition. This is implemented with the use of a DB2 Parallel Edition tool called Autoloader. With the use of this tool, the process of data extraction and loading is implemented and executed much more easily in that the entire process is executed in one shot.

8.1 AutoLoader

AutoLoader is a tool to transfer data from a source (such as MVS/VM datasets or AIX files) to AIX. The data is partitioned and it is loaded into DB2 Parallel Edition on the corresponding node. AutoLoader consists of a set of executable shell scripts that automate the loading process based on standard DB2 Parallel Edition split and load functions.

Note: The DB2 Parallel Edition AutoLoader will be available with DB2 Parallel Edition Version 1 Release 2.

8.1.1 Install AutoLoader

The following steps are recommended when installing AutoLoader.

- Check the directories of db2, db2split and mknod programs and modify shell variables DB2_DIR, DB2SPLIT_DIR and MKNOD_DIR in file generate_fifo, if needed.
- The shell program ksh is assumed to be in the /bin directory. If you find it in some other directory, change the first lines (`#!/bin/ksh`) of all the scripts to `#!your_ksh_directory/ksh`. You can test this by using the `which ksh` or `whence -v ksh` command.
- Make sure that the script rksh is in this directory.
- Put in $HOME/.netrc the following information:
  machine <machine_name> login <login_name> password <password>
  where <machine_name> is the host name of the source_system (that is the place we need to FTP from to get the data file). This is required if you want to load the data from the files on other host.
- Ensure that the permissions of .netrc files are -rw-------
  If not, use the following command to change the permissions:
  `$ chmod 600 .netrc`.
- If you are using delimited ASCII source files, you will have to adapt the file generate_fifo in the directory /autoloader. AutoLoader assumes that the '|' (pipe) character is the delimiter. If this is not the case for your source files, then edit the generate_fifo file and change `coldel|` to coldel <your delimiter>.
- AutoLoader expects the first column of db2nodes.cfg file (in $HOME/sqllib) to be free from leading 0 (zero). For example:

```
 0 <node1> 0 <switch1>
 1 <node2> 0 <switch2>
not
 00 <node1> 0 <switch1>
 01 <node2> 0 <switch2>
```

- Create a directory to store your configuration files.

8.1.2 Customize Environment Files

To run AutoLoader you have to provide several configuration files with source/path, split and load information. Those files have the extensions *.spec, *.cfg and *.load. To clarify the parameters in the configuration files, you will find examples of how to load table ORDERS in TPCD from the source local (local is in our example a named pipe).

8.1.2.1 Modify *.spec File

The purpose of the *.spec file is to provide AutoLoader all environment information needed to load data into DB2 Parallel Edition. The syntax is:

```
<directory_name on source_system>
<file_name 1> <table_name> <cfg file> <load script>
<file_name 2> <table_name> <cfg file> <load script>
<file_name 3> <table_name> <cfg file> <load script>
      :             :          :           :
<file_name 3> <table_name> <cfg file> <load script>
```

Figure 107. Syntax for AutoLoader Specification File

<directory_name on source_system>
Insert the name of the directory where the file to be loaded exists. If you are using a named pipe, indicate here the directory where this pipe will be initiated.
Leave the first line blank if the source_system is MVS.

<file_name x>
This is the name of the file to be loaded into DB2 Parallel Edition. If source is a named pipe indicate the name of the pipe.

<table_name>
Corresponding table name to load data into.

<cfg file>
The configuration file will be used by db2split. For more information, see 8.1.2.2, “Modify *.cfg File” on page 215.

<load script>
This configuration file will be used by db2load to load the (partitioned) data. AutoLoader expects a delimited (default pipe sign (|)) ASCII file as input and will do as many iterations as lines are available in this specification file. There is no possibility to comment a line in this file.

The following is an example of the spec file to load delimited data into the table ORDERS. Since the source in this example is delimited data, no load script is needed.

```
/u/db2peusr/tools/autoloader
local orders /u/db2peusr/cfgfiles/orders.cfg
```

For more information, see 8.1.2.3, “Modify *.load File” on page 217.

8.1.2.2 Modify *.cfg File

The AutoLoader configuration file contains all information to run db2split. Keywords in this file are not case sensitive and the order of them is not important. Enclosed you will find an example for loading the ORDERS table into the TPCD database using the named pipe called local.

```
; ************************************************************
; db2split configuration file for table ORDERS in TPCD
; ************************************************************
Description=tpcd
InFile=/u/db2peusr/tools/autoloader/local
RecLen=32000
Nodes=(1,2,3,4,5,6,7,8,11,12)
OutputNodes=(1,2,3,4,5,6,7,8,11,12)
OutputType=w
; MapFili=orders.map
MapFilo=orders.map
DistFile=Orders.distfile
LogFile=orders.log,w
OutFile=orders
CDelimiter=|
SDelimiter='
RunType=PARTITION
Msg_Level=WARN
Check_Level=NOCHECK
Partition=O_ORDERKEY,1,1,4,NN,INTEGER
Trace=100
; ************************************************************
```

Figure 108. Example: Configuration File orders.cfg

Description: This description will appear as a title in the log files. AutoLoader will update this information during run time with the database name.

InFile: Name of the input file for db2split. AutoLoader will update this information with the name of the pipe created during execution.

RecLen: This number indicates the input file record length.

- **Delimited data:** Take any value that is greater than the maximum physical record length of the input file but smaller or equal 32000. The db2split utility will allocate a buffer equal to this size. The bigger this number is, the more memory will be used, and that might impact the performance. Using delimited records, character data is the only valid data type.
- **Non-delimited data:** In case of using non-delimited data, make sure that the field `RecLen` is exactly equal to the length of columns in the data file (excluding end-of-line character). Every single input data line *must* have exactly the same length. All type of characters are allowed, including binary zero.

  Note: For VB (variable block size) file format from VM/MVS, the block size is a multiple of record length +4. Do not count the return character for this parameter. The program db2split will adjust itself accordingly, depending on where it is executed (VM/MVS or AIX).

  For FB (fixed block size) files, this value must be equal to that defined in the DCB (data control block) of the output files. Failures will either result in truncated output files or in blank output files.

Nodes: The Nodes parameter indicates on which nodes the table resides and will be populated with data. Make sure that the figures of this parameter are identical with the nodenum definition in the $HOME/sqllib/db2nodes.cfg or a subset of them. An alternative way to list the nodes in this parameter is Nodes=(1,2,m-n,...999).

Instead of indicating Nodes, you can also use the MapFili (partitioning map input file) parameter.

OutputNodes: Specify *every* node (0-999) in the OutputNodes parameter where the data is going to be stored. The numbers of nodes in OutputNodes is equal to the figures in Nodes or a subset of them. If you do not declare OutputNodes but the parameter Nodes is defined, AutoLoader will use the information from Nodes as default for OutputNodes. If nothing is indicated neither in OutputNodes nor in Nodes AutoLoader will take the information about available nodes from the file db2nodes.cfg.

Note: If MapFili parameter has been chosen instead of Nodes, you should declare OutputNodes.

OutputType: If OutputNodes only has one member, you can use the option OutputType to define whether the output should be written to a file (=w) or piped to stdout (=s). Default for this parameter is s. If OutputNode has more than one member, this option will be ignored.

MapFili: Indicate here the name of the input partitioning map file. This file is used to get the number of nodes to partition the data if no information is provided in the Nodes parameter. To get this information, run db2split with the parameter RunType in ANALYZE mode. After executing db2split in ANALYZE mode change RunType back to PARTITION.

Note: Specify either Nodes or MapFili but not both.

MapFilo: Put the name of the output partitioning map file here.

DistFile: Indicate the name of the distribution file. Default name is DISTFILE. This file can be used for the DB2 redistribution utility.

LogFile: Add the name of the table here. If you enter another name, AutoLoader will replace it with the table name. In addition your can specify a parameter to overwrite or to append your log information. Use:

w = overwrite
a = append

OutFile: This parameter will become an output file prefix. AutoLoader will add the table name to that parameter and append a five character suffix (00000..00999) to the end of this prefix to generate an output file name for example, ORDERS.00001. Under AIX, the output files will be named prefix.suffix. If the OutFile keyword is not specified (commented out), the default output filename prefix will be NOD.
In VM/MVS the output filename prefix will always be NOD. The user can optionally choose other filename when invoking the JCL/EXEC file.

CDelimiter: This parameter indicates, which column delimiter is used for db2split. Both db2split and db2load support following delimiters:
{ , " % & ' ( ) * . / : ; < = > ? _ |

Note: If the source file is non-delimited, this parameter has to be commented out.

SDelimiter: Indicate here the string delimiter. By default, SDelimiter is a double quote ("). SDelimiter has to be different from CDelimiter and cannot be a period (.).

RunType: RunType can either be ANALYZE or PARTITION. Running db2split with the parameter ANALYZE will produce a partitioning map and store this information in a file that can be used as input in MapFili. If you provide information for Nodes, you do not have to run ANALYZE. To split data you have to choose PARTITION.

Msg_Level: With Msg_Level = WARN, db2split will stop at every warning message. By indicating NOWARN (default), db2split will continue running at warnings.

Check_Level: Check_Level can be either CHECK or NOCHECK, where CHECK will check for truncation of records at input/output. For MVS, this will turn on binary checking for the first 100 records. For NOCHECK, the program will not check for truncation of records at I/O. CHECK is the default.

Partition: This parameter contains the following six fields:

- Column name: Name of the partitioning key used for output to log file.
- Position: Position of the partitioning key in each record Only valid for DEL data.
- Data Offset: Start of partitioning key (starts at column 1). Only valid for non-delimited data.
- Length: of partitioning key. Valid for non-delimited data. For CHARACTER and VARCHAR delimited data, this field *must* be specified as well. The length must be consistent with the corresponding column length in the database table.
- Null indicator:
  - N: Null data is allowed
  - NN: Data must not be Null
  - NNWD: Not Null With Default, will be treated the same way as NN
- Type of data conversion
  Used for hashing:
  - SMALLINT
  - INTEGER
  - CHARACTER
  - DECIMAL(x,y) (example: 123.45 would give x=5, y=2)
  - DATE in YYYY-MM-DD format
  - TIME in HH.MM.SS format
  - TIMESTAMP in YYYY-MM-DD-HH.MM.SS.XXXXXX format

One Partition statement is used for each column of the partitioning key and no spaces are allowed between the fields.

Trace: Trace indicates the number of records to be traced. It provides a dump of the data conversion process and output of hashing values.

8.1.2.3 Modify *.load File

Using non-delimited files, you must create an input file for db2load with the following format:

```
db2 connect to mydatabase
db2 "load from inputfile of asc method l (1,4,6,10,14,18) replace into mytable"
db2 connect reset
```

Figure 109. Example of a Load Configuration File

For more information about db2load, refer to *DB2 ParallelEdition for AIX, Administration Guide and Reference*, SC09-1982.

8.1.3 Start AutoLoader

Before invoking AutoLoader, ensure that database and tables to be populated with data are already created. You can either start AutoLoader from the command line or from a shell script.

The following is the syntax for:

```
autoloader [-d] [- h host_name] [-s spec_file] database_name
```

Figure 110. Syntax to Start AutoLoader

where the options are:

```
- d stands for delimited data. The default is non-delimited data.
- h Transfer the files from a remote place for loading the data.
    Omit this option if the source is local.
- s configuration file for db2split.
```

Example:

```
autoloader -d -h /db2piofs/data/order -s /u/db2peusr/cfgfiles/orders.cfg tpcd
```

AutoLoader creates a log file called autoloader.log. It contains messages from nodes that have completed the load process. Check this file to ensure all load activities completed successfully. You will find one entry per table per node loaded. That means, if you are loading M tables on N nodes, you will see M * N entries in autoloader.log.

AutoLoader also creates a file called <table_name>.log. This contains messages from the split process.

Finally run the `cleanup` program provided with AutoLoader to clean up all temporary files and pipes.

8.1.3.1 AutoLoader Activities

The following graphic shows the activity flow initiated by AutoLoader:

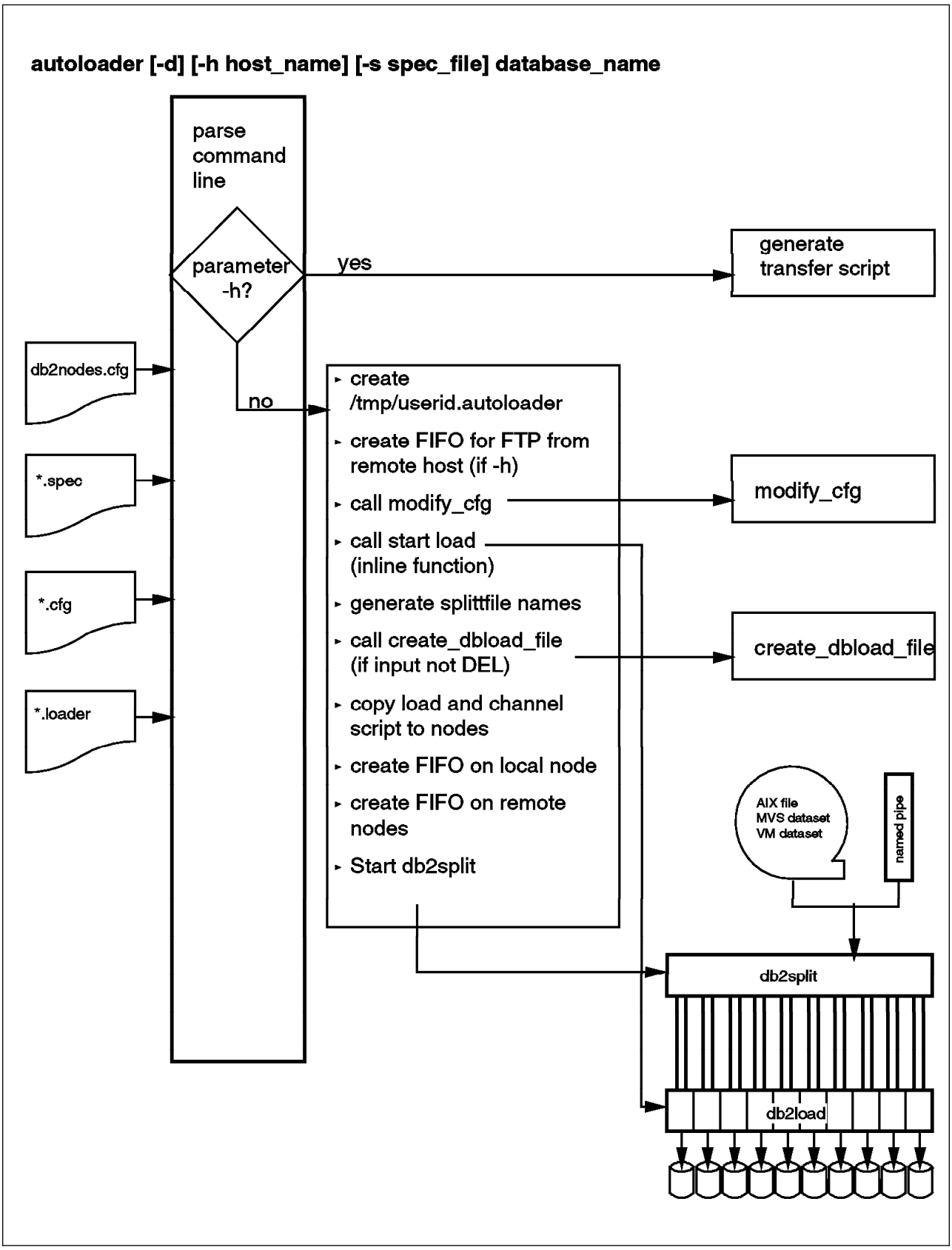


Figure 111. AutoLoader Processes

After having updated all environment information (*.spec, *.cfg, *.loader) enter the following command:

```
autoloader [-d] [-h] -s *.spec database-name
```

AutoLoader will execute following steps:

1. Process command line options.

   If <host_name> was indicated, AutoLoader calls the executable file generate_transfer. This script prepares a FTP file for a later FTP session started by AutoLoader as the last action. To use CLIO FTP or CLIO pipes, modify this file (<generate_transfer>) prior to execution of AutoLoader.

2. Call generate_fifo.

   This script creates all FIFO (named pipes) files, starts db2load and db2split.

3. Create on each node specified in db2nodes.cfg a directory called /tmp/myuserid.autoloader.

4. If the parameter -h is used, AutoLoader creates a named pipe (FIFO) on the local node (where AutoLoader runs) to get input from a remote host via FTP. The name of the pipe will be the first parameter of *.spec file.

5. AutoLoader gets the table name, name of db2split configuration file and name of load script from autoloader.spec.

6. AutoLoader then calls modify_cfg. This script sets the following parameters in the db2split configuration file (*.cfg)to:

   - InFile to FIFO name created in step 4
   - LogFile to table name
   - OutFile to table name
   - Description to database name

7. Calls script start_load. When the parameter OutputNodes in *.cfg is specified, AutoLoader executes for each node in OutputNodes following steps:

   - Creates a filename for split files like tablename.0000n.
   - If parameter InFile is not set to delimited, it calls script <create_dblod_file>.
   - AutoLoader copies this load script (only if not delimited data) to /tmp/userid.autoloader.
   - It creates named pipes on the remote node with a split file name like TABLENAME.0000n and starts the channel program to read from stdin and writes to the split file.
   - It starts db2load to read from (remote) splitfile (/tmp/userid/autoloader/splitfile).

   If OutputNodes are not specified in *.cfg, AutoLoader will use db2nodes.cfg for executing the steps above.

8. Start db2split and write to local pipes.

9. If -h is specified on the AutoLoader command, start FTP.
   The db2split program will read from FIFO (named pipes) created in step 4 (gernerate_fifo).
   If -h is not specified, db2split will read from local files.

Note

Since AutoLoader will create two named pipes for all nodes affected by db2load, you have to ensure that DB2 settings for maximum number of concurrent applications (default of *maxappls* is 20) is at least equal to the number of nodes you are going to load data to. If users are connected to DB2, the number of maxappls has to be increased accordingly.

Do not forget to execute cleanup.

8.1.4 Customizing AutoLoader Script Files

8.1.4.1 Accessing Mainframe Datasets with AutoLoader

If you want to access data sets on a VM/MVS system make sure of the following:

- The file $HOME/.netrc is created and contains the necessary information to access the mainframe. Set the permission with the following command `chmod 600`.
- The first line in your *.spec file is empty.
- The filename in *.cfg and *.spec is indicated correctly.
- The command for starting AutoLoader contains the parameter -h to indicate the remote source. Example:

```
autoloader -d -h db2mvs -s nation.spec tpcd
```

Figure 112. Run AutoLoader Accessing Files on MVS

Starting AutoLoader with the -h parameter will force it to run the shell script generate_transfer. This script will produce an executable file with a name ftp.<dataset name> that will be executed during the AutoLoader process to get the data set.

```
ftp db2mvs << EOF
ha
get db2mvs.nation99
quit
EOF
```

Figure 113. File ftp.db2mvs.nation99 Generated by generate_transfer

For the download of the data set, AutoLoader will, in addition to the process described above, do the following:

- Create a pip with the name of the data set in the current directory
- Connect to the remote system
- Get the data
- Close connection

8.1.4.2 Using CLIO FTP

If you want to use CLIO FTP instead of standard FTP, you can very easily switch by modifying the shell script generate_transfer. Replace the word FTP with CLFTP as shown in the file extract below.

```
 ....

for file in  awk '{ if (NR > 1) print $1}' $spec_file
do
   # Writing ftp script for file
   echo "clftp $hostname << EOF" > ftp.$file               <======
   # echo "bin" >> ftp.$file
   echo "ha" >> ftp.$file
   awk '{ if (( NR == 1) && (NF > 0))
             printf "cd %s\n", $1
           else
             if ((NR > 1) && ( $1 == "'$file'"))

  ...
```

Figure 114. Modify Script generate_transfer for CLIO FTP

8.1.4.3 Change Using High Performance Switch / Ethernet

Depending on whether we have a High Performance Switch installed in your System or not, you my have to change the AutoLoader shell script generate_fifo to make use of one or the other. The script generate_fifo will look up in the file /sqllib/db2nodes.cfg for the address of the switch or Ethernet, depending on the position you indicate in the shell script below.

```
#!/bin/ksh
# Merged Version: for Delimited and Non_delimited
function start_load
# Function to generate the FIFO files for data being transferred via ftp and
# for data being split
# and initiating the splitting and loading
# called as  start_load $table $cfg_file $dbload_file
#
{
 my_table=$1
 my_cfg_file=$2
 my_load_file=$3

 if (grep -i "¬ *OutputNodes" $my_cfg_file > /dev/null)
 then
  grep -i "¬ *OutputNodes * *=" $my_cfg_file| sed "s/OutputNodes *=
  *(//"|sed "s/) *$//"| awk 'BEGIN {FS = ","} { for
  (node=1; node <=NF; node++) printf "%d ", $node }'| read node_list
  for nodenum in $node_list
   do
    node_name= awk '{ if ($1 == '$nodenum') print $4}' $HOME/sqllib/db 2nodes.cfg
    #echo "The Node Name of $nodenum is $node_name"
    if ( test $nodenum -lt 10)
      then

      ...
```

Figure 115. Shell Script generate_fifo

To make AutoLoader use one or the other address change the parameter in the awk statement to either $4 or to $2:

```
node_name= awk '{ if ($1 == '$nodenum') print $4}' $HOME/sqllib/db2nodes.cfg
```

8.2 Loading Data without AutoLoader

If you don't want to use AutoLoader for populating your DB2 Parallel Edition database, you can use the standard DB2 functions db2split and db2load.

8.2.1 Function db2split

To run db2split issue the following command:

```
db2split -c <configfile>
```

Figure 116. Synatx of db2split

The <configfile> is exactly the same format and content as described in AutoLoader (see 8.1.2.2, “Modify *.cfg File” on page 215) For more information about db2split, refer to *DB2 ParallelEdition for AIX, Administration Guide and Reference*, SC09-1982.

If you intend to use db2split, make sure that there is enough space to store the partitioned files.

8.2.2 Function db2load

DB2 also offers the possibility to execute the load of the data with db2load function. Once the data is split to a file on the appropriate node you can run the load function of DB2. Below you will find an example script for loading the file ORDER.00001 into the table ORDERS on node 1.

```
db2 connect to tpcd
db2 "load from /db2piofs/data/order.00001 of del modified by
cordel| insert into orders"
```

Figure 117. Example: Load Command for ORDERS.00001

Notes

- Do not forget double quotes (") in the load statement
- Omit blanks after the parameter cordel

If you want to load data from a file only to a table that resides on one node. This table must belong to a single-node node group. To run the command successfully you have to modify your command from above by adding parameter noheader before the insert command. In this case db2load does not look for partitioning information in the file to be loaded.

Chapter 9. Using the Parallel I/O File System for Data Loading

This chapter provides a short overview about the capabilities of the IBM AIX Parallel I/O File System (PIOFS) and how to use those capabilities in a large database environment.

During the process to load data in large Decision Support or Data Warehouse environments, it might be a problem to handle files larger than 2 GB. The PIOFS offers the capability to handle files in sizes much larger than the current limit of AIX, which is 2 GB.

9.1 Overview of the Parallel I/O File System

The Parallel I/O File System (PIOFS) for the RISC/6000 SP is designed for serial or parallel applications that require large temporary files and high I/O bandwidth.

PIOFS lets you create files as large as 128 TB that might span multiple server nodes. A PIOFS file can be treated as a normal AIX file or it can be logically partitioned into subfiles, each containing a portion of the files data. Each subfile can then be processed in parallel by a separate task.

With PIOFS file partitioning, you can parallelize access to your data without the inconvenience and administrative overhead of maintaining multiple data files. PIOFS files can be dynamically partitioned into subfiles many different ways, all without altering or moving the contents of the file.

PIOFS supports parallelism in two complementary ways: physically and logically.

- A file can be divided physically over multiple disks and servers.
- A file can be divided logically into multiple subfiles.

Both the physical and logical division of the file contribute to the ability to process the file in parallel.

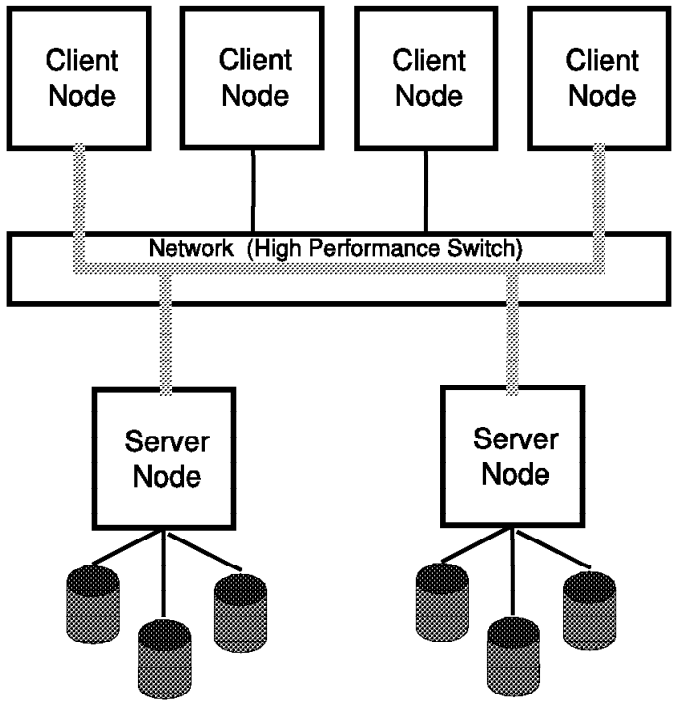
Another key benefit of PIOFS is its scalability. You can even achieve significant parallelism for your application just by using PIOFS to spread files across multiple disks and servers, without using file partitioning.

The Parallel I/O File System is designed as a client/server application. Server nodes provide file space for the PIOFS clients over the network. PIOFS kernel extensions are loaded into the AIX kernel within the virtual file system (VFS) layer. PIOFS is set up as a TCP/IP application and can be configured and used over any TCP/IP connection. However the best results will be achieved by using the PIOFS together with the High Performance Switch (HPS) of the RISC/6000 SP. The High Performance Switch connects a PIOFS client to the PIOFS servers over point-to-point, links providing the full bandwidth from each client to each server.

For more information about the PIOFS please refer to *IBM AIX Parallel I/O File System: Installation, Administration, and Use*, SH34-6065-00 and *IBM Systems Journal Vol. 34, No. 2, 1995 Parallel File Systems for the IBM SP Computers*, G321-0120-00.

9.2 Parallel File Access

Imagine your RISC/6000 SP as two client nodes connected through a network (the RISC/6000 SP High Performance Switch provides the most efficient access) to two server nodes. Your installation can actually have from one node up to the full complement provided by the RISC/6000 SP. Now imagine you want to store a large file. With the Parallel I/O File System you can have a single file spread across the two server nodes. Any one of the client nodes can access data at any of the server nodes as shown in Figure 118.



at two server nodes

Figure 118. The RISC/6000 SP with Several PIOFS Client Nodes Accessing Data. The Parallel I/O File System supports simultaneous access of server nodes by multiple client nodes.

In large database environments you can use your PIOFS files as normal AIX files. With this approach, you can immediately exploit several key PIOFS features. You will be able to do the following:

- Create files greater than 2 GB.
- Achieve moderate parallelism, and hence better I/O performance, by spreading files across multiple server nodes.
- Capture and save snapshots of PIOFS files during processing. If your program fails, you can use the checkpointed version of the file to restart the program where it was last checkpointed. You can also use file checkpointing to save a particular version of a PIOFS file.

9.2.1 Up to 128 TB File Support

With its own file offset function, the Parallel I/O File System supports files larger than 128 TB. The upper limit, subject to the block_limit set in your configuration file, is the combined space of all your disks.

The Parallel I/O File System is architected to support a file size of up to 2E63 bytes. The currently implemented limit, however, is a file size of (512 x 256 GB) = 128 GB.

9.2.1.1 Some Remarks

For files 2 GB or larger, you must define a profile to indicate that the files should be accessed using the PIOFS internal file offset.

Files can be treated as standard AIX files with the additional benefit that they can grow to the greater size supported by the Parallel I/O File System.

Files smaller than 2 GB can be copied between a Parallel I/O File System and another file system. You can also create new files.

9.2.2 Installation, Configuration and Use

This topic discusses our experience with the installation, configuration and usage of the PIOFS.

However, this is neither an installation nor a user's guide; it surely will not cover all aspects of the PIOFS. The intention is just to cover that part that is needed to use PIOFS in a large database environment. For detailed information about installation, administration, and use of PIOFS please refer to *IBM AIX Parallel I/O File System: Installation, Administration, and Use*, SH34-6065-00.

9.2.2.1 Installation

Installation of PIOFS on RISC/6000 SP and AIX is straightforward. All of the required information about disk space, product requirements and more can be found in the installp images. The installp command is used to install the Parallel I/O File System on the client and server nodes of your RISC/6000 SP.

9.2.2.2 Install Images

The Parallel I/O File System installp image consists of the following components:

1. **The client component**- Provides only the functions for PIOFS clients.
2. **The server component**- Provides the functions for PIOFS servers. It also includes the README file as well as the shell scripts to administer PIOFS.
3. **The common component**- Provides functions used by both the client and server parts of PIOFS. The common component is automatically installed when you install either the client or server component.
4. **The SMIT screens**- Allow you to use SMIT for PIOFS-related system management tasks. The SMIT screens require that the server component also be installed.

A separate installp image, piofs.doc contains the following PIOFS documentation:

- A softcopy file of the *IBM AIX Parallel I/O File System: Installation, Administration, and Use*, formatted for PostScript printing

- The InfoExplorer database for PIOFS
- PIOFS man pages

For the sake of simplicity, we installed all components on the two RISC/6000 SP nodes we intended to use as PIOFS server and client at the same time. We also installed all components on the control workstation.

The SMIT screens component for PIOFS contains the necessary PIOFS system management interfaces and SMIT menus to configure and administrate the PIOFS on the RISC/6000 SP.

9.2.2.3 Installation Procedure

PIOFS has to be installed on the file server nodes and on the nodes where you intend to mount a PIOFS file system (client nodes).

There are several ways to install software on the RISC/6000 SP. The most convenient way might be using the dsh command. You can set up a hostlist file that contains a list of the RISC/6000 SP nodes you intend to install the PIOFS client/server components. The name of the hostlist file is propagated to dsh using the environment variable WCOLL.

```
export WCOLL=/yourdir/hostlist
```

With this setup you are able to issue AIX commands in parallel to all the RISC/6000 SP nodes specified in the hostfile file.

For the installation, you should be root on the control workstation and authenticated as an administrative user to the RISC/6000 SP kerberos. The following example also assumes that the PIOFS installation image was copied to the directory /usr/sys/inst.images. The option used on the installp command below will commit the software without saving replaced files and automatically install prerequisite software.

```
# dsh
dsh> mount sp2cw0:/usr/sys/inst.images /mnt

The above command assumes that the directory /usr/sys/inst.images
is exported on the control workstation to be mountable on the
RISC/6000 SP nodes specified in the hostlist file.

dsh> installp -qacgNX -d /mnt piofs 1.1.0.0 all

Output of all the installp commands running on the
RISC/6000 SP nodes ..............

dsh> umount /mnt
```

You can verify the installation by using the lslpp command on the nodes.

```
# dsh lslpp -h piofs
```

License Keys: PIOFS uses iFOR/LS to regulate license use. Only servers are licensed. You need one license for each server node. Client nodes do not require licenses.

The following are the two options for iFOR/LS license keys:

- Nodelocked licenses on each server node
- Concurrent licenses served by an iFOR/LS license server

Nodelocked licenses do not require the installation of additional software on the node. Moreover, nodelocked licenses do not require an iFOR/LS server on the network. In case of nodelocked licenses the license keys are stored in a file on each node. The full path of the nodelock file is:

```
# /usr/lib/netls/conf/nodelock
```

This file has to contain the information (the license key) provided by the IBM license key center. To obtain the iFOR/LS license keys, the administrator must provide the CPU-IDs of all the nodes intended to run as PIOFS servers. The following command can be used:

```
# /usr/bin/uname -m

it returns the CPU-ID like
000009736700
```

Concurrent licenses can be shared among a cluster of hosts and allow more flexible administration of iFOR/LS licenses. However there must be an iFOR/LS server installed on the network. See the iFOR/LS manuals for more information about installation and configuration of NCS and iFOR/LS.

9.2.3 Configure the Parallel I/O File System

Files stored in a Parallel I/O File System are distributed across a set of server nodes (in our case we used two servers), each having one or more disks. Disks from the point of view of the PIOFS are AIX logical volumes. The Logical Volume Manager manages these logical volumes and hides the hardware characteristics of the underlying disks from the application. Before initializing a PIOFS, you need to specify the server nodes and the logical volumes to be used by configuring the PIOFS.

Planning Logical Volume Space Allocation: It is best to configure all server nodes with the same size and amount of logical volumes. A PIOFS uses about the same amount of logical volume space on each server. If the space allocated for the PIOFS at one server node becomes full, the PIOFS returns an out of space message even if space is available at other server nodes.

Note

The maximum logical volume size supported (or used) by the PIOFS is 2 GB.

9.2.4 Configuring the PIOFS Servers

Configuring the PIOFS server nodes requires the following steps:

1. Specify and create a configuration file.
2. Configure the PIOFS servers.
3. Format the servers.
4. Start the servers.

Note

It is recommended to do the following steps from the control workstation, either by using SMIT or the PIOFS command line interface.

9.2.4.1 Configuration File

The configuration file contains the addresses of the server nodes, network interfaces, and the size and number of logical volumes to be created and used.

The most straightforward way to create a configuration file is to copy the sample configuration file (sample.piofscfg) and name the copy myname.piofscfg. The sample file is located in the /usr/lpp/piofs directory. The PIOFS configuration file used to configure must be stored in the /var/piofs directory. To create a copy of this file, type:

```
cp /usr/lpp/piofs/sample.piofscfg /var/piofs/db2pe.piofscfg
```

Now use your favorite editor to modify/change the copied sample config file according to your needs. The following PIOFS configuration file defines two servers and eight 2 GB logical volumes per server.

```
# db2pe.piofscfg - This is the configuration file for the DB2PE
#                  parallel file system.
#
# The following table describes the valid formats.
#
#*****************************************************************************
#   block_limit number
#
#   block_limit  - is a keyword, it allows the administrator to specify the
#                  number of 64k data blocks that any one user can have on
#                  a single server node.
#       number   - is the number of blocks, 0< number <= 134217727.
#                  The default is unlimited, which can be specified by -1.
#*****************************************************************************
#   object_limit number
#
#       object_limit - is a keyword, it allows the administrator to specify the
#                      number of metadata objects that any one user can have on
#                      a single server node.
#       number       - is the number of objects, 0< number <= 134217727.
#                      The default is unlimited, which can be specified by -1.
#*****************************************************************************
#   IPports numbers
#
#       IPports      - is a keyword, it specifies the IP port numbers that will
#                      be used by the server nodes to communicate. The default
#                      is 9076 9077. Two ports are used for communications, one
#                      for sending information and the other for acknowledgements
#       numbers      - are the the IP port number to use.
#*****************************************************************************
#   network_interface_name name
#
#       network_interface_name  - is a keyword
#
```

```
#       name         - is the network interface name of the communications
#                      adapter that will be used for all the communications
#                      between severs and between clients and servers.
#                      The default is css0, the SP High Performance Switch.
#*******************************************************************************
#   udp_sendspace nnn
#
#       udp_sendspace  - is a keyword
#
#       nnn          - is the UDP socket send buffer size (in bytes) to be
#                      used by the Parallel I/O File System.  The recommended
#                      value is 131072 bytes.
#
#                      Note that if the udp_sendspace value is less than 65536,
#                      performance may suffer due to the larger volume of
#                      messages.
#*******************************************************************************
#   cient_udp_recvspace  nnn
#   server_udp_recvspace nnn
#
#       client_udp_recvspace  - is a keyword
#       server_udp_recvspace  - is a keyword
#
#       nnn          - is the UDP socket receive buffer size (in bytes) to be
#                      used by the Parallel I/O File System.  We recommend
#                      that you assign the values using the following formulas:
#
#                      client_udp_recvspace = 128k * (number of servers)
#                      server_udp_recvspace = 128k * (number of active clients)
#
#                      Note that if the UDP socket receive buffers are too
#                      small, messages will be dropped, requiring
#                      re-transmissions with resulting performance degradation.
#*******************************************************************************
#   storage_node lnode hostname IPaddr devnum lvsize volgrp pdisk
#
#       storage_node - is a keyword, it is used to create the underlying JFS
#                      filesystems that the parallel file system will use.
#       lnode        - the logical storage node number, these must start at 0
#                      and be in sequential order.
#       hostname     - the hostname that will be returned by the "hostname"
#                      command.
#       IPaddr       - the IP address to use for client-server communication,
#                      ie. the switch vs FCS vs E-net vs FDDI ...
#       devnum       - device number.  These represent the entities that PIOFS
#                      will stripe data across at each storage node.  A JFS
#                      logical volume will be created for each devnum as
#                      follows:
#                          /dev/pfsd1
#                          /dev/pfsd2
#                           .
#                           .
#                           .
#                          /dev/pfsmd   # logical volume for metadata
#                      The device numbers must start at 1 and increase
#                      sequentially to a maximum of 32.  There must also be a
#                      device specified as "metadata", which will hold the
#                      filesystem metadata.
```

```
#       lvsize      - size of the logical volume in physical partitions.
#                     The lvsize for metadata devices must be at least 28MB.
#                     The lvsize for all devices cannot exceed 2GB.
#       volgrp      - the volume group in which to create the logical volume.
#       pdisk       - physical disk on which to create the logical volume.
#                     You could create several logical volumes on one physical
#                     disk, but this would prevent you from getting multiple
#                     disk heads active at one time.
#
#*******************************************************************************
#
#
# below are the actual entries for our db2pe test system
# Commented lines (those that start with a #) are ignored.

block_limit     -1
object_limit    -1
IPports         9076 9077
network_interface_name css0
udp_sendspace   131072
client_udp_recvspace   262144     # 2 servers
server_udp_recvspace   262144    # 2 clients

#storage_node lnode hostname   IPaddr devnum   lvsize  volgrp    pdisk

storage_node    0    sp2n11  9.12.6.48   1        8   db2pevg11  hdisk5
storage_node    0    sp2n11  9.12.6.48   2        8   db2pevg11  hdisk5
storage_node    0    sp2n11  9.12.6.48   3        8   db2pevg11  hdisk6
storage_node    0    sp2n11  9.12.6.48   4        8   db2pevg11  hdisk6
storage_node    0    sp2n11  9.12.6.48   5        8   db2pevg11  hdisk6
storage_node    0    sp2n11  9.12.6.48   6        8   db2pevg11  hdisk5
storage_node    0    sp2n11  9.12.6.48   7        8   db2pevg11  hdisk5
storage_node    0    sp2n11  9.12.6.48   8        8   db2pevg11  hdisk5
storage_node    0    sp2n11  9.12.6.48   metadata 1   db2pevg11  hdisk5

storage_node    1    sp2n12  9.12.6.49   1        8   db2pevg12  hdisk2
storage_node    1    sp2n12  9.12.6.49   2        8   db2pevg12  hdisk2
storage_node    1    sp2n12  9.12.6.49   3        8   db2pevg12  hdisk3
storage_node    1    sp2n12  9.12.6.49   4        8   db2pevg12  hdisk3
storage_node    1    sp2n12  9.12.6.49   5        8   db2pevg12  hdisk3
storage_node    1    sp2n12  9.12.6.49   6        8   db2pevg12  hdisk4
storage_node    1    sp2n12  9.12.6.49   7        8   db2pevg12  hdisk4
storage_node    1    sp2n12  9.12.6.49   8        8   db2pevg12  hdisk4
storage_node    1    sp2n12  9.12.6.49   metadata 1   db2pevg12  hdisk2
```

9.2.4.2 Configure the Parallel I/O File System

Configuration creates logical volumes on each disk on each server node specified in the configuration file. It also verifies the syntax of the configuration file, copies the configuration file from the control workstation to the servers, and updates the System Data Repository (SDR).

You can configure a PIOFS, format the servers, and start the servers, all using SMIT. Alternatively, you can issue commands to perform these steps. If you use SMIT, go to the PIOFS menu as shown below and use it to do the following:

1. Configure the Parallel I/O File System
2. Format the servers
3. Start the servers

Then go directly to 9.2.5.1, “Mount the Parallel I/O File System on Client Nodes” on page 234. If you use the command interface, perform each step in sequence.

9.2.4.3 Using SMIT

The SMIT menu for all PIOFS operations can be reached from the SMIT main menu following this path:

```
# smit
  Physical & logical Storage
  File Systems
  Add/Change/Show/Delete File Systems
  Parallel I/O File System
```

9.2.4.4 Using the piofs_config Command

Run this command as root from the control workstation.

```
/usr/lpp/piofs/bin/piofs_config /var/piofs/db2pe.piofscfg
```

The piofs_config command creates the required logical volumes, file systems on each server node and mounts the JFS file systems used for storing PIOFS files and metadata. The file systems are named /piofsdev1 ... /piofsdevn and /piofsmdata.

9.2.4.5 Format the PIOFS Servers

Formatting cleans up the PIOFS file space created during the previous steps. You do this step *only* after configuring a Parallel I/O File System.

Attention

Formatting destroys all the files on the target logical volume.

Type:

```
/usr/lpp/piofs/bin/format_piofs_server db2pe.piofscfg
```

Specify only the configuration file name, not the full path name.

9.2.5 Start the PIOFS Servers

Type:

```
/usr/lpp/piofs/bin/start_piofs_server db2pe.piofscfg
```

Specify only the configuration file name, not the full path name.

After successfully starting all file server nodes, the Parallel I/O File System is set up completely and can now be mounted by clients.

9.2.5.1 Mount the Parallel I/O File System on Client Nodes

Before the PIOFS can be used as an additional AIX file system, it has to be mounted from the servers to clients. Although the Parallel I/O File System is a kind of network file system, the way to mount a PIOFS is different from NFS.

Each Parallel I/O File System client node must have a copy of the active configuration files to be able to successfully mount the PIOFS. The mount process loads also the Parallel I/O File System kernel extensions. Please remember that on the nodes you want to use as PIOFS clients, the client component and the common component of the PIOFS have to be installed.

To mount a Parallel I/O File System, log in on the client using the root ID and do the following steps:

1. Find the active configuration file in /etc on any configured server. The names are:

   /etc/config_file_name..active

   /etc/config_file_name **.IPaddresses.active**

   This is the configuration file you created with .active and .IPaddresses.active appended to it.

   Copy the active configuration file into the /etc on each of the clients.

   If your RISC/6000 SP has an active System Data Repository (SDR), you can omit this step. The file will be automatically retrieved and stored in the /etc directory.

2. Use the crvfs command to add the following line to /etc/vfs:

   ```
   piofs nn /usr/lpp/piofs/bin/piofsmnthelp none
   ```

   Replace nn with a number greater than 8 and less than 12 that is not used for any other file system type at your installation.

   To do this, type:

   ```
   crvfs "piofs:10:/usr/lpp/piofs/bin/piofsmnthelp:none"
   ```

3. Use the crfs command to add the following stanza to /etc/filesystems on all the client nodes where you intent to mount the Parallel I/O File System:

   ```
   /db2pefiofs:
         dev         = /piofs
         type        = piofs
         vfs         = piofs
         mount       = false
         options     = db2pe.piofscfg  # This is the active PIOFS cofigfile
         nodename    = piofs
   ```

   The first line of the above stanza is the directory the file system is going to be mounted on.

   To create the previous stanza in your /etc/filesystems, type

   ```
   crfs -v piofs -u piofs -d /piofs -m /db2pepiofs -a options=db2pe.piofscfg -n piofs
   ```

4. Create a mount point (directory) on each client the PIOFS is going to be mounted on. Remember that you can use the RISC/6000 SP command to distribute the following command to the nodes.

   ```
   mkdir /db2pepiofs
   ```

This is the mount point you specified for the -m option in the crfs command under the previous step 3.

5. Mount the file system. Using the information in /etc/filesystems you can enter:

```
mount /db2pepiofs
or
mount -t piofs
```

If the Parallel I/O File System is mounted only temporarily, there is no need to add an entry to /etc/filesystems. All necessary information can be passed with the mount command.

```
mount -n piofs -o db2pe.piofscfg -v piofs /piofs /db2pepiofs
```

9.2.5.2 Authorize Users

Users other than root must be individually authorized to create files within a Parallel I/O File System. This can be done by using the SMIT PIOFS interface or via PIOFS command add_piofs_user. Users not authorized this way cannot create new files in the file system, but, given sufficient standard AIX file permissions, they can read and write existing files.

To create directories for authorized users, use the AIX mkdir command.

9.2.5.3 The Parallel I/O File System Profile

The way a file is stored on the server nodes is highly configurable. PIOFS uses, for all control options, default values. These default values will split large files into subfiles and store them on the server nodes. To overwrite this defaults there is a hierarchy of profiles. The way data is placed on the server nodes can be influenced on file, user or file basis.

The system uses the first piofs.profile file found by searching as follows:

1. $HOME/file_name.piofs.profile

   This is the local default to be applied to file_name when you access that file.

2. $HOME/piofs.profile

   This is the local default to be applied to all files for this user.

3. /usr/lpp/piofs.profile

   This is default to be applied to all files on the client node.

A Sample piofs.profile File

```
# Profile for Parallel I/O File System.
# It gets read during file create and open time. The following
# types of entries are supported:
# 1) # comment
# 2) BSU               Specifies the basic striping unit in bytes.
#                      The default is -1 = 32768 bytes.
#                      The maximum is 2147483647 bytes.
# 3) CELLS             Specifies the number of cells to use when
#                      when creating the file. The default of -1
#                      will create one cell per server node.
# 4) USE_PIOFS_LSEEK   Indicates that 64-bit offsets are to be used
#                      instead of the AIX (32-bit) file offset.
# 5) CAUTIOUS          Indicates that the file system should force
#                      sequential atomicity.
```

```
# 6) BASE_NODE         Specifies the node at which cell 0 is to be
#                      located. It is recommended to use the default
#                      of -1, which is pseudo-random base node
#                      selection.
#
#
BSU                32768
CELLS              -1
USE_PIOFS_LSEEK
CAUTIOUS
BASE_NODE          -1
```

9.2.5.4 Parallel I/O File System Usage

The Parallel I/O File System is of great help in large DB2 Parallel Edition environments. It allows for greater flexibility in using disk space and does not limit files to the 2 GB size. Since Parallel I/O File System is implemented as a distributed file system the PIOFS can be made available on all nodes in the DB2 Parallel Edition environment. That means that in case of loading the DB2 Parallel Edition database no files have to be copied or transferred from the db2split process to the load running on the individual nodes. After the PIOFS is mounted on the DB2 Parallel Edition nodes, the load process can access the splitted files like a local file.

Another possibility of using PIOFS is to gather unused disk space. When the first load process for DB2 Parallel Edition is started it can well be that not all the disk space dedicated to the nodes is used up. With the help of PIOFS it is possible to combine, lets say in a 32 node RISC/6000 SP, 2 GB of disk space per node into a 64 GB file system and make this large disk space available on every node. This will allow for great flexibility during the load process.

The other extreme would be to have a file server node with a lot of disks, define a PIOFS on this node and make the diskspace available on all other nodes. The same is possible with NFS, however there are some disadvantages in using NFS:

- No files larger 2 GB yet
- Slower than PIOFS
- Higher CPU utilization than PIOFS

Using the Parallel I/O File System untuned, just out of the box, gave us a read/write speed of about 4 MB/sec. This is equivalent to read or write to a local disk.

Status information about the PIOFS can be obtained by the following command:

```
stat_piofs_server -l mount_point

stat_piofs_server -l /db2piofs
```

9.3 User Data

All tests in this book are based on the TPC-D database. The TPC Benchmark D is a decision support benchmark. It consists of a set of business oriented queries and a database that is populated with data that has an industry-wide relevance in this area. The database has 35 GB raw data and consists of the following DB2 tables shown in Table 14.

Table 14. Tables in the TPC-D Database

Tables	Columns	Rows	Size
NATION	N_NATIONKEY INTEGER NOT NULL N_NAME CHAR(25) NOT NULL N_REGIONKEY INTEGER NOT NULL N_COMMENT VARCHAR(152)	25	4 KB
REGION	R_REGIONKEY INTEGER NOT NULL R_NAME CHAR(25) NOT NULL R_COMMENT VARCHAR(152)	5	0.5 KB
PART	P_PARTKEY INTEGER NOT NULL P_NAME VARCHAR(55) NOT NULL P_MFGR CHAR(25) NOT NULL P_BRAND CHAR(10) NOT NULL P_TYPE VARCHAR(25) NOT NULL P_SIZE INTEGER NOT NULL P_CONTAINER CHAR(10) NOT NULL P_RETAILPRICE DECIMAL(12,2) NOT NULL P_COMMENT VARCHAR(23) NOT NULL		
SUPPLIER	S_SUPPKEY INTEGER NOT NULL S_NAME CHAR(25) NOT NULL S_ADDRESS VARCHAR(40) NOT NULL S_NATIONKEY INTEGER NOT NULL S_PHONE CHAR(15) NOT NULL S_ACCTBAL DECIMAL(12,2) NOT NULL S_COMMENT VARCHAR(101) NOT NULL)	6'400'000	
PARTSUPP	PS_PARTKEY INTEGER NOT NULL PS_SUPPKEY INTEGER NOT NULL PS_AVAILQTY INTEGER NOT NULL PS_SUPPLYCOST DECIMAL(12,2) NOT NULL PS_COMMENT VARCHAR(199) NOT NULL)		
CUSTOMER	C_CUSTKEY INTEGER NOT NULL C_NAME VARCHAR(25) NOT NULL C_ADDRESS VARCHAR(40) NOT NULL C_NATIONKEY INTEGER NOT NULL C_PHONE CHAR(15) NOT NULL C_ACCTBAL DECIMAL(12,2) NOT NULL C_MKTSEGMENT CHAR(10) NOT NULL C_COMMENT VARCHAR(117) NOT NULL)		
ORDERS	O_ORDERKEY INTEGER NOT NULL O_CUSTKEY INTEGER NOT NULL O_ORDERSTATUS CHAR(1) NOT NULL O_TOTALPRICE DECIMAL(12,2) NOT NULL O_ORDERDATE DATE NOT NULL O_ORDERPRIORITY CHAR(15) NOT NULL O_CLERK CHAR(15) NOT NULL O_SHIPPRIORITY INTEGER NOT NULL O_COMMENT VARCHAR(79) NOT NULL)	1'200'000	
LINEITEM	L_ORDERKEY INTEGER NOT NULL L_PARTKEY INTEGER NOT NULL L_SUPPKEY INTEGER NOT NULL L_LINENUMBER INTEGER NOT NULL L_QUANTITY DECIMAL(12,2) NOT NULL L_EXTENDEDPRICE DECIMAL(12,2) NOT NULL L_DISCOUNT DECIMAL(12,2) NOT NULL L_TAX DECIMAL(12,2) NOT NULL L_RETURNFLAG CHAR(1) NOT NULL L_LINESTATUS CHAR(1) NOT NULL L_SHIPDATE DATE NOT NULL L_COMMITDATE DATE NOT NULL L_RECEIPTDATE DATE NOT NULL L_SHIPINSTRUCT CHAR(25) NOT NULL L_SHIPMODE CHAR(10) NOT NULL L_COMMENT VARCHAR(44) NOT NULL)	48'000'000	

9.4 Data Definition for Tables in TPCD

Figure 119 depicts the structure of the definition of data in TPCD table. The first line in the table layout is a statement indicating a db2 connection to tpcd and this is followed by a series of create operations in the table.

```
db2 connect to tpcd;
db2 "CREATE TABLE NATION   ( N_NATIONKEY   INTEGER NOT NULL, \
                             N_NAME        CHAR(25) NOT NULL, \
                             N_REGIONKEY   INTEGER NOT NULL, \
                             N_COMMENT     VARCHAR(152)) \
                             PARTITIONING KEY(N_NATIONKEY) USING HASHING"
db2 "CREATE TABLE REGION   ( R_REGIONKEY   INTEGER NOT NULL,\
                             R_NAME        CHAR(25) NOT NULL, \
                             R_COMMENT     VARCHAR(152)) \
                             PARTITIONING KEY(R_REGIONKEY) USING HASHING"
db2 "CREATE TABLE PART     ( P_PARTKEY     INTEGER NOT NULL, \
                             P_NAME        VARCHAR(55) NOT NULL, \
                             P_MFGR        CHAR(25) NOT NULL, \
                             P_BRAND       CHAR(10) NOT NULL, \
                             P_TYPE        VARCHAR(25) NOT NULL, \
                             P_SIZE        INTEGER NOT NULL, \
                             P_CONTAINER   CHAR(10) NOT NULL, \
                             P_RETAILPRICE DECIMAL(12,2) NOT NULL, \
                             P_COMMENT     VARCHAR(23) NOT NULL) \
                             PARTITIONING KEY(P_PARTKEY) USING HASHING"
db2 "CREATE TABLE SUPPLIER ( S_SUPPKEY     INTEGER NOT NULL, \
                             S_NAME        CHAR(25) NOT NULL, \
                             S_ADDRESS     VARCHAR(40) NOT NULL, \
                             S_NATIONKEY   INTEGER NOT NULL, \
                             S_PHONE       CHAR(15) NOT NULL, \
                             S_ACCTBAL     DECIMAL(12,2) NOT NULL, \
                             S_COMMENT     VARCHAR(101) NOT NULL) \
                             PARTITIONING KEY(S_SUPPKEY) USING HASHING"
db2 "CREATE TABLE PARTSUPP ( PS_PARTKEY    INTEGER NOT NULL, \
                             PS_SUPPKEY    INTEGER NOT NULL, \
                             PS_AVAILQTY   INTEGER NOT NULL, \
                             PS_SUPPLYCOST DECIMAL(12,2)   NOT NULL, \
                             PS_COMMENT    VARCHAR(199) NOT NULL) \
                             PARTITIONING KEY(PS_PARTKEY) USING HASHING"
```

Figure 119 (Part 1 of 2). Data Definition Table

```
db2 "CREATE TABLE CUSTOMER ( C_CUSTKEY     INTEGER NOT NULL, \
                             C_NAME        VARCHAR(25) NOT NULL, \
                             C_ADDRESS     VARCHAR(40) NOT NULL, \
                             C_NATIONKEY   INTEGER NOT NULL, \
                             C_PHONE       CHAR(15) NOT NULL, \
                             C_ACCTBAL     DECIMAL(12,2)    NOT NULL, \
                             C_MKTSEGMENT  CHAR(10) NOT NULL, \
                             C_COMMENT     VARCHAR(117) NOT NULL) \
                             PARTITIONING KEY(C_CUSTKEY) USING HASHING"
db2 "CREATE TABLE ORDERS   ( O_ORDERKEY    INTEGER NOT NULL, \
                             O_CUSTKEY     INTEGER NOT NULL, \
                             O_ORDERSTATUS CHAR(1) NOT NULL, \
                             O_TOTALPRICE  DECIMAL(12,2) NOT NULL, \
                             O_ORDERDATE   DATE NOT NULL, \
                             O_ORDERPRIORITY  CHAR(15) NOT NULL, \
                             O_CLERK       CHAR(15) NOT NULL, \
                             O_SHIPPRIORITY  INTEGER NOT NULL, \
                             O_COMMENT     VARCHAR(79) NOT NULL) \
                             PARTITIONING KEY(O_ORDERKEY) USING HASHING"
db2 "CREATE TABLE LINEITEM ( L_ORDERKEY    INTEGER NOT NULL, \
                             L_PARTKEY     INTEGER NOT NULL, \
                             L_SUPPKEY     INTEGER NOT NULL, \
                             L_LINENUMBER  INTEGER NOT NULL, \
                             L_QUANTITY    DECIMAL(12,2) NOT NULL, \
                             L_EXTENDEDPRICE  DECIMAL(12,2) NOT NULL, \
                             L_DISCOUNT    DECIMAL(12,2) NOT NULL, \
                             L_TAX         DECIMAL(12,2) NOT NULL, \
                             L_RETURNFLAG  CHAR(1) NOT NULL, \
                             L_LINESTATUS  CHAR(1) NOT NULL, \
                             L_SHIPDATE    DATE NOT NULL, \
                             L_COMMITDATE  DATE NOT NULL, \
                             L_RECEIPTDATE DATE NOT NULL, \
                             L_SHIPINSTRUCT CHAR(25) NOT NULL, \
                             L_SHIPMODE    CHAR(10) NOT NULL, \
                             L_COMMENT     VARCHAR(44) NOT NULL) \
                             PARTITIONING KEY(L_ORDERKEY) USING HASHING"
```

Figure 119 (Part 2 of 2). Data Definition Table

Appendix A. How to Manage the DB2 Parallel Edition Database Segment Directory File Systems

A.1 Introduction

This section describes how to manually:

- Mount AIX JFS File Systems over database segment directories.
- Increase the size of a mounted database segment directory file system.
- Clean-up the database segment directory file systems after a drop database.

A.2 Procedures

Attention

The database should *not* be in use when the mount process is attempted since and/or data corruption may occur.

Notes:

1. In the following procedures, we assume that SQL00001 contains objects associated with database SAMPLE, and that we are mounting the SAMPLE database segment directory 16, that is SQLS0016. The new temporary file system directory mount point is /tmp/SQLS0016. Any valid directory could be used instead of /tmp/SQLS0016
2. The full database directory path in the examples is given by /u/use rid/db2instance/SQL00001 where db2instance is the name of the instance to which the database belongs. The leading path to the SQL00001 directory could be different in another environment.

A.2.1 Creating and Mounting AIX File System over Database Segment Directories

The procedure for manually mounting AIX JFS File Systems over database segment directories is the following:

- Create the new database - let's say the database name is SAMPLE.
- Create all the tables in the database SAMPLE.
- If you do not wish to mount a new file system over each database segment directory, then find out which segments will be used by a specific table (otherwise go to step 4):
  - db2start
  - db2 connect to SAMPLE
  - db2 select name,fid from sysibm.systables

```
> record the fid corresponding to the table name - let's say
  that the fid for table name TABLE5 is 48.

> calculate which segment number TABLE5 will start into:

  first_segment# = fid MODULO NUMSEGS
                 =  48 MODULO 32
                 =  16
```

– db2 terminate

– db2stop

- Create and mount a new file system over a database segment directory:

  1. Log on as ROOT.

     Repeat steps 2 to 11 for each database segment directory and/or database log directory.

  2. Create a new file system:

     ```
     crfs -v jfs -g rootvg -a size=1000 -m /tmp/SQLS0016 -A yes -p rw -t no

     Option: -v specifies the JFS virtual file system type
             -g  specifies the volume group
             -a  specifies the size of the new file system in
                 512-bytes block
             -m  specifies the temporary mount point.
             -A  specifies that the file system must be
                 automatically mounted at system restart
             -p  Sets the read-write permissions for the file
                 system
             -t  Sets NO accounting entry in the /etc/filesystems
     ```

  3. Mount the new file system:

     ```
     mount /tmp/SQLS0016
     ```

  4. Copy the content of the database segment directory to the new file system:

     ```
     cp /u/userid/db2instance/SQL00001/SQLS0016/* /tmp/SQLS0016/.
     ```

  5. Verify that the directory copy of the database segment directory to the new file system was successful:

     ```
     diff /u/userid/db2instance/SQL00001/SQLS0016 /tmp/SQLS0016
     ```

  6. Remove the content of the original database segment directory:

     ```
     rm -rf /u/userid/db2instance/SQL00001/SQLS0016/*

     Option: -f  Does not prompt before removing a write-
                 protected file.
             -r  Permits recursive removal of directories and
                 their contents.
     ```

7. Unmount the new file system before changing its mount point to the database segment directory:

```
umount /tmp/SQLS0016
```

8. Change new the file system mount point to the database segment directory:

```
chfs -m /u/userid/db2instance/SQL00001/SQLS0016 -A yes /tmp/SQLS0016

Option: -m  specifies new mount point
        -A  specifies that the file system must be
            automatically mounted at system restart.
```

9. Mount the file system at the new mount point:

```
mount /u/userid/db2instance/SQL00001/SQLS0016
```

10. Remove the temporary mount point:

```
rmdir /tmp/SQLS0016
```

11. Change the file and directory permissions of the new database segment directory file system:

    Refer to the original database segment directory permissions to look up db2instance and db2group. In this example, db2instance is the directory owner and db2group the directory group.

```
chown -R db2instance /u/userid/db2instance/SQL00001/SQLS0016

chgrp -R db2group /u/userid/db2instance/SQL00001/SQLS0016

chmod u=rwx,g=rx,o= /u/userid/db2instance/SQL00001/SQLS0016

Option: -R    Descends directories recursively, modifying the
              permissions for each file
        u=rwx allows read/write/execute permissions
              to the owner of the directory
        g=rx  allows read/execute permissions to the group
        o=    grants no permissions to the owner
```

12. Log off as ROOT.
13. The database can now be accessed as usual:

```
db2start

db2 connect to sample
```

The database segment directory file system will be remounted automatically at system restart.

A.2.2 Increasing the Size of a Database Segment Directory File System

The database segment directory file systems size can be increased by entering the following AIX command:

```
chfs -a size=2000 /u/userid/db2instance/SQL00001/SQLS0016

Option: -a specifies the size attribute in 512-byte blocks
```

This command may be invoked while the database is being accessed since AIX allows the online extension of JFS file systems.

Note that if the new file system size is not evenly divisible by the physical partition size, then the new file system size is rounded up to the closest number that is evenly divisible by the physical partition size.

A.2.3 Cleaning Up the Database Directory after a Drop Database

The drop database command does not deallocate the database segment directories that were mounted on a different file system. Once the Drop Database command has been issued, the following steps must be taken to remove the segment directory file systems:

1. Unmount the database segment directory file system:

   ```
   unmount /u/userid/db2instance/SQL00001/SQLS0016
   ```

2. Remove the database segment directory file system:

   ```
   rmfs -r /u/userid/db2instance/SQL00001/SQLS0016

   Option: -r  Permits recursive removal of the
               directory and its contents.
   ```

   This command removes the /u/userid/db2instance/SQL00001/SQLS0016 file system, its entry in the /etc/filesystems file, and the underlying logical volume.

3. Repeat steps 1 and 2 for all mounted database segment directories (including the database log directory SQLOGDIR if mounted).

4. Remove the SQLnnnnn directory:

   ```
   rm -fr /u/userid/db2instance/SQL00001

   Option: -f  Does not prompt before removing a write-
               protected file.
           -r  Permits recursive removal of directories and
               their contents.
   ```

A.3 Using the AIX SMIT Interface

Some of the procedures described in the previous sections can be done using SMIT.

Invoke SMIT by entering one of the following on the command line:

- smitty invokes the TTY or curses (character-based) version
- smit -C same as smitty
- smit invokes the MOTIF version if X-windows is running otherwise the character-based version is run.

To Create a New JFS File System:

1. Select **System Storage Management (Physical & Logical Storage)** .
2. Select **File Systems**.
3. Select **Add, Change, Show or Delete File Systems**.
4. Select **Journaled File Systems**.
5. Select **Add a Journaled File System**, and choose the following options as shown in Figure 120.

```
                        Add a Journaled File System

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                [Entry Fields]
    Volume group name                             rootvg
    * SIZE of file system (in 512-byte blocks)    [1000]
    * MOUNT POINT                                 [/tmp/SQLS0016]
    Mount AUTOMATICALLY at system restart?        yes
    PERMISSIONS                                   read/write
    Mount OPTIONS                                 []
    Start Disk Accounting?                        no


F1=Help         F2=Refresh       F3=Cancel       F4=List
F5=Undo         F6=Command       F7=Edit         F8=Image
F9=Shell        F10=Exit         Enter=Do
```

1

Figure 120. Adding a JFS

To Mount the New JFS File System:

1. Select **System Storage Management (Physical & Logical Storage)** .
2. Select **File Systems**.
3. Select **Mount a File System**, and choose the following options as shown in Figure 121 on page 246.

```
                        Mount a File System

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
    FILE SYSTEM name                                 [/tmp/SQLS0016]
    DIRECTORY over which to mount                    []
    TYPE of file system
    FORCE the mount?                                 no
    REMOTE NODE containing the file system           []
      to mount
    Mount as a REMOVABLE file system?                no
    Mount as a READ-ONLY system?                     no
    Disallow DEVICE access via this mount?           no
    Disallow execution of SUID and sgid programs     no
      in this file system?

F1=Help         F2=Refresh        F3=Cancel       F4=List
F5=Undo         F6=Command        F7=Edit         F8=Image
F9=Shell        F10=Exit          Enter=Do
```

Figure 121. Mount a JFS

To Unmount the New File System from Temporary Mount Point:

1. Select **System Storage Management (Physical & Logical Storage)** .
2. Select **File Systems**.
3. Select **Unmount a File System**, and choose the following options as shown in Figure 122.

```
                        Unmount a File System

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                        [Entry Fields]
    Unmount ALL mounted file systems?                  no
      (except /, /tmp, /usr)
             -OR-
    Unmount all REMOTELY mounted file systems?         no

    NAME of file system to unmount                     [/tmp/SQLS0016]
    REMOTE NODE containing the file system(s)          []
      to unmount

F1=Help         F2=Refresh         F3=Cancel        F4=List
F5=Undo         F6=Command         F7=Edit          F8=Image
F9=Shell        F10=Exit           Enter=Do
```

Figure 122. Unmount a File System

To Change the New File System Mount Point to the Database Segment Directory:

1. Select **System Storage Management (Physical & Logical Storage)** .
2. Select **File Systems**.
3. Select **Add / Change / Show / Delete File Systems**.

4. Select **Journaled File Systems**.
5. Select **Change / Show Characteristics of a Journaled File System**, and choose the following options as shown in Figure 123.

```
        Change / Show Characteristics of a Journaled File System

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                              [Entry Fields]
    File system name                         /tmp/SQLS0016
    NEW mount point                          [/u/userid/db2instance/SQL00001/SQLS0016]
    SIZE of file system (in 512-byte blocks) [8192]
    Mount GROUP                              []
    Mount AUTOMATICALLY at system restart?   yes
    PERMISSIONS                              read/write
    Mount OPTIONS                            []
    Start Disk Accounting?                   no


F1=Help         F2=Refresh        F3=Cancel       F4=List
F5=Undo         F6=Command        F7=Edit         F8=Image
F9=Shell        F10=Exit          Enter=Do
```

Figure 123. Changing a JFS

To Mount the JFS File System at the Database Segment Directory Mountpoint:

1. Select **System Storage Management (Physical & Logical Storage)** .
2. Select **File Systems**.
3. Select **Mount a File System**, and choose the following options as shown in Figure 124.

```
                        Mount a File System

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                 [Entry Fields]
    FILE SYSTEM name                            [/u/userid/db2instance/SQL00001/SQLS0016]
    DIRECTORY over which to mount               []
    TYPE of file system
    FORCE the mount?                            no
    REMOTE NODE containing the file system      []
      to mount
    Mount as a REMOVABLE file system?           no
    Mount as a READ-ONLY system?                no
    Disallow DEVICE access via this mount?      no
    Disallow execution of SUID and sgid programs  no
      in this file system?

F1=Help         F2=Refresh        F3=Cancel       F4=List
F5=Undo         F6=Command        F7=Edit         F8=Image
F9=Shell        F10=Exit          Enter=Do
```

Figure 124. Mounting a JFS

To Unmount the Database Segment Directory File System:

1. Select **System Storage Management (Physical & Logical Storage)** .
2. Select **File Systems**.
3. Select **Unmount a File System**, and choose the following options as shown in Figure 125.

```
                        Unmount a File System

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                    [Entry Fields]
    Unmount ALL mounted file systems?              no
      (except /, /tmp, /usr)
             -OR-
    Unmount all REMOTELY mounted file systems?     no

    NAME of file system to unmount                 [/u/userid/db2instance/SQL00001/SQLS0016]
    REMOTE NODE containing the file system(s)      []
      to unmount

F1=Help          F2=Refresh          F3=Cancel        F4=List
F5=Undo          F6=Command          F7=Edit          F8=Image
F9=Shell         F10=Exit            Enter=Do
```

Figure 125. Unmounting a JFS

To Increase the Size of a Database Segment Directory File System:

1. Select **System Storage Management (Physical & Logical Storage)** .
2. Select **File Systems**.
3. Select **Add / Change / Show / Delete File Systems**
4. Select **Journaled File Systems**.
5. Select **Change / Show Characteristics of a Journaled File System**, and enter the new size of the file system (in 512-byte blocks) as shown in Figure 126 on page 249.

```
        Change / Show Characteristics of a Journaled File System

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                              [Entry Fields]
    File system name                        /u/userid/db2instance/SQL00001/SQLS0016
    NEW mount point                         [/u/userid/db2instance/SQL00001/SQLS0016]
    SIZE of file system (in 512-byte blocks) [100000]
    Mount GROUP                             []
    Mount AUTOMATICALLY at system restart?  yes
    PERMISSIONS                             read/write
    Mount OPTIONS                           []
    Start Disk Accounting?                  no


F1=Help         F2=Refresh        F3=Cancel       F4=List
F5=Undo         F6=Command        F7=Edit         F8=Image
F9=Shell        F10=Exit          Enter=Do
```

Figure 126. Changing a JFS

To Remove the Database Segment Directory File System:

1. Select **System Storage Management (Physical & Logical Storage)** .
2. Select **File Systems**.
3. Select **Add / Change / Show / Delete File Systems**.
4. Select **Journaled File Systems**.
5. Select **Remove a Journaled File System**, and choose the following options as shown in Figure 127.

```
                        Remove a Journaled File System

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                              [Entry Fields]
    File system name                        /u/userid/db2instance/SQL00001/SQLS0016
    Remove Mount Point                      yes



F1=Help         F2=Refresh        F3=Cancel       F4=List
F5=Undo         F6=Command        F7=Edit         F8=Image
F9=Shell        F10=Exit          Enter=Do
```

Figure 127. Removing a JFS

A.4 db2sgmgr - Segment Manager Tool

The segment manager tool (db2sgmgr) is a front end utility which DB2 Parallel Edition database system administrators (SYSADMs) can use to map the segment subdirectories to separate file systems. This command line application is backed up by AIX System Management Interface Tool (SMIT) panels to provide an interface for the following:

- Querying the mount status and file system size of a database's segment directories.
- Creating new file systems in which to store the database files.
- Extending the size of existing database segment file systems.
- Cleaning up the database segment directories after a drop database.

Attention

Concurrent execution of the segment manager tool by multiple DB2 Parallel Edition instances is not recommended. AIX errors will result from simultaneously updating AIX file systems and volume group information.

Invocation

The db2sgmgr utility can be used with or without the optional SMIT interface to manage the database segment directory file systems. Only a user with SYSADM authority for the database manager instance can run the db2sgmgr against database maintained within that instance. The db2sgmgr utility must run as setuid-root. The owner of the program must be root and the db2sgmgr permissions should be set as ug+sx on installation. This utility can only be run against local databases which are cataloged in the system database directory.

```
ls -la db2sgmgr

-rwsr-sr-x 1 root system 51968 Aug 18 10:13 db2sgmgr
```

Interactions

The command syntax will be given by entering db2sgmgr without any parameter:

```
db2sgmgr

Command Syntax:

  db2sgmgr -lsdb
  db2sgmgr -lsmnt dbalias
  db2sgmgr -clndb dbdirectory
  db2sgmgr -chmnt dbalias start_seg# end_seg# new_size
  db2sgmgr -mnt dbalias start_seg# end_seg# volume_grp size [CONNECT|NOCONNECT]
```

Using the SMIT Interface

Extension definitions are also provided for the AIX System Management Interface Tool (SMIT). To access the db2sgmgr commands through SMIT do the following:

1. Invoke SMIT by entering one of the following on the command line:
   - smitty invokes the TTY or curses (character-based) version
   - smit -C same as "smitty"
   - smit invokes the MOTIF version if X-windows is running otherwise the character-based version is run.

2. Select the **Applications** menu option as shown in the following example in Figure 128 on page 251.

```
                              System Management

Move cursor to desired item and press Enter.

  Software Installation and Maintenance
  Devices
  System Storage Management (Physical & Logical Storage)
  Security & Users
  Communications Applications and Services
  Print Spooling
  Problem Determination
  Performance & Resource Scheduling
  System Environments
  Processes & Subsystems
  Applications
  Using SMIT (information only)

F1=Help         F2=Refresh         F3=Cancel       F8=Image
F9=Shell        F10=Exit           Enter=Do
```

Figure 128. System Management Menu

Help

DB2 Parallel Edition Segment Manager Help will be provided through the SMIT HELP key (F1).

The DB2 Parallel Edition Segment Manager Tool commands are described in detail in *DATABASE2 AIX/6000, Command Reference, SC09-1575-00*.

Error Logging

AIX errors received during the execution of the segment manager tool are logged in sqllib/db2dump/db2sgmgr.

Example: The segment manager log records the following type of information:

```
Mon Sep 27 14:13:01 EDT 1995
----------------------------
db2sgmgr -chmnt sample 0 0 512
Filesystem size changed to 98304

Mon Sep 27 14:14:44 EDT 1995
----------------------------
db2sgmgr -chmnt sample 1 3 512
0516-404 allocp: Not enough resources available to fulfill
         allocation. Either not enough free partitions or not enough
         physical volumes to keep strictness. Try again with
         different allocation characteristics.
0516-788 extendlv: Unable to extend logical volume.
```

Note: DB2/6000 database administrators should refer to the additional information contained in the db2sgmgr.log file when the segment manager tool reports AIX errors. Error messages are appended to the db2sgmgr.log file. Thus, the size of

db2sgmgr.log will grow with usage. DB2/6000 database administrators should remove the file periodically to free up space.

Appendix B. SNA Profiles

This appendix shows the SNA profiles needed to set up the communications between DDCS for AIX and DB2 for MVS using an ESCON channel.

Do the initial node setup:

```
smitty sna
 Configure SNA Profiles
   Initial Node Setup
```

Choose channel as the DLC type as shown in Figure 129 and Figure 130 on page 254.

```
                                Initial Node Setup

Type or select a value for the entry field.
Press Enter AFTER making all desired changes.

                                                        [Entry Fields]
* Choose the DLC type you wish this configuration to  channel                +
   represent

F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 129. Node Setup

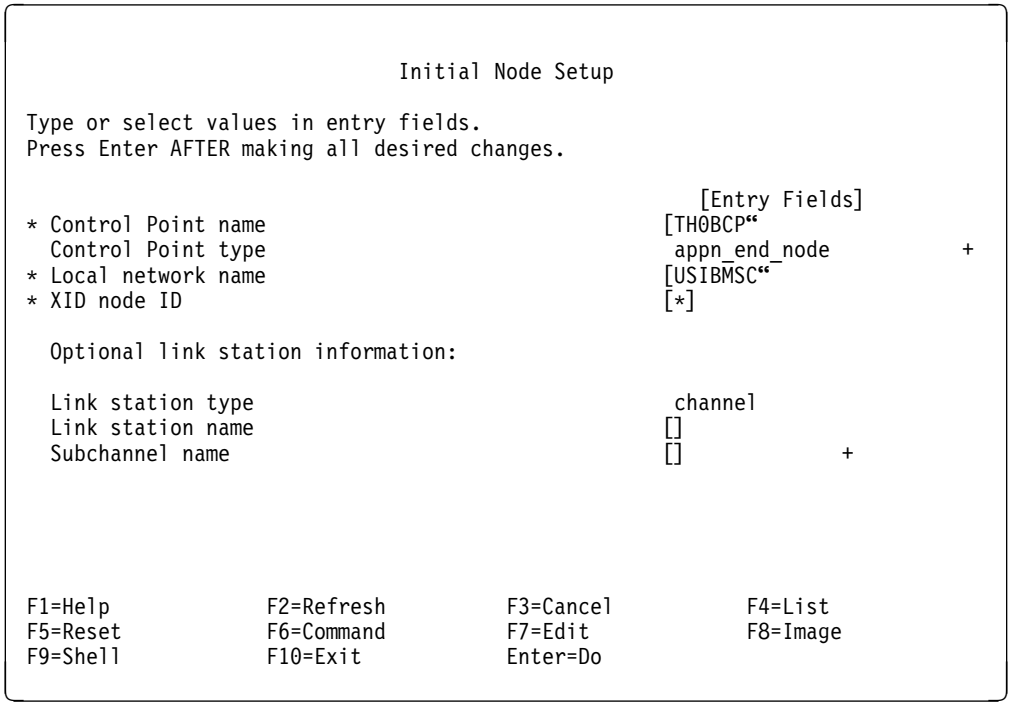


Figure 130. Initial Node Setup

Create the Channel Data Link control profile as shown in Figure 131 on page 255.

```
smitty sna
 Configure SNA Profiles
 Advanced Configuration
 Links
 Channel
 Channel SNA DLC
 Channel Link Station
```

```
                    Change/Show Channel SNA DLC Profile

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
  Profile name                                        rs6kdlc
  Channel device type                                 escon                 +
  Force disconnect time-out (1-600 seconds)          [600]           #
  User-defined maximum I-Field size?                  no                    +
     If yes, Max. I-Field size (265-4096)            [4096]          #

  Link Recovery Parameters
     Retry interval (1-10000 seconds)                [60]            #
     Retry limit (0-500 attempts)                    [20]            #

  Comments                                           []


F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 131. SNA DLC Profile

Create the Channel Link station profile as shown in Figure 132 on page 256 to Figure 135 on page 257.

```
smitty sna
 Configure SNA Profiles
 Advanced Configuration
 Links
 Channel
 Channel Link Station
```

```
                    Change/Show Channel Link Station Profile

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

[TOP]                                          [Entry Fields]
  Current profile name                          TH0BCP1
  New profile name                             []
  Use Control Point's XID node ID?              yes                      +
     If no, XID node ID                        [*]
* SNA DLC Profile name                         [rs6kdlc]      +
  Subchannel name                              [HCON1]        +
  Stop link station on inactivity?              no                       +
     If yes, Inactivity time-out (0-10 minutes) [0]             #
  LU address registration?                      no                       +
     If yes,
        LU Address Registration Profile name    []             +
  Trace link?                                   no                       +
     If yes, Trace size                         long                     +
[MORE...33]

F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 132. Channel Link Station Profile

```
                    Change/Show Channel Link Station Profile

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

[MORE...11]                                    [Entry Fields]
  Trace link?                                   no                       +
     If yes, Trace size                         long                     +

  Adjacent Node Identification Parameters
     Verify adjacent node?                      no                       +
     Network ID of adjacent node               []
     CP name of adjacent node                  []
     XID node ID of adjacent node (LEN node only) [*]
     Node type of adjacent node                 learn                    +

  Link Activation Parameters
     Solicit SSCP sessions?                     yes                      +
     Activate link station at SNA start up?     yes                      +
[MORE...22]

F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 133. Channel Link Station Profile Menu

```
                  Change/Show Channel Link Station Profile

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

[MORE...22]                                    [Entry Fields]
     Solicit SSCP sessions?                          yes                    +
     Activate link station at SNA start up?          yes                    +
     Activate on demand?                             no                     +
     CP-CP sessions supported?                       yes                    +
     If yes,
        Adjacent network node preferred server?      no                     +
     Partner required to support CP-CP sessions?     no                     +
     Initial TG number (0-20)                       [1]             #

  Restart Parameters
     Restart on normal deactivation?                 yes                    +
     Restart on abnormal deactivation?               yes                    +

[MORE...11]

F1=Help            F2=Refresh          F3=Cancel           F4=List
F5=Reset           F6=Command          F7=Edit             F8=Image
F9=Shell           F10=Exit            Enter=Do
```

Figure 134. Channel Link Station Menu

```
                  Change/Show Channel Link Station Profile

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

[MORE...33]                                    [Entry Fields]
     Restart on abnormal deactivation?               yes                    +

  Transmission Group COS Characteristics
     Effective capacity                             [39321600]      #
     Cost per connect time                          [128]           #
     Cost per byte                                  [128]           #
     Security                                        nonsecure              +
     Propagation delay                               minimum                +
     User-defined 1                                 [128]           #
     User-defined 2                                 [128]           #
     User-defined 3                                 [128]           #

  Comments                                          []
[BOTTOM]

F1=Help            F2=Refresh          F3=Cancel           F4=List
F5=Reset           F6=Command          F7=Edit             F8=Image
F9=Shell           F10=Exit            Enter=Do
```

Figure 135. Channel Link Station Menu Profile

Create the local LU profile as shown in Figure 136 on page 258.

```
smitty sna
 Configure SNA Profiles
 Advanced Configuration
 Sessions
 LU 6.2
```

```
LU 6.2 Local LU
Add a Profile
Change/Show a Profile
```

```
                    Change/Show LU 6.2 Local LU Profile

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
  Current profile name                                THOBCP1
  New profile name                                   []
  Local LU name                                      [TH0BCP1]
  Local LU alias                                     [TH0BCP1]
  Local LU is dependent?                              no                      +
     If yes,
        Local LU address (1-255)                     []              #
        System services control point
           (SSCP) ID (*, 0-65535)                    [*]
        Link Station Profile name                    []             +
  Conversation Security Access List Profile name     []
  Recovery resource manager (RRM) enabled?            no                      +

  Comments                                           []

F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 136. Creating Local LU Profile

Create the Side Information profile as shown in Figure 137 on page 259.

```
smitty sna
 Configure SNA Profiles
 Advanced Configuration
 Sessions
 LU 6.2
 LU 6.2 Side Information
```

```
                Change/Show LU 6.2 Side Information Profile

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
  Current profile name                               TH0BCP1
  New profile name                                   []
  Local LU or Control Point alias                    [TH0BCP1]      +
  Provide only one of the following:
     Partner LU alias                                []             +
     Fully qualified partner LU name                 [USIBMSC.SCLUDB41]
  Mode name                                          [IBMRDB]       +
  Remote transaction program name (RTPN)             [07F6C4C2]
  RTPN in hexadecimal?                               yes                    +

  Comments                                           []



F1=Help            F2=Refresh         F3=Cancel          F4=List
F5=Reset           F6=Command         F7=Edit            F8=Image
F9=Shell           F10=Exit           Enter=Do
```

Figure 137. Creating Side Information Profile

Create the Partner LU profile as shown in Figure 138.

```
smitty sna
 Configure SNA Profiles
 Advanced Configuration
 Sessions
 LU 6.2
 LU 6.2 Partner LU
```

```
                  Change/Show LU 6.2 Partner LU Profile

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
  Current profile name                               SCLUDB41
  New profile name                                   []
  Fully qualified partner LU name                    [USIBMSC.SCLUDB41]
  Partner LU alias                                   [SCLUDB41]
  Parallel sessions supported?                       yes                    +
  Session security supported?                        no                     +
  Conversation security level                        conversation           +

  Comments                                           []



F1=Help            F2=Refresh         F3=Cancel          F4=List
F5=Reset           F6=Command         F7=Edit            F8=Image
F9=Shell           F10=Exit           Enter=Do
```

Figure 138. Creating LU Profile

Create the Mode profile as shown in Figure 139 on page 260.

```
smitty sna
 Configure SNA Profiles
 Advanced Configuration
 Sessions
 LU 6.2
 LU 6.2 Mode
```

```
                        Change/Show LU 6.2 Mode Profile

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                        [Entry Fields]
  Current profile name                                   IBMRDB
  New profile name                                      []
  Mode name                                             [IBMRDB]
  Maximum number of sessions (1-5000)                   [8]              #
  Minimum contention winners (0-5000)                   [4]              #
  Minimum contention losers (0-5000)                    [0]              #
  Auto activate limit (0-500)                           [0]              #
  Upper bound for adaptive receive pacing window        [16]             #
  Receive pacing window (0-63)                          [16]             #
  Maximum RU size (128,...,32768: multiples of 32)      [32768]          #
  Minimum RU size (128,...,32768: multiples of 32)      [256]            #
  Class of Service (COS) name                           [#CONNECT]

  Comments                                              []

F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 139. Creating Mode Profile

Create the Partner LU Location profile as shown in Figure 140 on page 261.

```
smitty sna
 Configure SNA Profiles
 Advanced Configuration
 Sessions
 LU 6.2
 Partner LU Location
```

```
                Change/Show Partner LU 6.2 Location Profile

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
  Current profile name                                SCLUDB41
  New profile name                                   []
  Fully qualified partner LU name                    [USIBMSC.SCLUDB41]
  Partner LU location method                          owning_cp              +
  If owning_cp,
     Fully qualified owning Control Point (CP) name  [USIBMSC.SC30M]
     Local node is network server for LEN node?       no                    +
     Fully qualified network node server name        []
  If link_station,
     Local LU name                                   []
     Link Station Profile name                       []            +

  Comments                                           []


F1=Help            F2=Refresh         F3=Cancel          F4=List
F5=Reset           F6=Command         F7=Edit            F8=Image
F9=Shell           F10=Exit           Enter=Do
```

Figure 140. Creating LU Location Profile

List of Abbreviations

ADSM	ADSTAR Distributed Storage Manager
ADSTAR	Advanced Storage And Retrieval (now SSD - Storage Systems Division)
AIX	advanced interactive executive (IBM's flavor of UNIX)
API	application program interface
ARP	address resolution protocol
ASCII	American National Standard Code for Information Interchange
CD-ROM	(optically read) compact disk - read only memory
CLIO/S	IBM Client Input/Output Sockets (licensed program)
CPU	central processing unit
DDL	data definition language
DMA	direct memory access
DSM	data systems manager
DSM	distributed systems management
DSM	distribution services manager
Gb	gigabit (10**9 bits or 1,000,000,000 bits) not recommended, use Gbit
GB	gigabyte (10**9 bytes or 1,000,000,000 bytes) case should be Gb
GET	grand elapsed time
GUI	generalized end-user interface
GUI	graphical user interface
HACMP	high availability cluster multi-processing (AIX)
HPS	high performance switch
I/O	input/output
IBM	International Business Machines Corporation
INEWS	information news facility (IBM)
IP	internet protocol (ISO)
ITSC	International Technical Support Center (IBM)
ITSO	International Technical Support Organization
Mb	megabit (million bits) not recommended, use Mbit
MB	megabyte, 1,000,000 bytes (1,048,576 bytes memory) case should be Mb
MVS	Multiple Virtual Storage node $NFSHOSTS" $PROGNAME $NFSHOSTS
NFS	network file system (USA, Sun Microsystems Inc)
PC	Personal Computer (IBM)
PIOFS	Parallel I/O File System
POK	Poughkeepsie, NY
PSSP	AIX Parallel System Support Programs (IBM program product for SP1 and SP2)
PTF	program temporary fix
RISC	reduced instruction set computer/cycles
SCSI	small computer system interface
SDR	system data repository
SMIT	System Management Interface Tool (see also DSMIT)
SMP	shared multiprocessor
SP	IBM RS/6000 Scalable POWERparallel Systems (RS/6000 SP)
SQL	structured query language
TCP	transmission control protocol (USA, DoD)
TCP/IP	Transmission Control Protocol/Internet Protocol (USA, DoD, ARPANET; TCP=layer 4, IP=layer 3, UNIX-ish/Ethernet-based system-interconnect protocol)
TCPIP	transmission control protocol internet protocol
UNIX	an operating system developed at Bell Laboratories (trademark of UNIX System Laboratories, licensed exclusively by X/Open Company, Ltd.)
UPS	uninterruptible power supply/system

URL Uniform Resource Locator ***URL*** Universal Resource Locator

Index

T

U

V

W

ITSO Technical Bulletin Evaluation
RED000

International Technical Support Organization
Migrating and Managing Data on RS/6000 SP
with DB2 Parallel Edition
April 1996

Publication No. SG24-4658-00

Your feedback is very important to help us maintain the quality of ITSO Bulletins. **Please fill out this questionnaire and return it using one of the following methods:**

- Mail it to the address on the back (postage paid in U.S. only)
- Give it to an IBM marketing representative for mailing
- Fax it to: Your International Access Code + 1 914 432 8246
- Send a note to REDBOOK@VNET.IBM.COM

Please rate on a scale of 1 to 5 the subjects below.
(1 = very good, 2 = good, 3 = average, 4 = poor, 5 = very poor)

Overall Satisfaction ____

Organization of the book	____	Grammar/punctuation/spelling	____
Accuracy of the information	____	Ease of reading and understanding	____
Relevance of the information	____	Ease of finding information	____
Completeness of the information	____	Level of technical detail	____
Value of illustrations	____	Print quality	____

Please answer the following questions:

a) If you are an employee of IBM or its subsidiaries:
   Do you provide billable services for 20% or more of your time? Yes____ No____
   Are you in a Services Organization? Yes____ No____

b) Are you working in the USA? Yes____ No____

c) Was the Bulletin published in time for your needs? Yes____ No____

d) Did this Bulletin meet your needs? Yes____ No____
   If no, please explain:

_____

_____

What other topics would you like to see in this Bulletin?

_____

_____

What other Technical Bulletins would you like to see published?

_____

Comments/Suggestions: **(THANK YOU FOR YOUR FEEDBACK!)**

Name _____ Address _____

Company or Organization _____

Phone No. _____

ITSO Technical Bulletin Evaluation **RED000**
SG24-4658-00

Cut or Fold
Along Line

Fold and Tape **Please do not staple** Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM International Technical Support Organization
Mail Station P099
522 SOUTH ROAD
POUGHKEEPSIE NY
USA 12601-5400

Fold and Tape **Please do not staple** Fold and Tape

Cut or Fold
Along Line

Printed in U.S.A.