PSSP 3.2:
RS/6000 SP
Software Enhancements

Information on the latest hardware and software for the RS/6000 SP system

Install, tailor, and configure the new hardware and software

Planning information to help you design solutions

Dino Quintero, Kiriko Aoyama,
Marcelo Barrios, Dave Border,
Hassan Ahmed Elsetohy,
Cristian Manasoiu, Sooyoung Moon,
Marcelo Silva

ibm.com/redbooks

SG24-5673-00

International Technical Support Organization

PSSP 3.2: RS/6000 SP Software Enhancements

October 2000

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix B, “Special notices” on page 215.

First Edition (October 2000)

This edition applies to PSSP Version 3, Release 2 (5765-D51) for use with the AIX Version 4, Release 3 Operating System.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. JN9B Mail Station P099
2455 South Road
Poughkeepsie, NY 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.

Contents

Figures

Tables

Preface

In July 2000, IBM announced enhancements to the RS/6000 SP that included a new version of the Parallel System Support Programs, Version 3, Release 2 (PSSP 3.2). This new release of Parallel System Support Programs (PSSP) provides support for the new SP High Node, SP Switch2, and a cluster of S-series Enterprise Servers without the need for an SP frame. V3.2 provides additional security and switch communication function, enhanced utilization of shared disks, and improved performance, reliability, availability, and serviceability (RAS).

This IBM Redbook details the various hardware and software enhancements, which include:

- New hardware support for the SP Switch2, the next-generation switch for the RS/6000 SP system.
- Support for a cluster of S-series Enterprise Servers without an SP frame.
- Exploitation of Distributed Computing Environment (DCE) security services within PSSP distributed services.
- A new PSSP option for enhanced security called Restricted Root Access, which limits the uses of rsh and rcp within PSSP software.
- The installation and use of Kerberos V4 as an authentication method within the AIX remote commands and for system administration on the SP system or cluster is optional.
- Applications that use virtual shared disks can take advantage of the concurrent disk access environment supplied by the AIX operating system supported by Concurrent Virtual Shared Disk (CVSD).
- A new LAPI vector data transfer function that can improve performance of applications that use LAPI functions.
- Improved VSD communications performance via Kernel Low-level Application Programming Interface (KLAPI) reducing CPU utilization and, in many cases, improving bandwidth.
- Enhancements in diagnostic capabilities for the switch and switch adapter management particularly for the SP Switch2.
- First Failure Data Capture (FFDC) is a set of utilities for recording records of failures and significant software incidents, which can shorten the time needed for problem determination.
- A new Boot/Install Server Installation.

Other SP-related products have also announced new releases. This redbook discusses the following:

- General Parallel File System 1.3
- LoadLeveler 2.2
- Parallel Environment for AIX 3.1

This redbook is for IBM customers, technical and marketing professionals, Business Partners, and anyone seeking an understanding of the new hardware and software components and improvements included in this RS/6000 SP announcement.

The team that wrote this redbook

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization, Austin Center.

Dino Quintero is a project leader at the International Technical Support Organization (ITSO), Poughkeepsie Center. He has over nine years of experience in the IT field. He holds a BS in Computer Science and Mathematics and an MS degree in Computer Science from Marist College. Before joining the ITSO, he worked as a Performance Analyst for the Enterprise Systems Group and was a Disaster Recovery Architect for IBM Global Services. He has been with IBM since 1996. His areas of expertise include Enterprise Backup and Recovery, Disaster Recovery Planning and Implementation, AIX, and RS/6000 SP, and he is also a Microsoft Certified Systems Engineer. Currently, he focuses on RS/6000 Cluster technology, writes redbooks, and teaches classes worldwide.

Kiriko Aoyama is an I/T Specialist at the Systems Engineering Division in IBM Japan. She has seven years of experience in the AIX field. She is a certified Advanced Technical Expert. Her areas of expertise include a wide range of AIX related products, particularly RS/6000 SP systems and MQ products.

Marcelo Barrios is a project leader at the International Technical Support Organization, Austin Center. He has been with IBM since 1993 working in different areas related to RS/6000. Currently, he focuses on RS/6000 SP technology, writes redbooks, and teaches IBM classes worldwide.

Dave Border is a Software Support Specialist working in the RS/6000 Support Centre in Basingstoke, UK. He has worked for IBM for three years. His areas of expertise include AIX, RS/6000 SP, ADSM, and OnDemand. He is currently the ADSM technical advisor within the Support Centre. He has a BSc. Hons. Degree in Mathematics from Coventry University, UK.

Hassan Ahmed Elsetohy is a software customer engineer in IBM Australia. He has six years of experience in the IT field including UNIX and, in particular, networking. Hassan worked for IBM Egypt for five years before starting work for IBM Australia. He holds a B.Sc. degree in VLSI Electronics Engineering and has partially fulfilled his M.Sc. in VLSI design. His areas of expertise include AIX, RS/6000 SP, HACMP, ADSM, and LINUX.

Cristian Manasoiu is an IBM Certified Advanced Technical Expert - RS/6000 AIX working for IBM Romania. He has eight years of experience with UNIX systems. His areas of expertise include RS/6000 SP, HACMP, performance monitoring, and LINUX implementation. He teaches AIX, HACMP, RS/6000 SP, and LINUX courses in the IBM Education Center and customers areas. He holds an MS degree in Electronic Engineering from the Polytechnic University of Bucharest, Romania.

Sooyoung Moon is an IT Specialist at the Technical Support Center in IBM Korea. He has been involved in parallel computing on RS/6000 SP since 1995 and joined IBM in 1996. He holds the degrees of BS in atmospheric science and MS in Physics from Seoul National University. His areas of expertise include AIX and RS/6000 performance tuning, HACMP clustering, and RS/6000 SP High Performance Computing benchmark.

Marcelo Silva is a Systems Analyst in Brazil. He has four years of experience in AIX, RS/6000, and Lotus Notes Administration and two years of experience with RS/6000 SP and Kerberos. He holds a degree in Systems Analysis from Universidade Paulista, Brazil. His areas of expertise include Lotus Notes Administration, Windows 95/98/NT, OS/2, Notesview, Token-Ring, VTAM, REXX Language, and Replic-action.

Thanks to the following people for their invaluable contributions to this project:

IBM Poughkeepsie
Robert Blackmore, Richard Coppinger, Robert Curran, Abbas Farazdel, Marty Fullam, Rama Govindaraju, Roger Haskin, Brian Herr, Ted Hoover, Chulho Kim, Yoshimichi Kosuge, Joan McComb, Linda Mellor, Mary Nisley, Mike Roberts, Gautam Shah, John Simpson, Richard Treumann, William Tuel

Comments welcome

Your comments are important to us!

We want our Redbooks to be as helpful as possible. Please send us your comments about this or other Redbooks in one of the following ways:

- Fax the evaluation form found in “IBM Redbooks review” on page 241 to the fax number shown on the form.
- Use the online evaluation form found at **ibm.com**/redbooks
- Send your comments in an Internet note to redbook@us.ibm.com

Chapter 1. PSSP 3.2: What's new?

This chapter provides an overview on the enhancements and changes available in PSSP 3.2 components.

1.1 Installation, migration, and coexistence

This section discusses changes in the PSSP 3.2 installation and migration processes, issues to consider during the planning phase for new installations or migrations of existing up and running systems, and some coexistence issues.

1.1.1 Changes

Minor changes have been made to the PSSP 3.2 install code. In releases prior to PSSP 3.2, the default is to create a Boot Install Server (BIS) for each frame in the SP complex, and the first node of each frame happens to be the BIS for all nodes in that frame. These BIS nodes, in turn, use the CWS as their install server. In a single frame SP system, the CWS is the BIS for all nodes.

Note

PSSP 3.2 installation or migration requires AIX 4.3.3.

Each BIS node, along with its clients, make a unique NIM domain; therefore, having multiple BIS means that multiple NIM databases are existing, which implies that the CWS does not have install information about nodes served by other BIS nodes. Moreover, having multiple BISs is more difficult to maintain.

In PSSP 3.2, the BIS default has changed. In PSSP 3.2, if the number of nodes in the SP complex is less than or equal to 40 nodes, then the CWS is the BIS for all nodes, otherwise, (if the number of nodes is greater than 40 nodes) the old default applies, which is having the first node in each frame as the BIS for nodes in that frame.

This iis ionie istep itoward eitncouraging as many customers as possible to use only one BIS, and as such, having the CWS knowing install information of all nodes in the SP complex.

The BIS defaults can be changed by using the `spchvgobj` command.

There are no migration issues related to the change in BIS defaults. The new default settings of BIS only apply when new nodes are added to the system or the system is re-installed. During migration, SDR settings are not changed, and the existing BIS settings will remain in effect.

All PSSP 3.2 components, except the PTPE, have been enhanced to use the security services. SP security service's new capabilities include:

- Kerberos V4 is now optional and is not mandatory to install, configure or use.
- PSSP supplies commands to configure PSSP use of the DCE security services on the CWS. For the nodes, PSSP install code installs and configures DCE for you.
- DCE authentication can be used for AIX remote commands that requires root.
- DCE can also be used for trusted services authentication and for AIX remote command authentication.

Please refer to Section 1.7, "Security management" on page 40, for more details.

1.1.2 Planning for installations and migrations

Due to the changes introduced in PSSP 3.2, and in the new hardware supported, thorough planning is required before doing a new installation or performing a migration. In this section, various planning issues are discussed. For more details about SP planning, please refer to *RS/6000 SP: Planning, Volume 2, Control Workstation and Software Environment*, GA22-7281.

1.1.2.1 New hardware supported

Installing or migrating to PSSP 3.2 will allow the installation of the following new hardware:

- The SP Switch2 and SP Switch2 adapter (only supported by the POWER3 SMP nodes).
- The 375 MHz POWER3 SMP thin (F/C 2056), wide (F/C 2057), and POWER3 SMP High (F/C 2058) nodes. However, the 375 MHz POWER3 SMP thin and wide nodes are also supported by PSSP 3.1.1, but the 375 MHz POWER3 SMP High node requires PSSP 3.1.1 plus APAR IY11093.
- New PCI SSA adapters.

1.1.2.2 SP Switch2

Migrating an existing SP partitioned system to PSSP 3.2 with the SP Switch2 requires that the system be unpartitioned, as the SP Switch2 does not support partitioning.

The SP Switch2 supports only the POWER3 SMP nodes. Therefore, if the customer has, for example, one of the SP-attached servers (S70, S7A, and/or S80) and wants to keep them, upgrading to the SP Switch2 is not a valid upgrade path.

Please refer to Section 1.5, “SP Switch support” on page 37, for more details about the SP Switch and SP Switch2 support in PSSP 3.2.

1.1.2.3 SP Switch Router

PSSP 3.2 continues the support of the SP Switch Router (GRF) as a dependent node. The SP Switch Router is supported by the SP Switch but *not* by the SP Switch2.

The SP Switch router has two optional sizes. The smaller unit has four internal slots, and the larger unit has sixteen. One slot must be occupied by an SP Switch router adapter card, which provides the SP connection. The other slots can be filled with any combination of network connection cards.

Please refer to *IBM 9077 SP Switch Router: Get Connected to the SP Switch*, SG24-5157, or SP Switch router documentation for more details.

1.1.2.4 Security issues

PSSP components have been enhanced to provide an additional, optional level of security that restricts the root use of `rsh` and `rcp` from a node. This restricted mode can be used with Kerberos V4 or with DCE.

If HACMP is implemented in the SP complex, DCE should not be used as the security management protocol in the partition where HACMP is implemented. The same applies for HACWS.

This removes the need of PSSP to internally run remote `rsh` and `rcp` commands as root. When this feature is enabled, PSSP does not grant the root user the authority to run `rsh` or `rcp` from a node, which impacts many other procedures. For example, HACMP and HACWS.

1.1.2.5 Reserving ports

Components of high-availability infrastructure, such as topology services, group services, and event management, need port numbers from the range 10000 to 10100, inclusively. Therefore, these port numbers must be reserved in the /etc/services file on the CWS before the high-availability components are installed.

For procedures on how to reserve port numbers, please refer to *RS/6000 SP: Planning, Volume 2, Control Workstation and Software Environment*, GA22-7281

1.1.3 Installation

This section is not meant to replace the PSSP installation and migration guide, it only throws some light on the new steps required to be done when installing a new SP system. These new steps are mainly related to the DCE configuration; otherwise, the installation procedure is pretty similar to that of PSSP 3.1; so, please use the *IBM Parallel System Support Programs for AIX: Installation and Migration Guide*, GA22-7347, to install or reinstall your SP system.

SP TaskGuides is a new feature introduced in PSSP 3.2. SP TaskGuides is a wizard-like application that makes the SP system installation and some administration tasks easier. SP TaskGuides is discussed in more detail in Section 1.4, “SP TaskGuides” on page 19.

The following section explains the procedure to install DCE security in your environment. For more detailed information on how to use security in your SP system, refer to the redbook, *Exploiting RS/6000 SP Security: Keeping It Safe*, SG24-5521.

If you are not going to use DCE as your SP security manager, you can skip the rest of this section.

If you are going to use DCE, security and Cell Directory Service (CDS) servers must be accessible from the CWS. However, for the nodes, PSSP installation installs and configures DCE for you.

1.1.3.1 Installation procedure with DCE

The installation process starts, as in prior releases of PSSP, by preparing the CWS:

1. Preparing, installing, and connecting the hardware and cables
2. Making file systems and directory structures
3. Copying AIX LPPs, installing necessary AIX PTFs
4. Tuning necessary operating system and network parameters
5. Copy the PSSP LPPs and AIX mksysb images
6. Install the PSSP 3.2 on the CWS

After completing the above tasks, it is now the time to set the authentication method. At this time, you can either choose Kerberos V4, DCE, or Standard AIX authentication. If you have chosen DCE, you start by creating DCE groups, principals, and accounts by running the `config_spsec` command.

> **Note**
>
> To run the `config_spsec`, you must `dce_login` first; otherwise you will receive the error shown in Figure 1.

```
[sp5en0:mbulynko$config_spsec -c -v
This command requires cell administrator authority. Continue? (y/n) y

config_spsec: 0016-505 Failed to create group: spsec-admin.
config_spsec: 0016-511 The dcecp command below returned non-zero return
code.
/opt/dcelocal/bin/dcecp -c group create spsec-admin -inprojlist yes
Command output is:
Error: msgID=0x17122081  User is not authorized to update record.
```

Figure 1. config_spsec must be run by the cell admin

Running the `config_spsec` command successfully produces an output similar to that shown in Figure 2 and continued in Figure 3 on page 7.

```
[sp5en0:# config_spsec -c -v

This command requires cell administrator authority. Continue? (y/n) y

Running "check_prereqs" subroutine ...
Checking state of DCE ...
Running "open_io" subroutine ...
Running "parse_defaults" subroutine ...
Parsing spsec_defaults file ...
Running "parse_overrides" subroutine ...
Running "create_default" subroutine ...

Please enter cell administrator id to be added to ACL admin group: weaam

Adding cell administrator id to group spsec-admin ...
Creating org - spsec-services ...
Creating group - spsec-services ...
Running "create_accts_on_cws" subroutine ...

Your cell administrator password is required to create accounts.
Please enter your cell administrator password:
Processing service principal - LoadL/GSmonitor ...
Processing service principal - LoadL/Kbdd ...
Processing service principal - LoadL/Master ...
Processing service principal - LoadL/Negotiator ...
Processing service principal - LoadL/Schedd ...
Processing service principal - LoadL/Startd ...
Processing service principal - LoadL/Starter ...
Processing service principal - mmfs/mmfsd ...
Processing service principal - ppe/dpcl ...
Processing service principal - ppe/pmdv3 ...
Processing service principal - rsct/hats ...
Processing service principal - rsct/rsct ...
Processing service principal - ssp/css ...
Processing service principal - ssp/hardmon ...
Processing service principal - ssp/pmand ...
Processing service principal - ssp/sdr ...
Processing service principal - ssp/sp_configd ...
Processing service principal - ssp/spbgroot ...
Processing service principal - ssp/spmgr ...
Processing service principal - ssp/switchtbld ...
Processing service principal - ssp/sysctl ...
```

Figure 2. config_spsec command output

```
Running "create_CDSpath" subroutine ...
Running "create_groups" subroutine ...
Creating group - sdr-system-class-write-services ...
Creating group - sdr-write ...
Creating group - sdr-write-services ...
Creating group - switchtbld-clean ...
Creating group - switchtbld-load ...
Creating group - switchtbld-status ...
Creating group - sysctl-cwsroot ...
Creating group - sysctl-default ...
Creating group - sysctl-install ...
Creating group - sysctl-install-services ...
Creating group - sysctl-logmgt ...
Creating group - sysctl-logmgt-services ...
Creating group - sysctl-master ...
Creating group - sysctl-mmcmd ...

Running "add_mbrs" subroutine ...
Adding members to group haem-services
Adding members to group hm-control-services
Adding members to group mmfs-services
Adding members to group sdr-admin-services
Adding members to group sdr-system-class-admin-services
Adding members to group sdr-write-services
Adding members to group sdr-write-services
Adding members to group sdr-write-services
Adding members to group sysctl-install-services
Adding members to group sysctl-logmgt-services
Adding members to group sysctl-mmcmd-services
Adding members to group sysctl-switch-services
Adding members to group sysctl-vsd-services.
```

Figure 3. config_spsec command output (continued)

Note

It is important to log into the SP Administrative principle because it is required for any command that does sdr writes if you have DCE but not K4.

There must be a DCE principal that is a member of hm-admin, sdr-admin, sdr-system-class-admin, sdr-restricted, spsec-admin, and the hm-control groups to continue the install.

Use the appropriate DCE commands to define an administrative principal. The principal can be added to the SP access groups by a cell administrator using `dcecp`:

- `dcecp -c group add sdr-admin -member your_principal`
- `dcecp -c group add hm-admin -member your_principal`

The administrative principals may need access to additional SP groups. Refer to *RS/6000 SP: Planning, Volume 2, Control Workstation and Software Environment*, GA22-7281, for a complete list of groups defined by PSSP to DCE. The access groups (ACC-GRP), whose names do not end in -services are intended for end users. For example, their control facilities could be sysctl, problem management, event management, the switch commands, LoadLeveler, Parallel Environment, and so on.

As root only (remember to log out now from DCE), run the following command to create the CWS-specific keyfiles:

```
#create_keyfiles -c -v
```

Output similar to that in Figure 4 should appear.

```
[sp5en0:/]# create_keyfiles -c -v
Running "check_prereqs" subroutine ...
Checking state of DCE ...
Running "parse_defaults" subroutine ...
Parsing spsec_defaults file ...
Running "parse_overrides" subroutine ...
Running "create_keys" subroutine ...
Running "do_keytab_work" subroutine ...
Creating keytab object, "LoadL/sp5en0/GSmonitor" and randomizing keys.
Running "do_keytab_work" subroutine ...
Creating keytab object, "LoadL/sp5en0/Kbdd" and randomizing keys.
Running "do_keytab_work" subroutine ...
Creating keytab object, "LoadL/sp5en0/Master" and randomizing keys.
Running "do_keytab_work" subroutine ...
Creating keytab object, "LoadL/sp5en0/Negotiator" and randomizing keys.
Running "do_keytab_work" subroutine ...
Creating keytab object, "LoadL/sp5en0/Schedd" and randomizing keys.
Running "do_keytab_work" subroutine ...
Creating keytab object, "LoadL/sp5en0/Startd" and randomizing keys.
Running "do_keytab_work" subroutine ...
Creating keytab object, "LoadL/sp5en0/Starter" and randomizing keys.
Running "do_keytab_work" subroutine ...
Creating keytab object, "mmfs/sp5en0/mmfsd" and randomizing keys.
Running "do_keytab_work" subroutine ...
Creating keytab object, "ppe/sp5en0/dpcl" and randomizing keys.
Running "do_keytab_work" subroutine ...
Creating keytab object, "ppe/sp5en0/pmdv3" and randomizing keys.
Running "do_keytab_work" subroutine ...
Creating keytab object, "rsct/sp5en0/hats" and randomizing keys.
Running "do_keytab_work" subroutine ...
Creating keytab object, "rsct/sp5en0/rsct" and randomizing keys.
Running "do_keytab_work" subroutine ...
Creating keytab object, "ssp/sp5en0/css" and randomizing keys.
Running "do_keytab_work" subroutine ...
Creating keytab object, "ssp/sp5en0/hardmon" and randomizing keys.
.
.
```

Figure 4. Part of create_keyfiles command output

The `chauthts` command must be run; otherwise, the `install_cw` command will fail. Refer to the *PSSP: Installation and Migration Guide*, GA22-7347, for all the possible options.

If you have done dce_login with your SP Administrative principle, then `chauthts dce` followed by `install_cw` will work.

Proceed with the installation steps (`install_cw`, enter site environment data, spframe and reinitialize the SDR, update supervisor microcodes, enter node information, acquire hardware ethernet addresses, configure additional adapters and initial hostnames, and so on).

After completing these basic tasks, plus additional optional ones, it is time to enter the RS/6000 SP Security installation and configuration, which does the following:

- Selects security capabilities that will be installed when the nodes are installed
- Creates DCE hostnames
- Updates SDR
- Configures DCE clients
- Selects the authorization method that the AIX remote commands will use
- Last, but not least, configures SP trusted services to use DCE authorization

You can use smit panels to accomplish these tasks by running the command, `#smitty spauth_config`. The menu shown in Figure 5 should appear. However, PSSP commands will be used until the end of this section.

```
  RS/6000 SP Security

Move cursor to desired item and press Enter.

  Select Security Capabilities Required on Nodes
  Create DCE hostnames
  Update SDR with DCE Master Security and CDS Server Hostnames
  Configure DCE Clients (Admin portion)
  Select Authorization Methods for AIX Remote Commands
  Configure SP Trusted Services to use DCE Authentication
  Create SP Services Keyfiles
  Enable Authentication Methods for AIX Remote Commands
  Enable Authentication Methods for SP Trusted Services
  Hardware Monitor DCE Objects
  Manage SP ACLs
```

Figure 5. SP security smit menus

Now, select the nodes' security capabilities. You must include k4 if at least one of the nodes will be running PSSP code versions prior to 3.2:

```
#spsetauth -p partition_name -i dce k4
```

Create DCE hostnames in the SDR:

```
#create_dce hostname
```

Update the SDR with the DCE Master security and CDS server and configure the DCE clients admin portion:

```
setupdce -u -s sp5en0.pok.ibm.com -d sp5en0.pok.ibm.com
```

```
setupdce -c cell_admin -l /.:/lan_profile
```

Now, configure the SP trusted services to use the DCE authentication:

```
#config_spsec -v
```

The following step is required because it runs updauthfiles, which creates /.k5login, /klogin, and/or /.rhosts:

spsetauth -d `-p partition_name dce`

Start the key management daemon:

```
#/usr/lpp/ssp/bin/spnkeyman_start
```

After DCE setup is completed, as explained above, the rest of the installation procedure can be carried out normally.

1.1.4 Migration

Table 1 illustrates the AIX and PSSP direct migration paths available. If the CWS or any of the nodes are not at any of the mentioned starting levels, it should be migrated to the appropriate level before migrating to PSSP 3.2 and AIX 4.3.3.

Table 1. AIX and PSSP direct migration paths

From	To
PSSP 2.2 and AIX 4.1.5 or 4.2.1	PSSP 3.2 and AIX 4.3.3
PSSP 2.3 and AIX 4.2.1 or 4.3.3	PSSP 3.2 and AIX 4.3.3
PSSP 2.4 and AIX 4.2.1 or 4.3.3	PSSP 3.2 and AIX 4.3.3
PSSP 3.1 or PSSP 3.1.1 and AIX 4.3.3*	PSSP 3.2 and AIX 4.3.3

* If you plan to use pssp_script to upgrade the PSSP, PSSP must be at the highest modification level. For example, PSSP 3.1 must be upgraded to PSSP

3.1.1 before using pssp_script to upgrade to PSSP 3.2. But, if you are going to use the alternative upgrade method explained in the next section, you do not need to do this intermediate step.

1.1.4.1 Alternative method for PSSP migration

Migration steps to PSSP 3.2 are fairly clear and are detailed in the *IBM Parallel System Support Programs for AIX: Installation and Migration Guide*, GA22-7347. We recommend that you follow these steps. However, in the step where PSSP 3.2 installation is carried on the CWS, it is mentioned that the following command can be used:

`installp -aXd /spdata/sys1/install/pssplpp/PSSP-3.2 all`, or

`installp -agd /spdata/sys1/install/pssplpp/PSSP-3.2 -X ssp rsct`

It might be easier to use `smitty update_all` to upgrade the PSSP on the CWS instead of the `installp` command, which will save you from deciding which filesets to install and which filesets to leave. However, using the `smitty update_all` for the migration frequently fails. The `update_all` updates installed software to the latest level (update all). It works if the old and the new file sets have identical names. However, the RSCT files have been repackage several times and `update_all` may migrate only a portion of the RSCT files and not install anything with a completely new name unless the file with the new name is required by prerequisites or corequisites. This leaves you with a mix of old, new, and missing files. You must make sure that the new RSCT file sets have been installed.

A similar approach can be followed to upgrade the PSSP on the nodes. In *IBM Parallel System Support Programs for AIX: Installation and Migration Guide*, GA22-7347, the `pssp_script` is used to upgrade the PSSP code and then to customize the node. This procedure works fine; however, the only way to back out this change is restore the node's mksysb, which is a relatively lengthy process because `pssp_script` commits the filesets.

From our experience, we have seen some situations where backing out the upgrade was necessary; and because they were time critical, the mksysb restore procedure was not the best method. A slight change to the normal upgrade procedure can be done, which makes the upgrade back-out easier and faster. The following steps illustrates the alternative procedure:

Step1: Make full backups

Make sure to make full system, user volume groups, and data backups for the nodes to be migrated. Also, archive the SDR and NIM DB.

Step 2: Apply the AIX 4.3.3 upgrade on the node.

If not yet done, upgrade the AIX version on the node to AIX 4.3.3 by mounting the lppsource directory from the CWS then issue the proper upgrade command (whether it is `update_all` or `installp`) on the node.

Verify the upgrade by:

```
#oslevel -l 4.3.3.0
```

Then, reboot the node.

Step 3: Upgrade the PSSP on the node

Mount the */spdata/sys1/install/pssplpp/PSSP-3.2* file system from the CWS on the node on any mount point, for example */mnt*. Run `smitty update_all`. Modify the attributes as shown in Figure 6.

```
 Update Installed Software to Latest Level (Update All)

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

                                                       [Entry Fields]
* INPUT device / directory for software              /mnt
* SOFTWARE to update                                 _update_all
  PREVIEW only? (update operation will NOT occur)    no +
  COMMIT software updates?                           no +
  SAVE replaced files?                               yes +
  AUTOMATICALLY install requisite software?          no +
  EXTEND file systems if space needed?               yes +
  VERIFY install and check file sizes?               no +
  DETAILED output?                                   no +
Process multiple volumes? yes +
```

Figure 6. smit menus for update_all PSSP upgrade method

Note

You might need to do a preview install before doing the actual upgrade to make sure that the installation will not fail because of missing prerequisites.

Step 4: Change Node configuration data

Now, you need to set the appropriate SDR attributes for the node, especially the *code_version* and *boot_response.* For example, for the node 3 frame, you might use:

```
#spchvgobj -r rootvg -p PSSP-3.2 -v aix433 1 3 1
```

```
#spbootins -s no -r customize 1 3 1
```

Step 5: run setup_server

We have chosen to put the task of running `setup_server` in a separate step because it is a common mistake to forget to run `setup_server` after issuing `spbootins` and/or `spchvgobj`.

Wait for the command to complete processing and make sure that it finishes successfully (rc=0).

Step 6: Reboot the node

Reboot the node. The *boot_response* should be returned to disk again, and the host and switch responds should be alright.

1.1.4.2 Hints and tips

If you decided to change authentication method on your SP system, we highly recommend *not* to do it during the migration process. That is to say, keep whatever security management policy you are using as is, then after the migration is completed and everything is tested and working fine, you can go and change the security management policy, especially if you want to start using DCE.

> **Important**
>
> You must add and establish the new security configuration before the original security configuration is decommissioned.

For example, if you want to start using DCE on an SP system that is already using Kerberos k4 security, first DCE must be initialized on the CWS for the system partition. Initial setup can be shown by running `splstdata -p`.

The output may look as shown in Figure 7 (the portion of interest is highlighted).

```
[sp5en0:/var/adm/SPlogs/css]# splstdata -p
List System Partition Information

System Partitions:
------------------
sp5en0

Syspar: sp5en0
----------------------------------------------------------------
syspar_name     sp5en0
ip_address      192.168.5.150
install_image   default
syspar_dir      ""
code_version    PSSP-3.1.1
haem_cdb_version 949675880,212859648,0
auth_install    k4
auth_root_rcmd  k4
auth_methods    k4:std
ts_auth_methods compat
```

Figure 7. Initial security setup as shown by splstdata -p command

After enabling DCE, the output may look as shown in Figure 8 (only the part of interest is shown).

```
auth_install    dce:k4
auth_root_rcmd  dce:k4
auth_methods    dce:k4:std
ts_auth_methods dce:compat
```

Figure 8. security setup after enabling DCE.

After the nodes have completed the transition to DCE, Kerberos k4 can be decommissioned, and the output of `splstdata -p` will be the same as in Figure 7 but without any reference to `k4`.

Refer to *IBM Parallel System Support Programs for AIX: Administration Guide*, SA22-7348, for detailed procedures on how to change the authentication method.

1.1.5 Coexistence

PSSP 3.2 supports multiple levels of AIX and PSSP in the same system partition. However, only certain combinations of PSSP and AIX are supported to coexist in the same system partition.

Coexistence is supported in the same system partition or a single default system partition (the entire SP system) for nodes running any combination of:

- PSSP 3.2 and AIX 4.3.3
- PSSP 3.1.1 and AIX 4.3.3
- PSSP 2.4 and AIX 4.2.1 or AIX 4.3.3
- PSSP 2.3 and AIX 4.2.1 or AIX 4.3.3
- PSSP 2.2 and AIX 4.1.5 or AIX 4.2.1

Figure 9 illustrates different AIX and PSSP levels that can coexist in the same system partition.

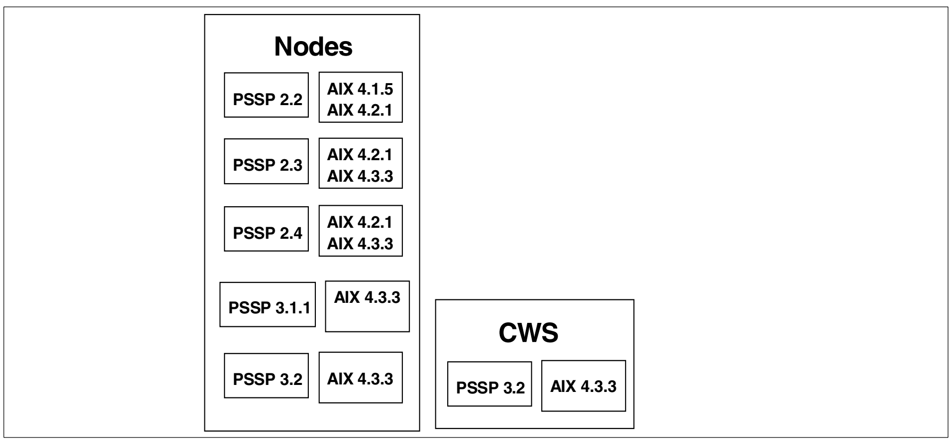

Figure 9. AIX and PSSP level coexistence

Important

If AIX 4.3 is to be used, it must be AIX 4.3.3. PSSP 3.1 support is discontinued; therefore, we recommend to use PSSP 3.1.1.

In general, any combination of the PSSP levels listed in Figure 9 can coexist in a system partition, and you can migrate to a new level of PSSP or AIX one node at a time. However, some PSSP components and related LPPs still have some limitations. Also, many software products have PSSP and AIX dependencies. You must ensure that the proper release levels of these products are used on the nodes.

1.1.5.1 Switch management coexistence issues

Nodes running different supported levels of PSSP can coexist in a mixed system partition. However, if a dependent node is attached to the SP Switch, the primary and backup primary nodes must be running PSSP 3.2 or later, plus the required PTFs. Otherwise, the dependent node will be disabled automatically.

SP Switch and SP Switch2 *cannot* coexist in the same SP system.

1.1.5.2 Extension node coexistence issues

Extension node support in PSSP 3.2 functions in a mixed system partition of nodes running different supported PSSP levels. However, the primary and backup primary nodes must be running PSSP 2.3 or later, plus the required PTFs. Otherwise, the extension node will be disabled automatically. SP systems with SP Switch2 does not support extension nodes.

1.1.5.3 High Availability Cluster Multi-Processing

Although PSSP has no direct dependence on HACMP, if HACMP 4.4 is used, then PSSP 3.2 is required. Restricted root *rsh* or DCE authentication may not be used with HACMP 4.4.

HACMP and HACMP/ES can coexist in the same partition. However, HACMP and HACMP/ES nodes cannot coexist in the same cluster. Only one version of the software can be installed on a node.

1.1.5.4 Virtual Shared Disk coexistence issues

If PSSP 3.2 and any other supported lower-level version of PSSP are running in the SP system partition, the level of the VSD functionality available is that of the lower version of PSSP.

The VSD component does not interfere with migrating one node at a time.

1.1.5.5 General Parallel File System coexistence issues

GPFS 1.3 does not coexist or interoperate with earlier versions of GPFS. However, any file system created by GPFS 1.1 or 1.2 can be used after migrating to GPFS 1.3.

1.1.5.6 AMD and Automount coexistence issues

As of PSSP 2.3, use of the AMD daemon and BSD automounter was replaced with the native AIX Automount that is available as part of NFS filesets of the

AIX BOS. In AIX 4.3.1, the AIX automount daemon was replaced with the AutoFS implementation.

The SP system supports and maintains both AMD and AIX automounter directories and map files. The SP will configure and run the AMD daemon for nodes running PSSP 2.2 and the AIX automounter on nodes running PSSP 2.3 or later.

If the SP user management is also configured, the CWS also creates and maintains both the AIX automounter map file, /etc/auto/maps/auto.u, and the AMD map file, /etc/amd/amd-maps/amd.u. This also applies if file-collections is also used. Both /etc/auto/maps (automounter map file) and /etc/amd/amd-maps (AMD map file) files will be distributed to all nodes, and the node will ignore the inappropriate file (the file that is not compatible with its release level).

1.2 National language support

The PSSP software is NLS-enabled. This means that the software has been made translation-ready and can be translated to any language supported by AIX.

All of the PSSP software components and the following PSSP-related software products have been enabled for internationalization with National Language Support (NLS):

- High Availability Cluster Multi-Processing
- LoadLeveler

The SP software is configured, by default, to operate in the US English language locale. Before you decide to translate the software and run AIX and PSSP in another language, you need to consider the effects of using different language locales. If you decide to use another language locale, after you get the software translated, do the following:

- Set the AIX language locale
- Set the SP administrative locale

For more information on the effects using different language locales, setting the AIX language locale, and setting the SP administrative locale, please refer to *RS/6000 SP: Planning, Volume 2, Control Workstation and Software Environment*, GA22-7281.

1.3 Network Time Protocol (NTP) replacement

There are several ways you might currently be handling synchronization time-of-day clocks on your control workstation and processor nodes. You might already be using Network Time Protocol (NTP), either locally or through the Internet, or you might be using some other time service software.

In PSSP 3.2, the public domain NTP 3.3 has been replaced with AIX NTP 3.4. The rationale behind the replacement of NTP 3.3 is that it is not NLS-enabled. AIX NTP 3.4 is NLS-enabled.

In order to support AIX NTP 3.4 in PSSP 3.2, some changes have been made:

- AIX NTP 3.4 is supported under System Resource Controller (SRC). Internal changes made to start/stop *xntpd* using *src* commands.
- No longer ship public domain NTP 3.3 with PSSP filesets.
- PSSP NTP support configured using spsiteenv or smit site_env_dialog (no changes required)

For more information about how to configure NTP, refer to *IBM Parallel System Support Programs for AIX: Administration Guide*, SA22-7348

1.4 SP TaskGuides

SP TaskGuides run on the control workstation and interact with nodes as a form of advanced online assistance designed to walk you through complex or infrequently performed tasks. Each TaskGuide does not simply list the required steps. It actually performs the steps for you, automating the steps to the highest degree possible and prompting you for input only when absolutely necessary.

1.4.1 Why use SP TaskGuides?

When using SP TaskGuides, you can not forget a step, do steps in the wrong order, execute inapplicable steps, or have to decide if a step is needed. The interface is user friendly and significantly reduces the probability of input errors.

Once launched, a TaskGuide is presented as a sequence of GUI panels that the user moves between by clicking a standard set of navigation buttons along the bottom of each panel (Back, Next and Cancel).

The TaskGuides panels themselves are described in scripts (using HTML-like tags) that a TaskGuide Viewer reads and interprets and then renders on-screen.

The following SP TaskGuides are available:

- Set Site Environment Information
- Add Frames
- Configure New Nodes
- Create Node Image

1.4.2 Architecture

The architecture is depicted in Figure 10.

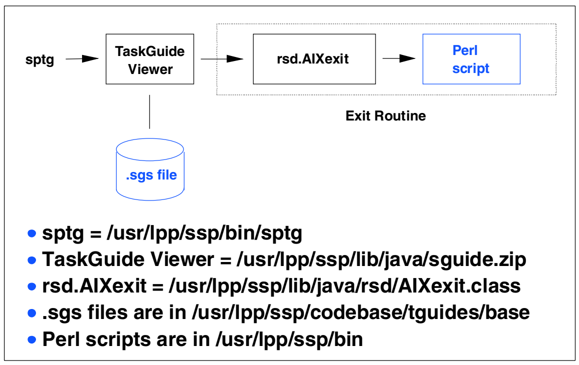

Figure 10. SP TaskGuides architecture in PSSP 3.2

An SP TaskGuide consists of the following:

- A TaskGuide script, which describes the TaskGuide's panels
- The Perl script, which performs the task steps for the user

  `sptg` is a simple SP command for starting a TaskGuide.

Figure 10 shows the SP TaskGuides architecture in PSSP 3.2.

The TaskGuide Viewer is a Java-based GUI. The Viewer's job is to read and interpret a TaskGuide script file, render the TaskGuide on-screen, and respond to user selections, clicks, and input.

The '.sgs' file is a file that describes the various panels of a TaskGuide, sometimes specifies the relationships between the panels, the exit routines to run, and when and how to run them.

The exit routine is the custom program that performs calculations, accesses files, formats data, and/or runs the system commands that need to be run as the user proceeds through the TaskGuide. Exit routines in SP TaskGuides are written in Perl.

1.4.3 Starting SP TaskGuides

Before starting the SP TaskGuide, it is necessary to install the optional *ssp.tguides* fileset.

From the Perspectives panel, double-click on the **SP TaskGuides** icon to access the IBM RS/6000 SP TaskGuides as shown in Figure 11.

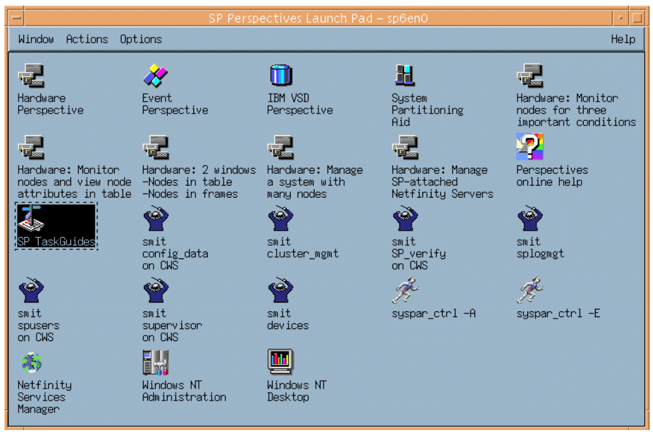

Figure 11. Perspectives panel for access to the IBM RS/6000 SP TaskGuides

Alternatively, from the command line, use the `sptg` command to access the main panel of SP TaskGuides. When this command is executed, the message “Starting SP TaskGuide selection GUI in the background now. The message, “Please be patient while it starts up”, will be displayed.

The screen shown in Figure 12 is displayed.

.

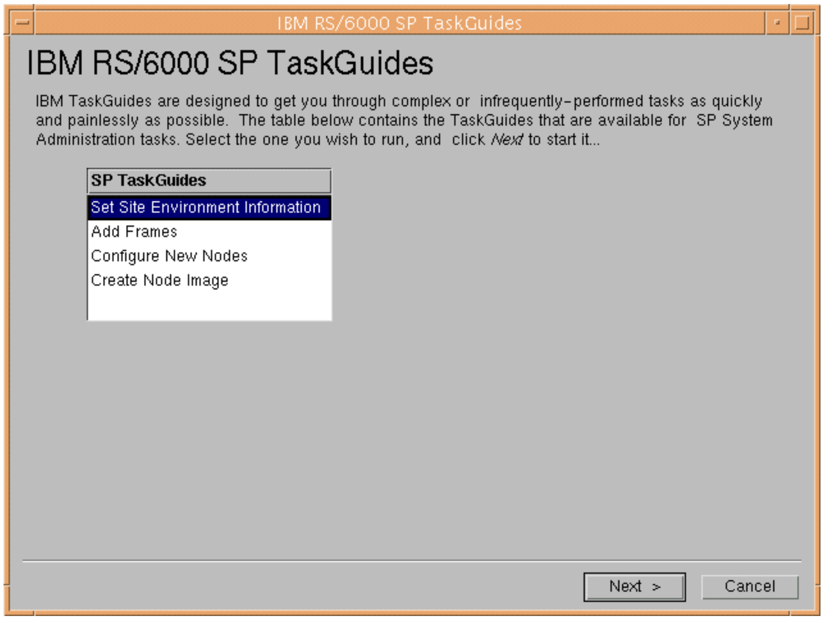

Figure 12. Select the task

To directly start a specific SP TaskGuide, from the command line, use the parameters to access the tasks as follows:

- `sptg setsitenv` to access the Set Site Environment Information
- `sptg addframe` to access the Add Frames task.
- `sptg confnode` to access the Configure New Nodes task.
- `sptg createim` to access the Create Node Image task.

1.4.4 How to use Set Site Environment Information

Each panel of a TaskGuide generally has the same layout. A title may be present along the top. On the left side, there may be an image. Along the bottom are the navigation buttons.

After starting the `sptg` command from Perspectives or the command line, select the **Set Site Environment Information** TaskGuide as shown in Figure 12.

The TaskGuide screens shown in Figure 13 and Figure 14 on page 24 will help you establish the Site Environment information for your SP system. Click on **Next** to continue.

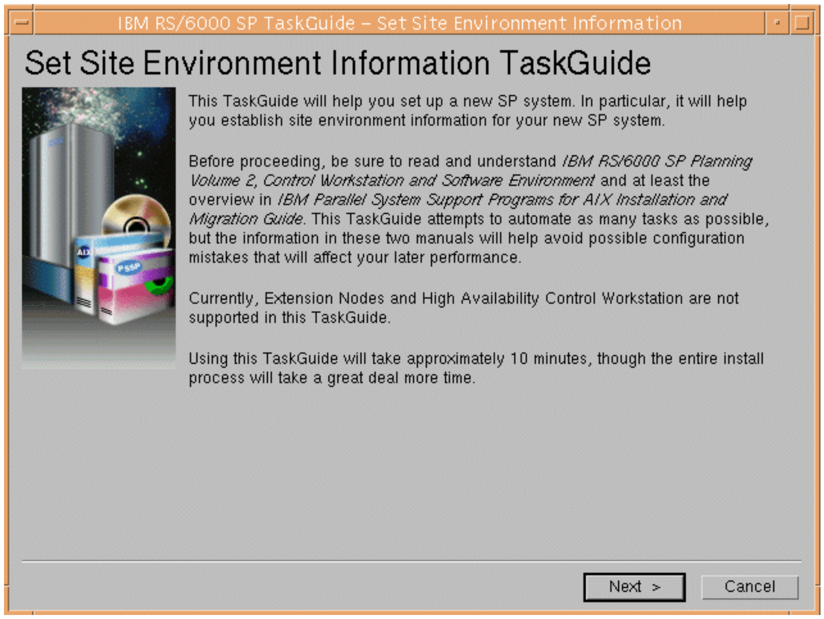

Figure 13. Set Site Environment Information

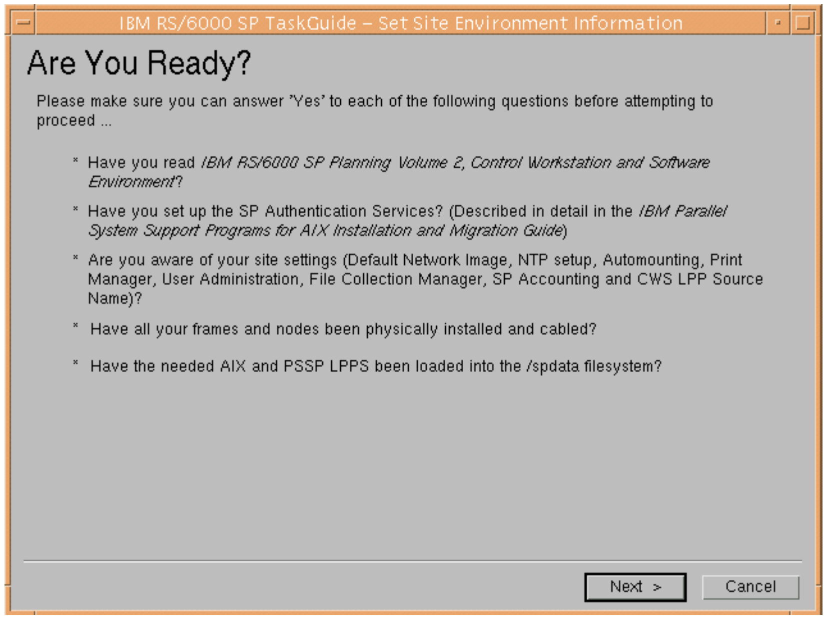

Figure 14. Are You Ready?

The window shown in Figure 15 will help you establish site environment information for your SP system.

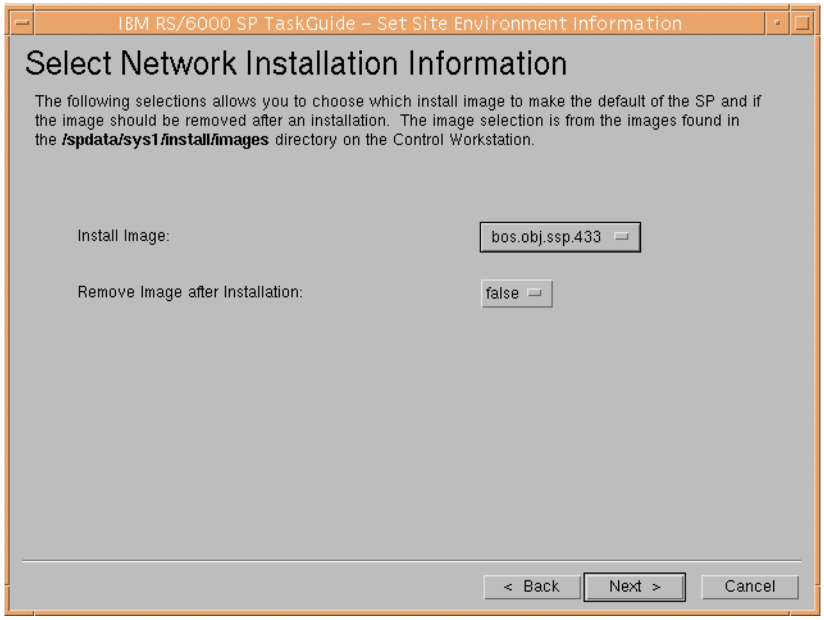

Figure 15. Selecting the Network Installation Information

Select which image to install, click on the button to change the *Install Image* or *Remove image after installation* and then click on **Next** to continue.

The next panel display, shown in Figure 16, will help you answer some questions in order for you to proceed with the environment configuration. Select your Network Time Protocol (NTP), enter your server hostname and the version, and click on **Next** to continue.

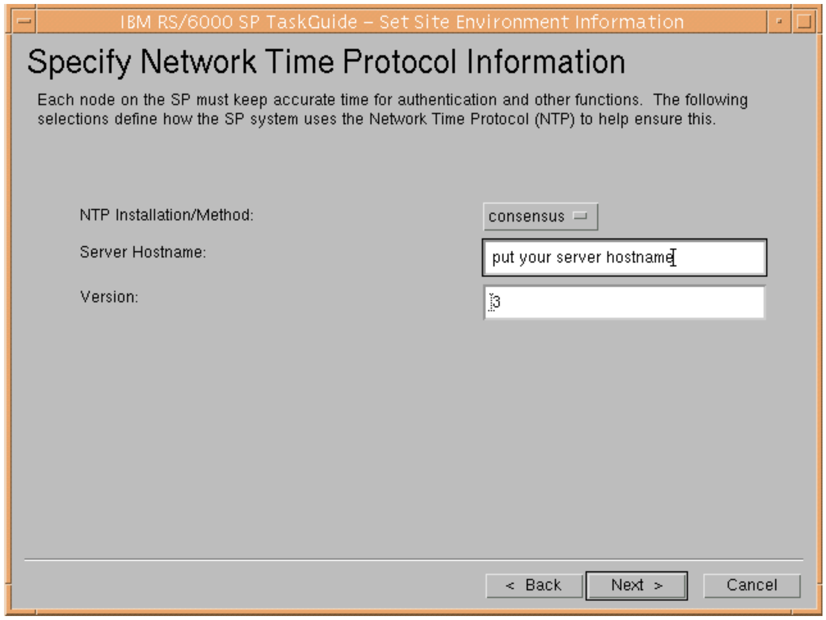

Figure 16. Specify Network Time Protocol Information

As shown in Figure 17, select the Automount mode and click on the **Next** button.

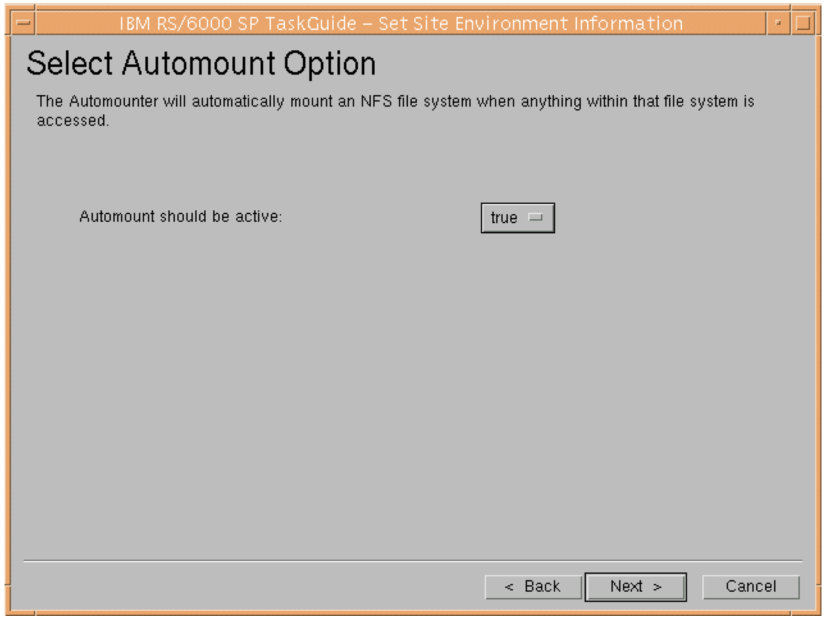

Figure 17. Selecting the Automount Option

Figure 18 shows the Specify Print Manager Mode screen. Define the Print Manager and click on the **Next** button.

.

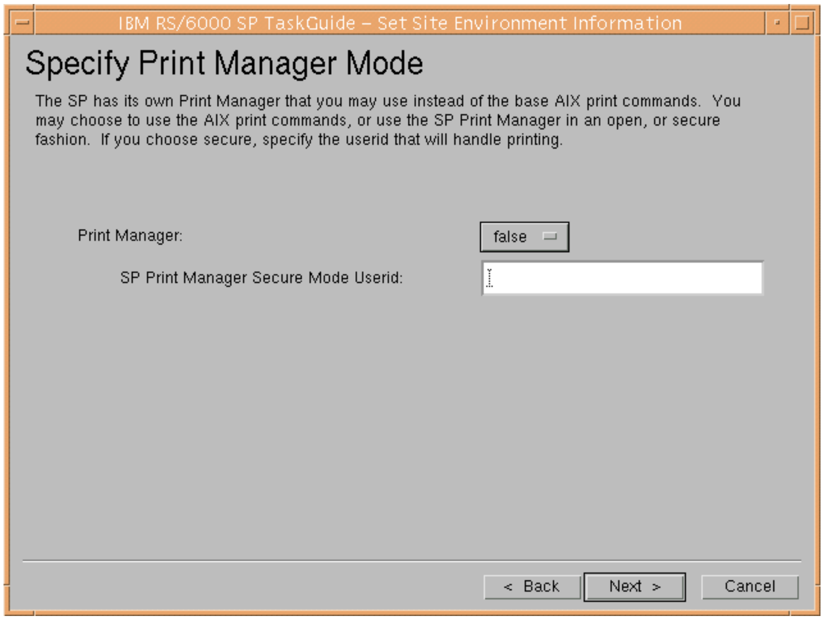

Figure 18. Specify Print Manager Mode

Figure 19 shows the Specify User Administration Options screen. Choose the administration interface and input the user administration information. Then, click the **Next** button.

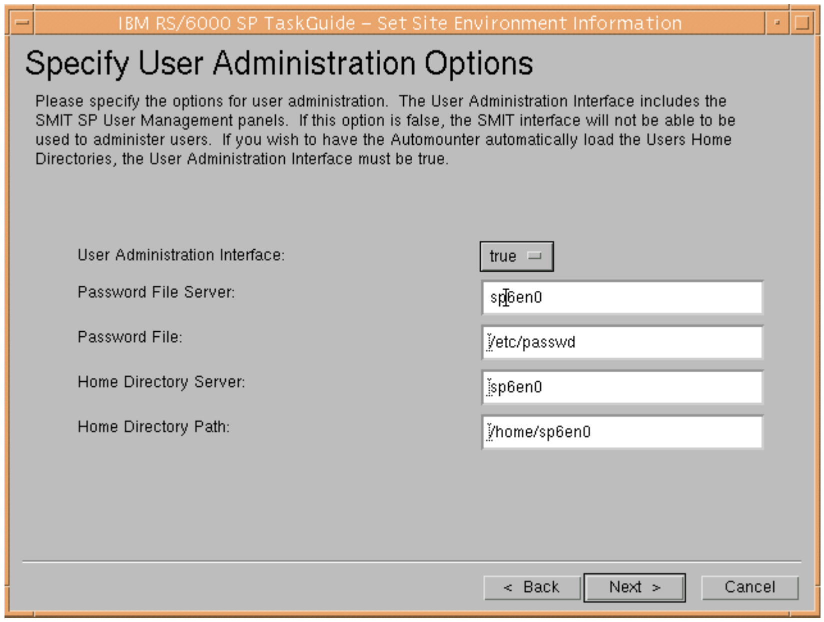

Figure 19. Specify User Administration Options

Specify the File Collection options, as shown in Figure 20, and click on **Next** to continue.

.

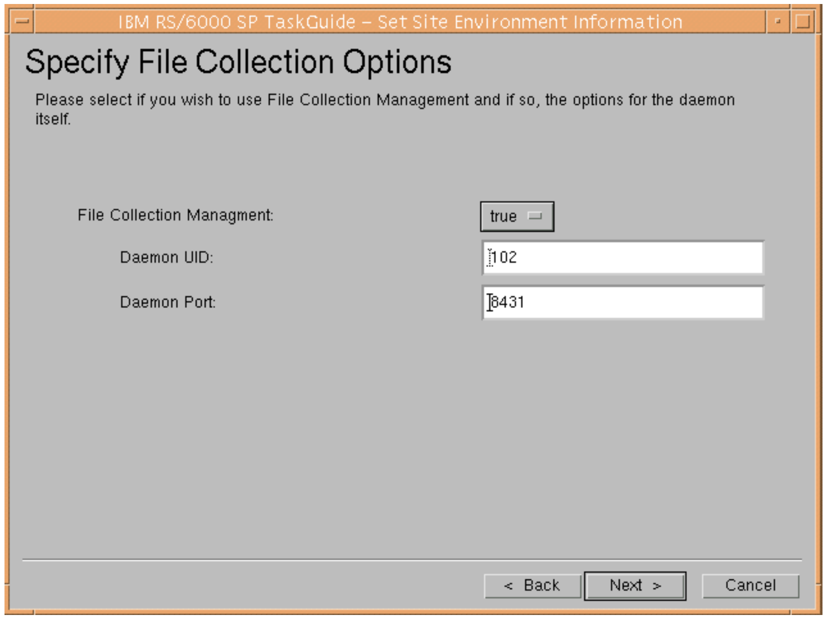

Figure 20. Specify File Collection Options

In the Specify your Accounting Options screen shown in Figure 21, select **true** or **false** and click on **Next**.

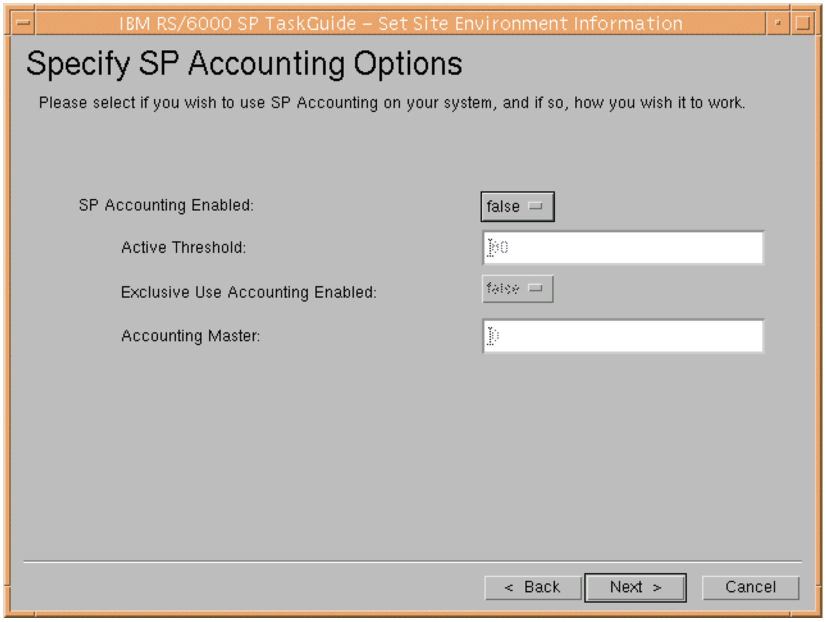

Figure 21. Specify SP Accounting Options

Select the Control Workstation LPP Source Name as shown in Figure 22, and click on **Next**.

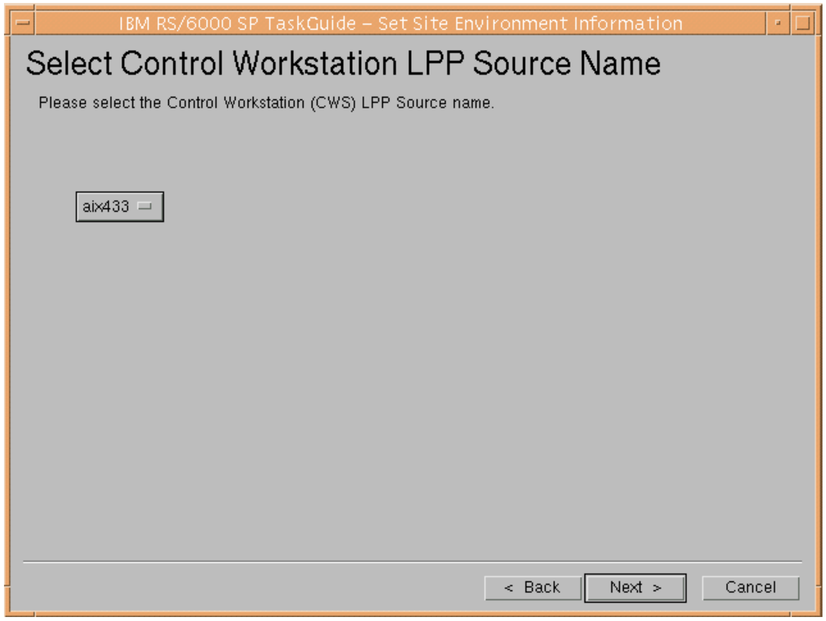

Figure 22. Select the Control Workstation LPP Source Name

The next steps will set the values for the control workstation tunables as shown in Figure 23 and Figure 24 on page 34.

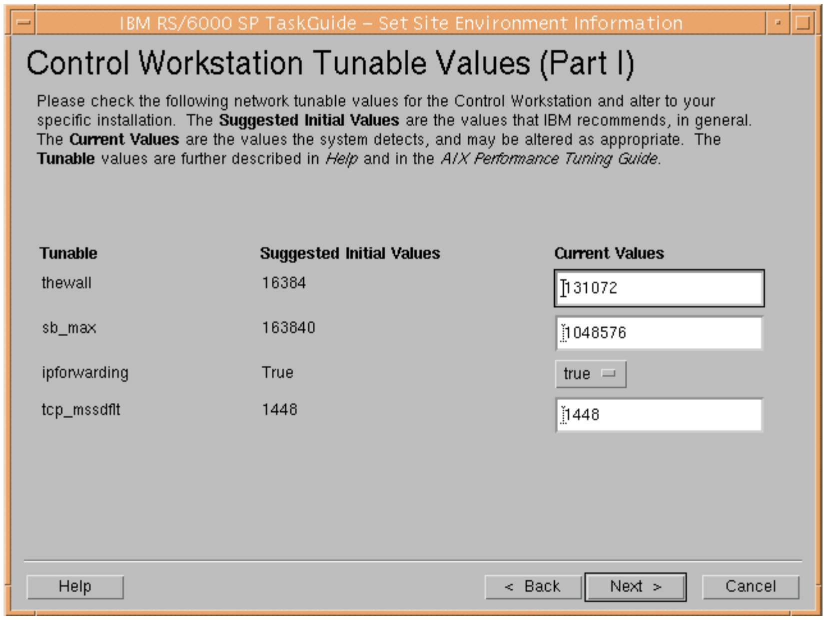

Figure 23. Control Workstation Tunable Values (Part I)

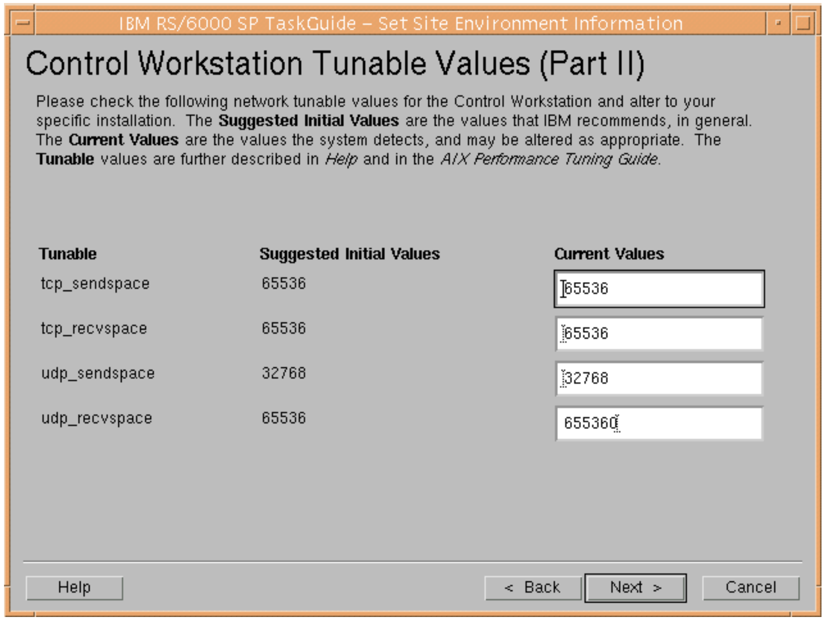

Figure 24. Controlling Workstation Tunable Values (Part II)

In the panel shown in Figure 25, when clicking on **Next**, all changes will be applied. Use the back button if necessary to review the values before you apply the changes.

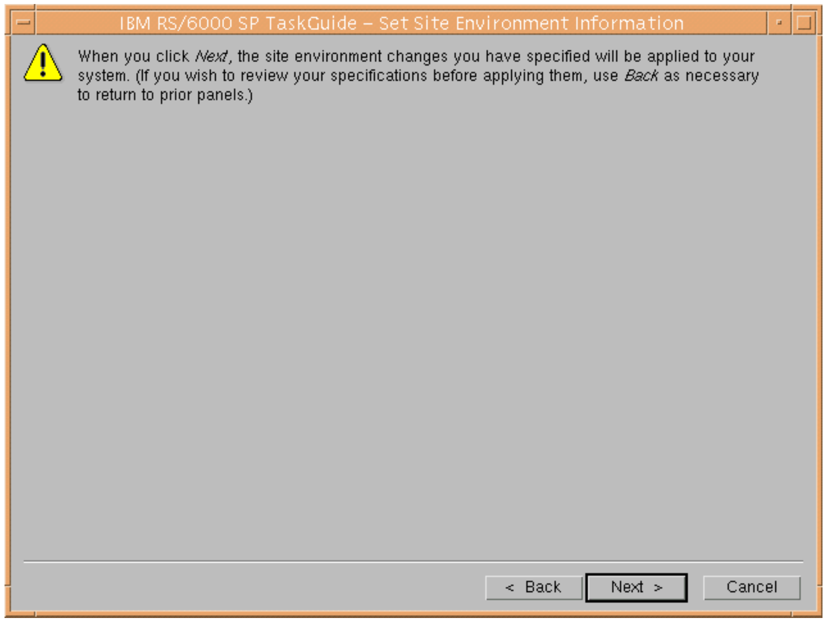

Figure 25. Review your specifications before applying them

1.4.5 Resume or start fresh?

If the SP TaskGuide was interrupted using the Cancel button, the next time that the SP TaskGuides is used, a message will appear in the screen 'Resume or Start Fresh?' with the last incomplete work as shown in Figure 26 on page 36.

Two options are available:

- Start this TaskGuide from scratch. This option will start ignoring the last incomplete work.
- Resume the selected incomplete run from where it was interrupted. Select an incomplete task and click on **Next**.

To view the log of a particular executed TaskGuide, click on the View Log button.

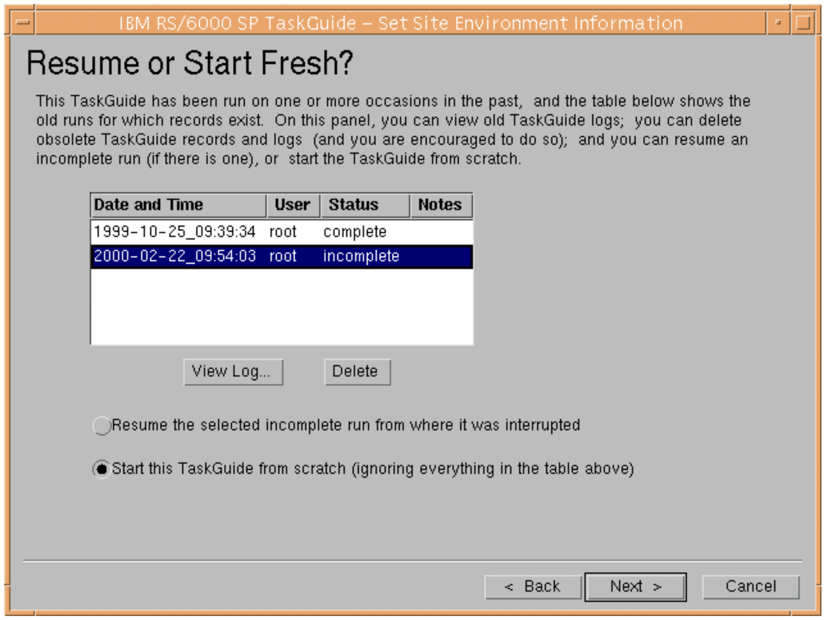

Figure 26. Resume or Start Fresh?

Figure 27 shows the log of the executed Set Site Environment Information TaskGuide.

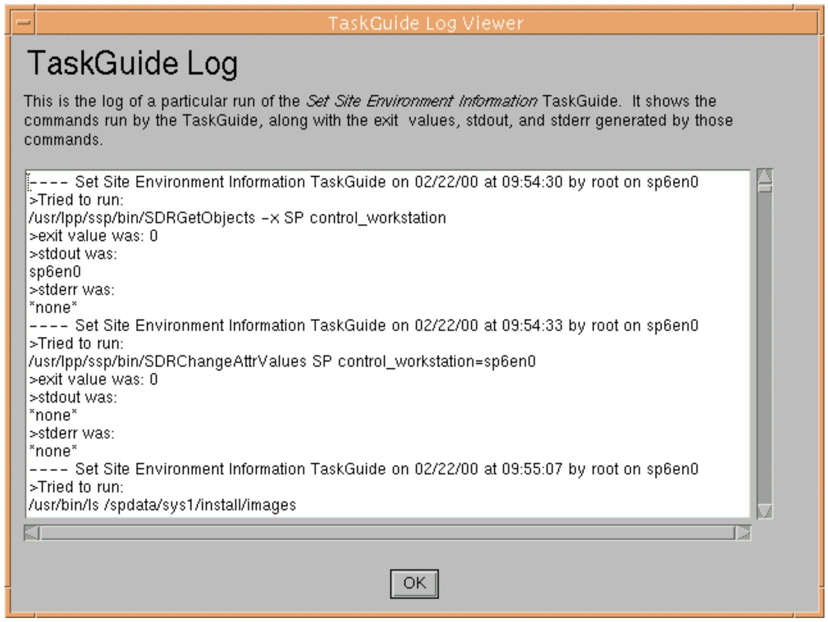

Figure 27. View the TaskGuide Log

1.5 SP Switch support

For information about SP switch support, see Chapter 2, “SP Switch support” on page 75.

1.6 Clustered enterprise servers

PSSP 3.2 introduces support for clustered enterprise servers. The term *clustered enterprise servers* is used in a generic sense to mean a cluster of RS/6000 Enterprise Server S70, S7A, or S80 computers, each running the PSSP software, connected to one control workstation running PSSP 3.2, and connected to the SP Ethernet but without any SP frame in the system. You are no longer required to have at least one SP frame and a node in order to use the PSSP software.

The clustered enterprise servers have different hardware features, which result in some limitations on hardware control and function compared to a standard SP system. The following considerations are significant in planning for PSSP software on a clustered enterprise server:

- It functions like a standard SP processor node, running the PSSP software and many PSSP-related LPPs honoring all the coexistence rules, but there is no physical frame anywhere in the system.
- 64-bit processing is not exploited by PSSP, but you can run 64-bit applications on this server that does not require any PSSP services.
- It connects directly to the SP administrative network (the SP Ethernet) and it connects to the control workstation directly with two RS/232 cables.
- No type of SP switch is supported in a clustered enterprise server configuration. This means that functions that depend on a switch are also not available in this system environment, such as GPFS and user space jobs. However, if with future growth, you might eventually add SP frames, your system will then be subject to all the rules of a standard SP system, and these servers will be SP-attached servers. In this case, plan your system with appropriate frame numbers and switch port numbers in advance (as for SP-attached server) so that you can migrate to an upscale SP system without having to totally reconfigure existing servers.
- Virtual shared disks are not supported.
- If the server to be managed by PSSP is already in use and connected to an IPv6 network, you must remove it from the IPv6 network before configuring PSSP. Some PSSP components tolerate IPv6 aliases for the IPv4 network addresses but not with DCE, HACMP, HACWS, or an SP Switch. For information about the SP tolerating IPv6 aliases for IPv4 network addresses, see the appendix on the subject in the *IBM Parallel System Support Programs for AIX: Administration Guide*, SA22-7348.
- Since it has no SP frame supervisor or SP node supervisor, there is limited control and monitoring from the control workstation. It is, otherwise, treated functionally by PSSP as if it is an SP frame. You must assign a frame number, but assigning a switch port number is optional. If it is possible that you might scale up to using an SP frame and nodes in the future, it is best to plan and assign switch port numbers in advance, following all the rules for an SP-attached server in a standard SP system, to facilitate any upgrade. Otherwise, you will have to completely reconfigure the entire system.
- System partitioning is not supported with clustered enterprise servers.
- You might be able to use HACWS if you understand and accept the limitations. For more information on the limitations and restrictions of HACWS in a clustered enterprise servers system, please refer to RS/6000

SP: Planning, Volume 2, Control Workstation and Software Environment, GA22-7281.

Figure 28 illustrates a system of clustered enterprise servers.

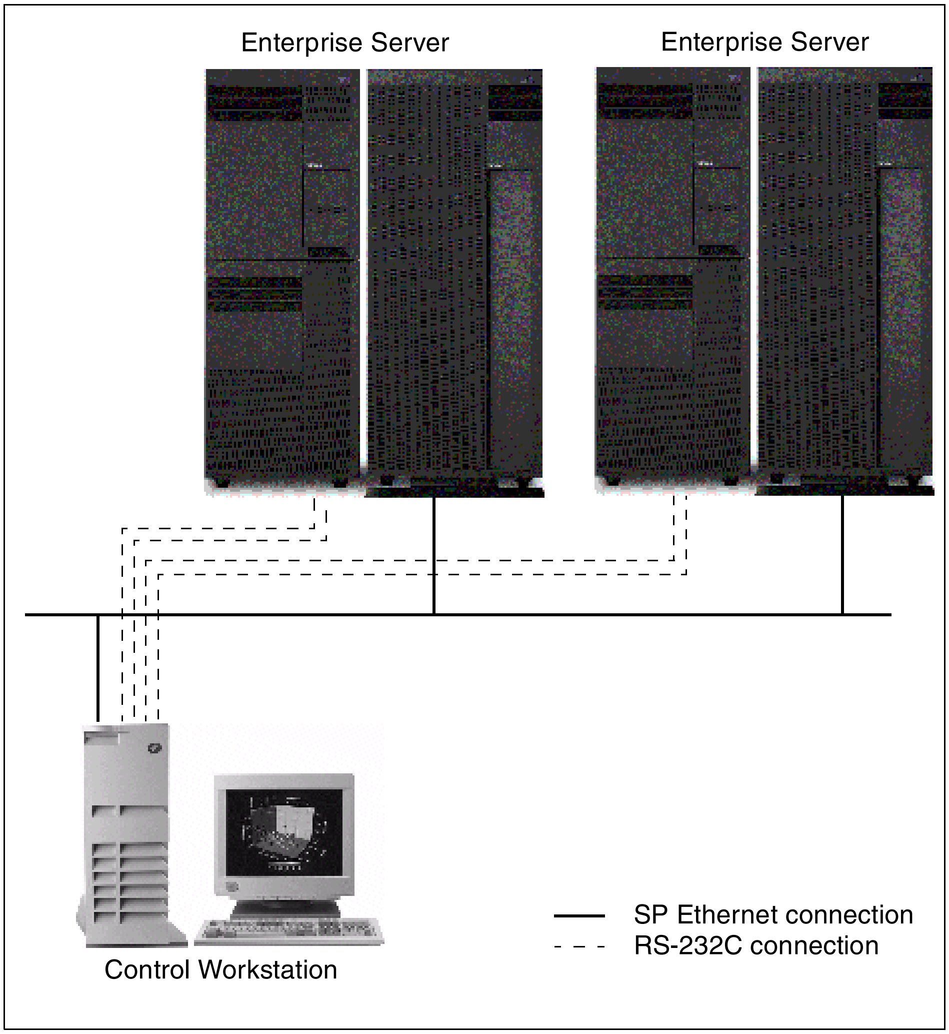

Figure 28. IBM RS/6000 clustered enterprise servers

For more information about clustered enterprise servers, refer to the *IBM RS/6000 Clustered Enterprise Servers Handbook*, SG24-5978.

1.7 Security management

This release of PSSP can be customized to call various authentication methods: Kerberos 4 (K4), Kerberos 5 (K5), and Standard AIX.

To use Kerberos 5, DCE has to be installed. The installation of Kerberos 4 (K4) is optional in PSSP 3.2.

As shown in Figure 29, the structure of the remote shell command (`rsh`) using PSSP 3.2 and DCE is supplied by Kerberos 5 for authentication and DCE for authorization. For the SP Trusted Services, authentication and authorization is supplied by DCE.

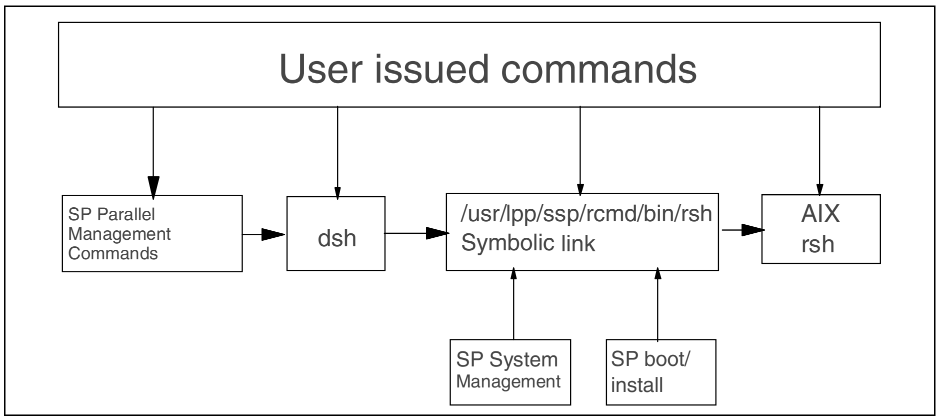

Figure 29. Remote shell structure in PSSP 3.2

PSSP commands will continue calling the PSSP `rsh` command, which is now linked to the AIX `rsh` command. The `rsh` and `rcp` commands in AIX can be configured to use multiple authentication and authorization methods. Figure 29 shows the remote shell structure in PSSP 3.2.

There are three authentication services supported on RS/6000 SP:

- Kerberos V4
- Distributed Computing Environment (DCE) with Kerberos V5
- Standard AIX

Kerberos V4 is now optional to be installed and configured on the control workstation and nodes, and you may also install additional authentication services, such as DCE.

If you decide to use DCE as an authentication mechanism, you are required to have a DCE server accessible from the CWS, and the CWS must be a client in the DCE cell. The CWS can be the DCE server, but it is not required to be. The installation on the nodes is automatic. Time synchronization provided by PSSP is optional, but make sure that time is kept synchronized between the nodes and the control workstation. DTS or NTP, for example, can be used to provide time synchronization.

To use the AIX authenticated remote commands within the SP and allow the system use of `rsh` and `rcp` commands, you can set the following:

- For authorization to use root user access using the authenticated remote commands within each system partition, the options are DCE, Kerberos V4, and Standard AIX.
- For authentication methods enabled in each system partition, the options are Kerberos V5, Kerberos V4, and Standard AIX.

When determining which authentication method to use for remote commands, AIX examines the precedence set by the AIX `chauthent` command. This order determines which authentication method is used when the remote commands are issued. This means that if the first method fails, the second method is tried, and so on.

The order of precedence, defined as being from the highest to the lowest level of security, is DCE, K4, and Standard AIX.

The authentication method is set per partition. Nodes get the information about the authentication settings from the SDR. At boot time, the `rc.sp` script will call the `spauthconfig` script to set up the right authentication method for that node.

1.7.1 Restricting rsh with root access

PSSP 3.2 provides the option to disable the ability of the root user on an SP node to issue an `rsh` or `rcp` command back to the SP control workstation or to another SP node. If this option is used, then:

- The root user on the CWS can continue to issue `rsh` and `rcp` commands to any node within the SP system.
- The root user on an SP node no longer has the ability to issue `rsh` and `rcp` commands to the CWS or to any other node within the SP system.
- The root user on a boot install server cannot issue `rsh` and `rcp` commands to the CWS and to all SP nodes that it serves. The ability to do both must be manually configured by the system administrator.

Removing access for a root user on a node to issue `rsh` and `rcp` commands to the CWS is, at the same time, removing users access to the `telnet`, `ftp`, and `rlogin` commands for Kerberos V5. These are all controlled through DCE and the /.k5login files.

By turning on restricted mode, the capabilities of root users from running `rsh` and `rcp` commands from nodes is not disabled. Restricted mode removes all PSSP software dependencies requiring that node root users must be able to issue remote commands to the CWS and other nodes. You can still manually set up your system to allow root on a node to issue these commands. PSSP simply does not do this automatically anymore.

1.7.1.1 Secure mode indicator

The automatic generation of the `rsh` and `rcp` commands' authorization files was changed in PSSP 3.2, such that entries allowing all nodes access across the system are no longer created.

This indicator in the system specifies that the administrator wishes to operate in this new, highly-secure environment. By default, it is turned off and must be explicitly turned on to enable this new function once all nodes have been upgraded to PSSP 3.2 or later.

This indicator can be set on the CWS using the `spsitenv` command. When the indicator is false, the PSSP code will run in an open, unrestricted environment, executing all `rsh` and `rcp` commands as in previous releases. When the indicator is true, PSSP assumes that it is to run in the restricted environment executing the new command replacements instead of the `rsh` and `rcp` commands.

1.7.1.2 Authorization files

The `rsh` and `rcp` commands' authorization files are constructed and distributed to the CWS and all SP nodes such that the root user on the CWS and any SP node is authorized to issue an `rsh` and `rcp` command as the root user to the CWS and all other SP nodes. If the secure mode indicator is on, this changes, such that only the following entries in the `rsh` and `rcp` commands authorization files are automatically created by PSSP:

- On all SP nodes: Only one entry in the `rsh` and `rcp` authorization files for the root user on the control workstation. For the /.k5login file, this includes the selfhost principal and spbgroot principal for the CWS.
- On the SP CWS: For the /.klogin files, an entry for root.admin and for the local CWS rcmd host (for Kerberos V4 server support). For Kerberos V5 and Standard AIX, there are automatically created entries in the `rsh` and

`rcp` authorization files. On the SP CWS, the /.k5login remains in restricted mode:

- Self host entry
  - `hosts/c187cw.ppd.pok.ibm.com/sef@c187dcecell`
- spbgroot entry for the CWS
  - `ssp/c187cw.ppd.pok.ibm.com/spbgroot@c187dcecell`

The .rhost file contains entries in restricted mode also. However, the file contains control workstation entries only.

For K4 in PSSP 3.2, the nodes refresh their copy of the `rsh` and `rcp` authorization files (for example, /.klogin) by using `rcp` to *pull* the files from the CWS in a non-restricted mode. In a restricted mode, PSSP now generates the K4 authorization files on the nodes. For K5 and standard AIX (for example, /.k5login and /.rhosts), the files are generated on the node containing entries for the CWS and all nodes.

1.7.1.3 Replacing uses of rsh and rcp

All uses of the `rsh` and `rcp` commands that are issued by SP system components on SP nodes and that target the SP CWS or any other SP node were changed to use sysctl or some other means to perform the required function instead. All uses of `rsh` and `rcp` that are issued on the CWS remain unmodified since the CWS continues to have rcmd authorization to the nodes.

In order to change invocations of `rsh` to sysctl, the commands issued remotely through the `rsh` command are packaged as sysctl commands, which reside only on the CWS. The new sysctl commands are protected in a way that is appropriate for that particular command. Typically, this means that any root user on an SP node, within the SP system, can issue the command. This is done using the root.SPbgAdm Kerberos4 principal and the DCE ssp/spbgroot group.

1.7.1.4 Boot/Install services

NIM requires that each NIM master have standard AIX `rsh` access, .rhosts file, to all nodes in its domain; and starting with AIX 4.3.3, NIM also supports Kerberos V4 rcmd authorization once the initial install has been completed on the nodes. The boot/install servers require remote command authorization to all SP nodes that it serves. Also, the SP requires that the boot/install server has `rsh` and `rcp` command access to the CWS. In order to run with restricted `rsh` access, systems in this environment are limited to a single boot install server. For a single BIS, it must be the CWS.

The list of changes for the BIS and timing information includes:

- The entries in the CWS authorization files must be done before setup_server is run on the BIS node.
- On the BIS node, you need to edit /etc/sysctl.conf to include the following entries:
  - include /usr/lpp/ssp/sysctl/bin/instll.cmds
  - include /usr/lpp/ssp/sysctl/bin/switch.cmds
- Also on the BIS, you need to install the sysctl commands required for firstboot.cust customization.
- sysctld needs to be recycled on the BIS.
- After the node is installed, the entries for the BIS node can be put into the client node’s authorization files.

Depending on the network configuration and speed within the SP, you may not be able to install the system with a single boot/install server. If you do require multiple boot install servers while operating in this more secure, environment, you need to manually update the `rsh` and `rcp` commands authorization files with the correct values.

These updates would include:

- An entry in the CWS authorization files for each boot/install server.
- An entry in the authorization files on each node served by a boot/install server for that server.

1.7.2 DCE

Distributed Computing Services (DCE) provides a variety of common services that users need for development of distributed applications.

A group of DCE machines that work together and that DCE administers as a unit is called a cell.

A DCE environment is a group of one or more DCE cells that can communicate with each other. A cell becomes a part of a DCE environment when it obtains access to one or more global directory services in which the other cells in the environment are registered.

DCE provides a high-level, coherent environment for developing and running applications on a distributed system. The DCE components fall into two categories: Tools for developing distributed applications, and services for running distributed applications. The tools, such as DCE RPC and DCE

Threads, assist in the development of an application. The services, such as the DCE Directory Service, Security Service, and Distributed Time Service, provide the support required in a distributed system that is analogous to the support an operating system provides in a centralized system.

Distributed Computing Environment is a layer between the operating system and network on the one hand, and the distributed application on the other. DCE provides the services that allow a distributed application to interact with a collection of possibly heterogeneous computers, operating systems, and networks as if they were a single system.

1.7.2.1 SP trusted services

Security is not provided in a system by a set of security-specific services independent of all other services in the system. Rather, security is provided through each service or application being trusted to correctly enforce the security policies of the system. Each service that provides access to the system or to any operation or information in the system is responsible for enforcing the system security policies applicable to that service. Such services are called trusted services.

The SP trusted services satisfy the following requirements:

- Identification and authentication — All SP system servers authenticate the identity of their clients prior to providing services to those clients. If a client is to pass/receive sensitive information to/from a server, then the client must also authenticate the identity of the server prior to sharing such information with the server. All SP system client/server based services are capable of authenticating the identity of the users on whose behalf the client requests service.
- Discretionary access control - All objects and operations defined by each service are protected from unauthorized access. The SP system protects all SP service objects with well-defined, discretionary access control (DAC) policies.
- The installation and use of Kerberos V4 authentication within the SP system and SP trusted services is optional.
- The use (or non-use) of DCE security services is configurable to meet the requirements of individual security policies.
- Reasonable and secure defaults are provided for all configuration values and options in administering the system and its services.

In order to utilize SP security service interfaces, each SP trusted service defines:

- The service name for each of its servers
- The names of the groups used by the service for discretionary access control
- The attributes of the service names and groups for use in service configuration
- Any pre-defined members, service or machine principals, of groups used by its servers

These service and group names are specified in a service configuration file. PSSP provides a mechanism to override the standard configuration of service names. The service configuration information is used during the configuration of the service for DCE to create the following entities in the DCE registry and the DCE Cell Directory Service (CDS) database:

- A DCE organization and group whose members are the trusted service principals
- A DCE group with authority to administer the trusted services DCE ACLs
- The DCE principal for each service
- The DCE group(s) required for each service
- The DCE account associated with each service principal
- The DCE key file associated with each service principal
- The DCE CDS paths used to locate trusted services' DCE ACLs

The use of DCE for client-server authentication requires that the server's principal name and encryption key be contained in a file (for example, the key file) accessible by the server. There is one key file for each SP trusted service that contains the principal names(s) and encryption key(s) for a server. The server uses the information in the file to:

- Log in to DCE as the service principal
- Decrypt client credentials as part of the authentication process

The expiration of service principal keys is controlled by a DCE cell administrator by setting the password expiration policy for the primary organization for the service principal. The default policy is that keys will never expire.

If a password expiration policy is set for an SP trusted services principal's primary organization, then that service's principal's key (as stored in both the DCE registry and the service key file on the local system/node) must be changed prior to the expiration of the key; otherwise, the services will not be able to use the principal name and key in its local key file to log in to DCE and will not be able to authenticate the identities of the service's clients.

1.7.2.2 Authentication methods

SP trusted services authenticate service clients based on the authentication methods enabled:

- When DCE is the only authentication method enabled, SP trusted services exploit DCE authentication through the use of the SP Security Services interfaces.
- When compatibility is the only authentication method enabled, SP trusted services utilize the authentication methods used by the service in prior releases; if no authentication methods were used by the service in prior releases, then the service provides unauthenticated services compatible with prior releases of the trusted service.
- When both DCE and compatibility are enabled as authentication methods, SP trusted services are capable of utilizing any authentication method(s) used by the service in prior releases, or it may use DCE. SP trusted services whose clients and servers are not wholly contained within a single SP system partition also are capable of authenticating client requests through the use of the SP Security Services interfaces. In this case, authentication of the client's identity, via either DCE or the compatibility mechanism, is sufficient; the order in which the authentication mechanisms are used is not important.
- When no authentication methods are enabled, SP trusted services that are critical to the installation, administration, and operation of the system function without the use of a strong, distributed authentication mechanism.

If a server cannot reliably determine the set of authentication methods enabled for its local system partition, then the server rejects all subsequent requests, that otherwise might require authentication, until such time as the enabled set of authentication methods can be determined. For example, such a situation might occur if a server cannot load the libraries used to check the authentication methods, or if the library routines used to check the authentication methods return errors.

Since the set of authentication methods enabled within a system partition may be changed at any time by a system administrator, SP trusted services are able to dynamically adjust to such changes and process all subsequent requests according to the authentication methods as reset by the system administrator. If a trusted service already has connections with a client (or with multiple clients) at the time a change in authentication methods is detected, the existing connections continue to be serviced without requiring any re-authentication of the client; however, all new connections are authenticated according to the authentication methods enabled at the time the connection is established.

1.7.2.3 Cluster technology (RSCT)

Topology Services and the Reliable Message Service (RMS) both use connection less protocols (UDP/IP) for daemon to daemon communication within an SP system partition.

During configuration of security on the CWS, a DCE keyfile is created on the CWS for the topology services principal. Whenever an SP node is booted, the security configuration script on the node retrieves the topology services file from the CWS. The topology services daemons within the SP can then use the DES encryption key associated with the topology services principal in the file to calculate an encrypted checksum for each UDP message between any two daemons in the SP.

The system administrator should periodically change the key associated with the topology services principal in the keyfile. In order to change the key, the system administrator can use the `dcecp` command to generate a new key. Once the key is changed, the administrator must copy the keyfile to all nodes within the SP system.

Once the file has been copied to all SP nodes, the administrator needs to issue a refresh of the topology services subsystem in order for the key change to take effect. Any nodes to which the key file was not copied will not be recognized as available by the topology services subsystem and will be marked as *down*. Also, when the administrator changes the key, via the `dcecp` command, a new key and version number will be added to the keyfile; previous keys will remain in the keyfile. The administrator may want to periodically remove entries containing old keys from the keyfile.

1.7.2.4 Hardware monitor

The hardmon daemon runs only on the CWS; hardmon clients may run anywhere in the network. In prior releases, the hardmon service used only Kerberos V4 for client-server authentication and used a proprietary form of ACLs for authorizing Kerberos V4 principals to access hardmon objects. In the PSSP 3.2, hardmon supports client-server authentication based on the authentication methods configured for SP distributed services.

- DCE only: The hardmon service utilizes only the SP Security Services interfaces for client-server authentication and supports the use of a DCE ACL manager through the SP Security Services interfaces. Kerberos V4's ACLs are *not* supported.

- Compatibility only: The hardmon service supports only Kerberos V4 authentication in a manner compatible with pre-PSSP 3.2 systems, and authorization is based on the pre-PSSP 3.2 style ACLs using Kerberos V4 principals.
- DCE and compatibility: The hardmon service supports both the SP Security Services and Kerberos V4 interfaces for client-server authentication. Authorization is provided by both the use of a DCE ACL manager through the SP Security Services interface and through the use of pre-PSSP 3.2 style ACLs using Kerberos V4 principals.
- No methods enabled: No authentication checking is done.

1.7.2.5 sysctl

The sysctl daemon runs on every node within an SP and may run on other AIX systems anywhere in the network; sysctl clients may run anywhere in the network. In prior releases, sysctl used only Kerberos V4 for client-server authentication and used a proprietary form of ACLs for authorizing Kerberos V4 principals to access sysctl objects.

In the PSSP 3.2, sysctl supports client-server authentication based on the authentication methods configured for SP distributed services. The sysctl daemons and clients function as follows:

- DCE only: sysctl utilizes only the SP Security Services interface for client-server authentication and supports the use of a DCE ACL manager through the SP Security Services interfaces. Kerberos V4's ACLs are *not* supported.
- Compatibility only: sysctl supports only Kerberos V4 authentication in a manner compatible with pre-PSSP 3.2 systems. Authorization is based on the pre-PSSP 3.2 style ACLs using Kerberos V4 principals.
- DCE and compatibility: sysctl supports both the SP Security Services and Kerberos V4 interfaces for client-server authentication. Authorization is provided by both the use of a DCE ACL manager through the SP Security Services interface and by the use of pre-PSSP 3.2 style ACLs using Kerberos V4 principals.
- No methods enabled: sysctl does check the ACL files and performs client authentication.

1.7.2.6 System Data Repository

SDR daemons run only on the CWS; there is one SDR daemon for each partition within the SP system.

In prior releases, the SDR daemon allowed public reads of all SDR objects. In order to create or modify an SDR object, the SDR daemon required that the connection/message from the client be constructed, such that the client appeared to be the root user on a node within the SP system.

- DCE only: The SDR utilizes only the SP Security Services interfaces for client-server mutual authentication and supports the use of a DCE ACL manager through the SP Security Services interfaces. Kerberos V4's ACLs are *not* supported.
- Compatibility only: The SDR service supports only Kerberos V4 authentication in a manner compatible with pre-PSSP 3.2 systems, and authorization is based on the pre-PSSP 3.2 style ACLs using Kerberos V4 principals.
- DCE and compatibility: The SDR service supports both the SP Security Services and Kerberos V4 interfaces for client-server authentication. Authorization is provided by both the use of a DCE ACL manager through the SP Security Services interface and through the use of pre-PSSP 3.2 style ACLs using Kerberos V4 principals.
- No methods enabled: No authentication checking is done.

In PSSP 3.2, the SDR supports client-server authentication based on the authentication methods configured for SP distributed services.

1.7.2.7 Parallel environment

The pmd daemon authenticates the identity of the Parallel Environment client prior to servicing the client's request. A user is authorized to initiate a parallel application on a node, via Parallel Environment, if and only if the user is authorized to initiate an `rsh` and `rcp` command on that node with the given AIX user ID.

In prior releases, the pmd daemon authenticated the Parallel Environment client using an internal mechanism. This mechanism will continue to be supported when compatibility is enabled for trusted service authentication within a partition.

In PSSP 3.2, Parallel Environment supports client-server authentication based on the authentication methods configured for SP distributed services.

When DCE is enabled as an authentication method, the Parallel Environment client delegates DCE credentials to the pmd daemon if the delegation of credentials is allowed by the DCE principal's registry entry. If delegation of the user's credentials is not allowed or failed for any reason, then the pmd daemon continues with the authentication and authorization of the client without the delegated credentials. If delegation of the user's credentials succeeded, then the pmd daemon attaches the delegated credentials to the user process started by the pmd daemon.

- DCE only: The Parallel Environment client attempts to obtain delegable DCE credentials for the user through the SP Security Services interface. If credentials can be obtained, they will be passed to the Parallel Environment daemon on the target system.
- Client Authentication
  - Compatibility only: The Parallel Environment client calculates the encrypted user information as supported in pre-PSSP 3.2 versions of Parallel Environment. This information is then passed to the Parallel Environment daemon on the target system.
  - DCE and compatibility: The Parallel Environment client attempts to obtain delegable DCE credentials for the user through the SP Security Services interface; the Parallel Environment client also calculates the encrypted user information as supported in pre-PSSP 3.2 versions of Parallel Environment. The Parallel Environment client will pass the DCE credentials and the encrypted user information to the Parallel Environment daemon on the target system.
  - No methods enabled: The Parallel Environment client attempts to obtain delegable credentials for the user through the SP Security Services interface; however, since all attempts to obtain credentials or to authenticate a client's identity will fail, the Parallel Environment service will effectively be disabled.
- Daemon authentication
  - DCE only: The Parallel Environment daemon will authenticate the user's identity based on the DCE credentials passed from the Parallel Environment client. If delegable credentials were obtained from the client, then these credentials will be attached to the process prior to checking the user's authorization on the local node.
  - Compatibility only: The Parallel Environment daemon will authenticate the user's identity based on the encrypted user information passed from the Parallel Environment client.
  - DCE and compatibility: If DCE credentials were passed from the Parallel Environment client, then the Parallel Environment daemon will authenticate the user's identity based on these credentials. If delegable credentials were obtained from the client, then these credentials will be attached to the process prior to checking the user's authorization on the local node. If DCE credentials were not passed from the Parallel

Environment client or if the authentication of the user's DCE credentials failed, then the Parallel Environment daemon will authenticate the user's identity based on the encrypted user information passed from the Parallel Environment client.

 - No methods enabled: Parallel Environment utilizes only the SP Security Services interface for client-server authentication; however, since all attempts to obtain credentials or to authenticate a client's identity will fail, the Parallel Environment service will effectively be disabled.

To check the client's authorization to initiate processes on the server node, the pmd daemon uses the AIX library routines for checking the authorization of a user to initiate an rsh process on the server node. The pmd daemon checks the client's authorization as follows:

- Authentication checking performed by the daemon
  - DCE only: The pmd daemon uses the AIX kvalid_user() routine to determine if the client's authenticated DCE principal name is authorized to initiate a process on the daemon's node.
  - Compatibility only: The pmd daemon uses the AIX ruserok() routine to determine if the client's authenticated AIX user ID is authorized to initiate a process on the daemon's node.
  - DCE and compatibility: If DCE authentication succeeded, the pmd daemon will use the AIX kvalid_user() routine to determine if the client's authenticated DCE principal name is authorized to initiate a process on the daemon's node. If DCE authentication failed, then the pmd daemon will use the AIX ruse-rok() routine to determine if the client's authenticated AIX user ID is authorized to initiate a process on the daemon's node.
  - No methods enabled: Authentication fails, and the pmd daemon will reject the request.

1.7.2.8 GPFS

In PSSP 3.2, GPFS provides authentication for all TCP/IP based daemon-to-daemon communications based on the authentication methods for SP distributed services. It functions as follows:

- DCE only: GPFS daemons utilize only the SP Security Services interface for authentication.
- Compatibility only: GPFS daemons use the pre-PSSP 3.2 style protocol; no authentication is performed.

* DCE and compatibility: GPFS daemons utilize the SP Security Services interface for authentication.
* No methods enabled: GPFS daemons use the pre-PSSP 3.2 style protocol; no authentication will be performed.

1.7.3 SMIT panel

Figure 30 shows the new SMIT panel for configuring the authorization methods and enabling the authentication services. You need to access this panel to configure the basic settings.

To access this panel, you can use the fast path `smitty spauth_config` command.

```
                        RS/6000 SP Security

       Move cursor to desired item and press Enter.

         Select Security Capabilities Required on Nodes
         Create DCE hostnames
         Update SDR with DCE Master Security and CDS Server Hostnames
         Configure DCE Clients (Admin portion)
         Select Authorization Methods for AIX Remote Commands
         Configure SP Trusted Services to use DCE Authentication
         Create SP Services Keyfiles
         Enable Authentication Methods for AIX Remote Commands
         Enable Authentication Methods for SP Trusted Services
         Hardware Monitor DCE Objects
         Manage SP ACLs




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

Figure 30. Main SMIT panel for SP Security settings

In the main SMIT SP Security Panel, you can access the following:

* Select Security Capabilities Required on Nodes: This information is entered per partition. You input the System Partition name and the Authentication Methods, for example, DCE, Kerberos V4 (K4), or standard AIX (std).

- Create DCE hostnames: Runs the create_dce_hostname script. There are no parameters to input. Creates DCE hostname entries in the SP and node classes in the SDR.
- Update SDR with DCE Master Security and CDS Server Hostnames: This option allows you to update the hostname of the DCE Master Security and CDS Server in the SDR. You need to configure the Master Security Server and the CDS server before executing this step.
- Configure DCE Clients: Choose this option for configuring the admin portion of DCE clients on the control workstation. You need to input the cell administrator ID and the LAN profile ID. The cell administrator password is requested.
- Select Authorization Methods for AIX Remote Commands: For selection of authorization methods for root access to AIX remote commands. This information is entered per partition. You need to input the System Partition name and the Authentication Methods, for example, DCE, Kerberos V4 (K4), or standard AIX (std).
- Configure SP Trusted Services to use DCE Authentication: Runs the config_spsec script to configure SP Trusted Services to use DCE Authentication. You need to be logged into the DCE cell as the cell administrator.
- Create SP Services Keyfiles: Runs the create_keyfiles script to create keytab objects in the DCE Security database and to store the keys in to local keyfiles.
- Enable Authentication Methods for AIX Remote Commands: Choose this option to select the authentication methods to be enabled for AIX remote commands. This information is entered per partition. The authentication methods are: K5, K4, and std.
- Enable Authentication Methods for SP Trusted Services: This option is for selecting authentication methods to be enabled for SP Trusted Services. This information is entered per partition. The trusted services authentication methods are: DCE, K4, or blank (for none).
- Hardware Monitor DCE Objects: This option is for manipulating the Hardware Monitor DCE Objects for the control workstation.
- Manage SP ACLs: List the menus for managing DCE Access Control List for SP Trusted Services objects. Use this option to list, add, change, or remove ACL entries for one or more instances of SP trusted objects.

For more information on PSSP installation and configuration changes, refer to Section 1.1.3, “Installation” on page 4.

For more information about how to use security in your SP system, refer to the redbook *Exploiting RS/6000 SP Security: Keeping It Safe*, SG24-5521.

1.8 Serviceability

Serviceability is one of the most important characteristics of software programs. Good serviceability allows for subsequent analysis to determine the root cause of problems that affected the operation of a program and any correlation between problems that might have caused the malfunction.

To enhance SP and cluster software serviceability, the First Failure Data Capture (FFDC) facility is introduced in this version of PSSP, and some modifications have been done in the CE diagnostics tool.

The following two sections describe the FFDC facility and the CE diagnostics tool.

1.8.1 First failure data capture

FFDC is a facility to enhance SP and cluster software serviceability, so its function is oriented towards the internal components and applications of the cluster software and *not* towards the end-user programs. For this reason, it is mainly implemented in the topology and group services cluster components.

1.8.1.1 Objectives

The main objectives of the FFDC are:

- Provide interfaces to SP and cluster internal software components and applications to record sufficient information about failures.
- Provide correlation between related failures, as some failures might cause other failures.
- Ensure that key information about the failure is recorded in a consistent manner.
- Provide information to event management that simplifies the system administrator’s task to determine abnormal recording activities.
- Reduce maintenance windows by eliminating the need for problem regeneration.
- FFDC should not introduce workload overhead when no software failure conditions are encountered.

Note that although FFDC enhances software Reliability, availability, and serviceability (RAS), it does not prevent failures from happening. That is to say, FFDC is mainly intended to eliminate the need to wait for, or to force regeneration of failure, in order to gather information required for troubleshooting.

Enhancements introduced in PSSP 3.2 that improve the SP software serviceability that are associated with FFDC are:

Improved AIX error log facility — The AIX error log facility has been improved in several ways to work with the FFDC, basically by improving the error message content as follows:

- More flexibility in creating error log templates
- More precise error log templates
- Ability to record error information in an NLS-compliant manner

Improved error log templates — The error log templates have been improved to allow more meaningful error entries to appear in the AIX error log, especially in the response field where messages, such as “Contact IBM Service” and “Perform Problem Determination Procedures”, are no longer acceptable. Likewise, Failure fields, Cause fields and Details fields are improved by replacing general statements by more accurate and meaningful ones.

FFDC Error Stack — The FFDC Error Stack is modeled after the Error Stack used by SP Perspectives. It is a facility that allows programs to associate error information passed from various functions that operate in a nested manner. Each function is required to indicate an error condition to the invoking module using a hierarchical error reporting mechanism.

The primary module is responsible of recording the Error Stack and setting the Error Stack variable to its process ID. The mechanism includes the following components:

FFDC Error Stack file — A recording that contains correlated error entries that are correlated by using unique tokens called *FFDC Error IDs.*

FFDC ID — The FFDC ID indicates the location and instance of a failure report and, as such, it indicates where the information is recorded (either in the FFDC Error Stack or the AIX error log), the entry’s location within the FFDC Error Stack or the AIX error log, the time that entry was recorded and the version code and IP address of the node where the record is residing.

The FFDC was designed to be a 42-character string, as illustrated in Figure 31. The FFDC ID can be decoded by the `fcdecode` command.

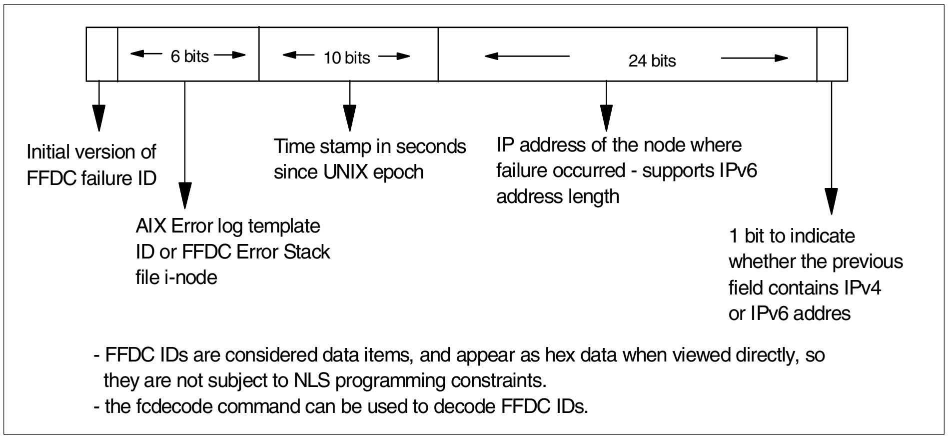

Figure 31. FFDC ID

FFDC log monitor — The FFDC log monitor is an Event Management Resource Manager application that provides a status log for logs that exist in the /var/adm/SPlogs directory on a node. Basically, it provides one resource variable to Event Management representing the current status of the log. Variable values can be one of the following:

Active - The software component that uses this log file is active and in normal operation mode (normal traffic).

Inactive - The software component does not seem to be active.

Reset - The software component appears that it has truncated, replaced, or switched to a next version of the log file.

Investigate - The software component seems to be using the log file heavily, and the log traffic is above normal.

1.8.1.2 Packaging, installation and coexistence

In this section, the FFDC packaging, installation, migration, and coexistence issues are discussed.

Packaging

The FFDC installable files are included in the rsct.clients.rte.1.2 fileset. Table 2 lists the FFDC installable files.

Table 2. FFDC installable files

Description	Location	Perms.	Owner
Header file	/usr/sbin/rsct/include/ct_ffdc.h -> /usr/include/rsct/ct_ffdc.h	0444	bin:bin
API shared C language library	/usr/sbin/rsct/lib/libct_ffdc.a -> /usr/lib/libct_ffdc.a	0444	bin:bin
FFDC Directory	/var/adm/ffdc	0777	root:system
Stack directory	/var/adm/ffdc/stacks	1777	root:system
Dump directory	/var/adm/ffdc/dumps	1777	root:system
CLI commands	/usr/sbin/rsct/bin/*command* -> */usr/bin/command*	0755	bin:bin
End-user commands	/usr/sbin/rsct/bin/*command* -> */usr/sbin/command*	0755	bin:bin
Msg catalog	/usr/lib/nls/msg/locale/ffdc.cat	0444	bin:bin

Installation

The *rsct.clients.rte* fileset is used to install the FFDC environment that requires *rsct.clients.sp* and *rsct.basic.rte* filesets as prerequisites. These filesets must be installed on any node where FFDC is going to be used.

FFDC creates a directory for its own, which is /var/adm/ffdc in the /var file system. Usually, this file system gets filled up quickly because most of log files (whether AIX, SPlogs, or others) use this file system. Therefore, we recommend that you create a separate file system for use by FFDC and, of course, to be mounted on /var/adm/ffdc. This will prevent FFDC from filling up the /var file system.

FFDC is a node-centric utility. FFDC administration duties have to be performed on a per-node basis, but some of these tasks may be automated through cron.

Migration

Because FFDC is a node-centric utility facility, it incorporates a built-in versioning control. This allows the future releases of FFDC to understand the FFDC ID generated by previous FFDC releases.

FFDC installation or migration does not impact the PSSP migration of the whole SP system or the migration of individual number of nodes from previous versions of PSSP or RSCT. The SP system can still be migrated in stages.

Coexistence

There is no interdependency between FFDC facilities on different nodes.

FFDC IDs have version identifiers that will allow future releases of FFDC to understand and work with them so that they can coexist, and there will be no need to migrate FFDC on all nodes to the same level.

1.8.1.3 Implementation

FFDC provides C programming libraries and user commands to facilitate serviceability. The C library interfaces comply with the POSIX P 1003.1 standard, and the command interfaces comply with the POSIX P 1003.2 utilities guidelines.

FFDC utilities are completely portable to other AIX-based platforms; so, they can be installed on stand-alone RS/6000 boxes. So far, FFDC utilities are not portable to other UNIX platforms; however, they can be ported by recompiling the source code on UNIX platforms that support the BSD System Log utility.

FFDC interfaces applications requiring no special security measures. These interfaces do not need resources from external nodes, therefore, there is no need to use DCE or Kerberos. AIX security measures are sufficient.

Internationalization using NLS is achieved by FFDC, as the FFDC Error Stack records failure information using keywords and codes that are internal to FFDC. These keywords and codes are interpreted and translated by the `fcstkrpt` command into the human language in the proper locale.

FFDC Error Stack is implemented as a binary file that resides in the /var/adm/ffdc/stacks directory. This file is created when the first entry is written to the FFDC Error Stack and *not* when the stack is created or inherited (which avoids cleanup of empty stacks).

The Error Stack file is of fixed length (circular file). Spaces within the file are allocated when the file is created. Since the FFDC Error Stack file is binary, it can not be opened by an editor. Instead, FFDC commands (`fcstkrpt` and `fcreport`) extract information from the file and display it to the caller.

Figure 32 illustrates the Error Stack file.

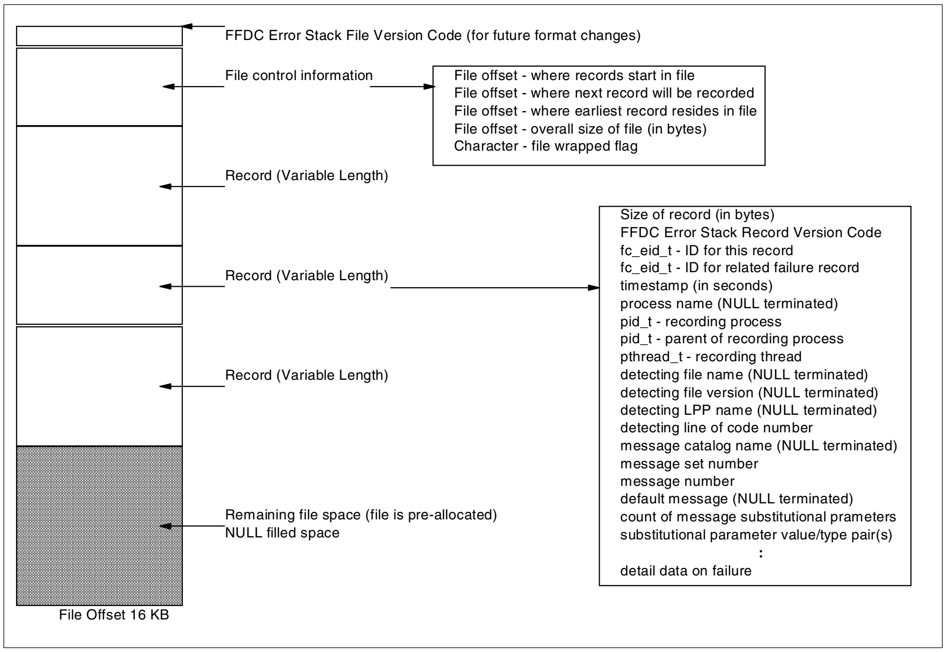

Figure 32. FFDC Error stack file

Process environment variables set and used by FFDC are:

FFDCSTACK - Absolute path name of the FFDC Error Stack file that will be the same value for the process that creates it and the processes that inherit it.

FFDCPID - PID value. If the value is the same as the PID of the current process, the process has established an FFDC Environment.

FFDCORIG - PID value. This indicates the process that created the FFDC Error Stack.

FFDCPNAME - Name of the process. If the value is the same as the name of the current process, the process has established an FFDC Environment. This is used only to generate a dump file name.

FFDCADDR - IPv4 or IPv6 address. This is a base address of one of the node's IP interfaces. It is encoded in base-64 character notation. This is used to generate an FFDC ID and identifies where in the SP or in the cluster the failure occurred.

FFDCSDISABLE - If set to a non-null value, will prevent FFDC from creating FFDC Error Stacks. Used when file system space is low or FFDC is misbehaving.

FFDCDEBUG - Absolute path name of a trace output file. If set, FFDC writes ASCII-formatted debug information to a circular text file with this name.

Figure 33 illustrates the procedure to set up an FFDC environment.

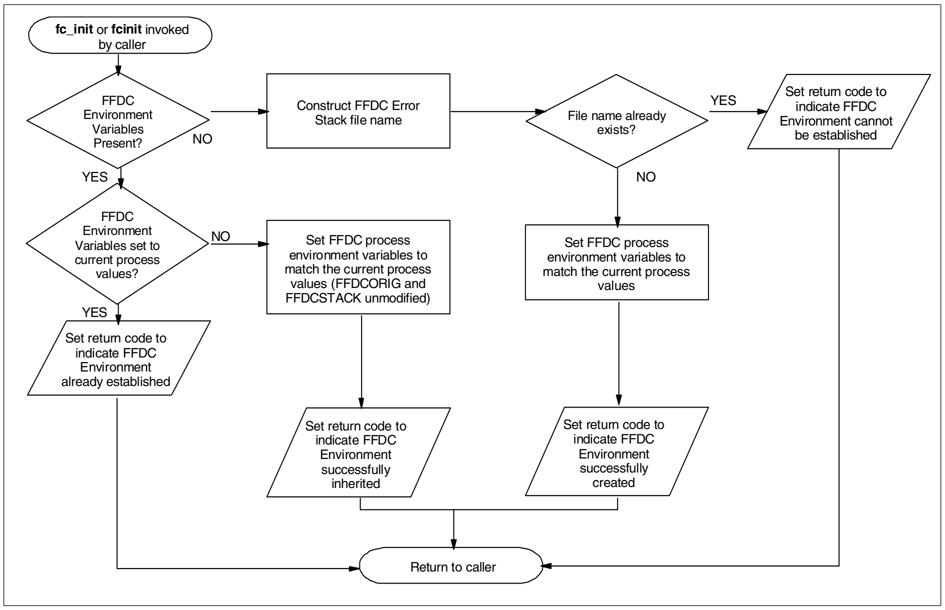

Figure 33. FFDC Error Stack environment setup

Processes create their own FFDC Error stack files, and, as such, they do not share a global or common Error Stack file. When a process creates an FFDC Error Stack, FFDC interfaces set up some FFDC environment variables, and when a process inherits an FFDC Error Stack, FFDC interfaces verify that the process environment variables exist and modifies their values to suit the child process.

Figure 34 on page 62 illustrates the procedure to create an FFDC Error stack file. As shown in this figure, the default file size of the FFDC Error Stack is 16 KB. However, if the file system does not have 16 KB + 5% of free space, the file size is decreased by 2 KB. That is to say, 14 KB file size is attempted, and so, on until 8 KB file size, which is the final attempt before an error message is returned indicating that an FFDC Error stack cannot be created.

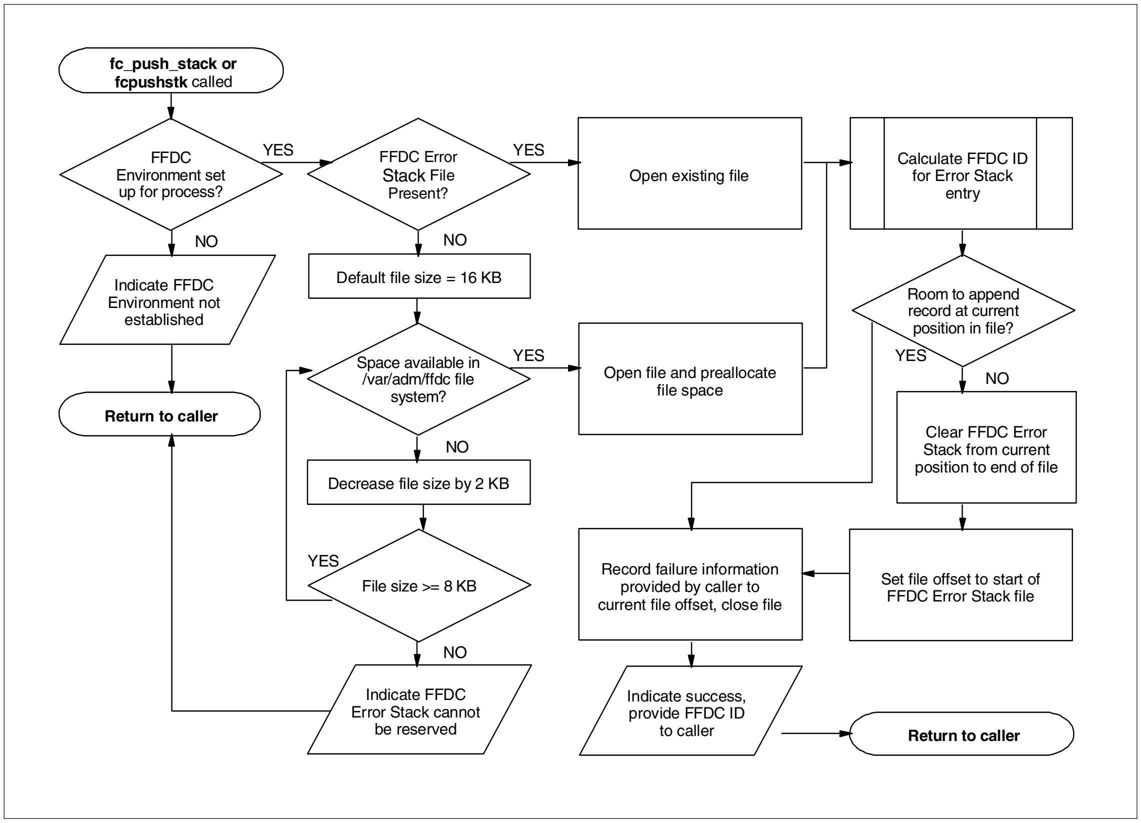

Figure 34. Error Stack file creation procedure

1.8.1.4 End-user commands and example

Commands that can be used by the end user are:

`fcdecode` — Decodes information stored in an FFDC failure ID

`fcstkrpt` — Displays FFDC Error Stack contents or an individual records from an FFDC Error Stack

`fcreport` — Displays entire lists of failures related to an FFDC ID

`fcclear` — Removes FFDC Error Stack files and error data files

`fcfilter` — Extracts FFDC IDs from input stream

`fccheck` — Performs basic troubleshooting on the FFDC utilities

For more information about commands, refer to *First Failure Data Capture Programming Guide and Reference*, SA22-5474.

The following is a simple example demonstrating how the FFDC Error Stack would be created and used.

Let us assume that the process hierarchy shown in Figure 35 exists.

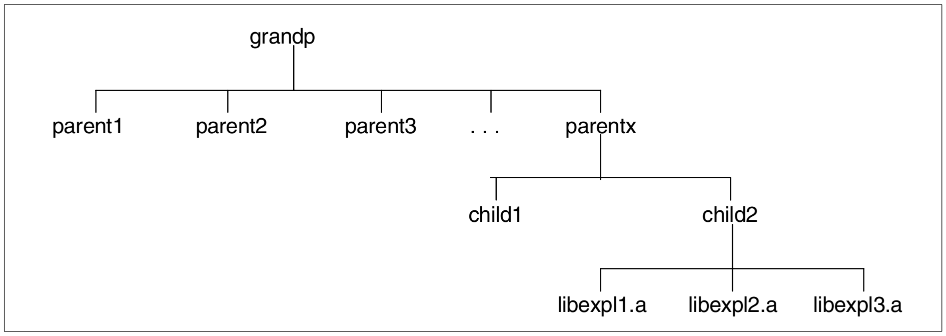

Figure 35. Example - Process hierarchy

The *grandp* process code initializes FFDC upon its start; so, it runs the `fcinit` interface to create the FFDC Error stack as shown in Figure 36.

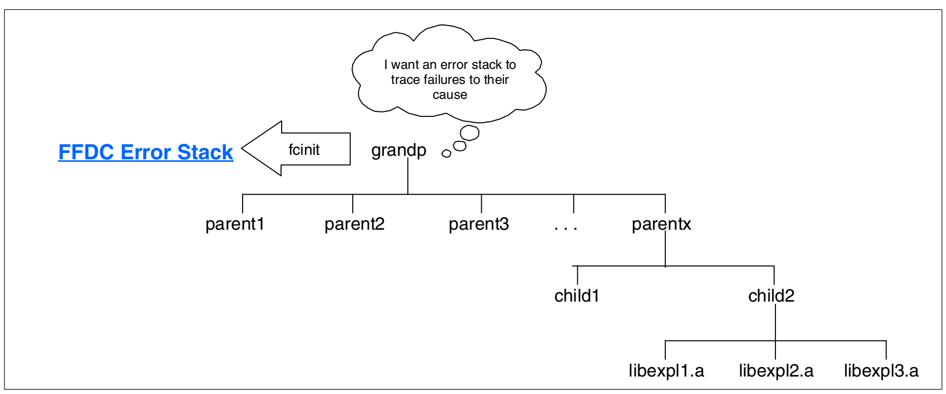

Figure 36. Example - grandp process initializing FFDC

Now, descendent processes establish the FFDC environment by inheriting the existing FFDC Error Stack environment as shown in Figure 37.

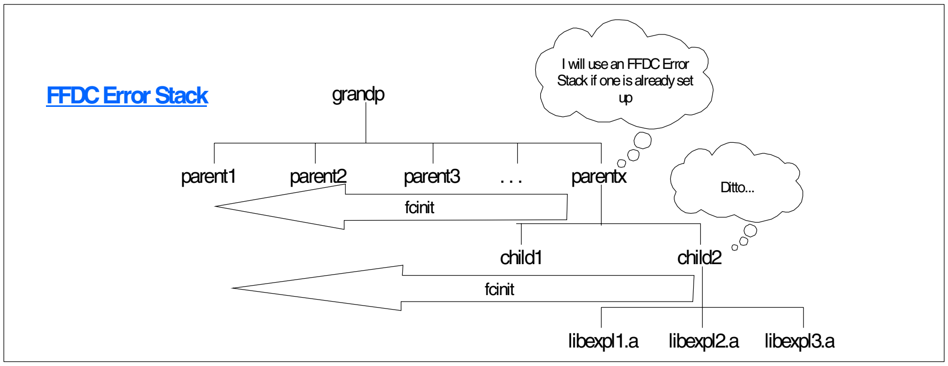

Figure 37. Descendent processes inherit the FFDC Error Stack

FID `a` (or FFDC ID `a`) is generated due to the failure of process *grandp*, saying that it could not start the *parentx* application and that the related FID is `z`.

FID `z` says there is a failure to start the *parentx* application due to the configuration failure of the *child2* application and that the related FID is *y.*

FID `y` is a child2 failure due to user `mdevine` not authorized to access configuration data and the related FID is `x`.

FID `x` is a principle authorization failure due to user `mdevine` not being authorized for the requested function, and the related FID is `none` as shown in Figure 38.

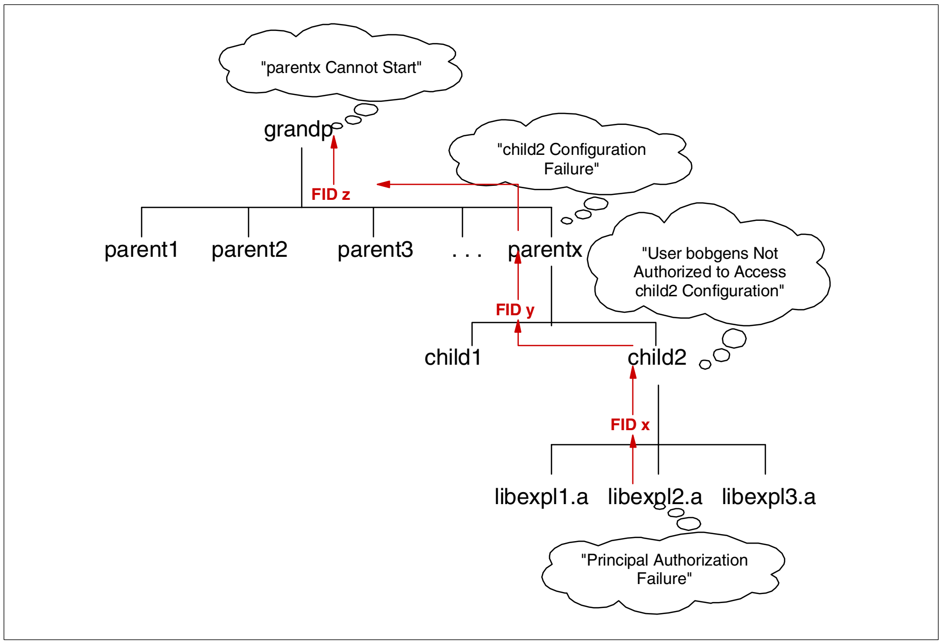

Figure 38. Example - FFDC IDs correlation

1.8.2 CE diagnostics

The CE diagnostics tool is intended to be used by the IBM CE for the SP system hardware at the customer site when the system is just delivered to the customer or when a new frame is to be added to an existing up and running system.

The primary objective of the CE diagnostics tool is to test the functionality of the SP frame(s) and each node within these frames before the system or the new frame(s) are handed to the customer. It does this by running the suggested processor, memory, hard disks, and adapters hardware tests.

Before PSSP 3.2, the CE diagnostics used to be run from an AIX RISC laptop, which the CE connects to each frame one at a time by the RS232 and Ethernet ports. The AIX RISC laptop provides an isolated environment on its own with the SP frames. The CE can set up temporary IP addresses for their laptop and the nodes or use the customer IP addresses and hostnames.

The AIX RISC laptop has been in use since the SP1, but currently it is obsolete and no longer manufactured; moreover, the latest level of supported AIX is 4.2.1; so, the laptop is not able to test newer hardware that is supported only by AIX 4.3.3 or later, such as the new switch adapters and the new high and wide nodes.

In PSSP 3.2, the CE diagnostics is ported to the control workstation where it will be run. For a new system, the control workstation will be delivered to the customer site preloaded with AIX and PSSP and a set of test IP addresses. Frames will be connected to the control workstation at a time, the nodes will be network-booted, and the CE will invoke the menus and run the proper diagnostics. Several nodes can be network-booted at a time.

It is worth to recall that when a new frame is added, the CE diagnostics must be run in a maintenance window, as the new nodes will boot in diagnostic mode and, therefore, this outage time should be planned ahead with the customer.

The scope of tests are:

- Processor and memory
- Hard drives
- Switch adapters, switch connectivity and functionality
- All other adapters

1.8.2.1 Objectives

The PSSP 3.2 CE diagnostics tool was designed to satisfy the following:

- Reduce the burden on the CE in installing and setting up software so that the CE can finish their job in one day and avoid returning another day to the customer, which adds to the outage time of the system.
- A tool that supports all configurations, new systems, and expanding existing systems.
- Retain the test suite that the CE is used to seeing and is familiar with, which is a menu-driven procedure.
- Reduce the need for customer involvement.
- Provide cleanup scripts to run before handing the system to the customer.

Using the menu driven procedure, the CE is able to perform the following steps:

1. Define/read system, frame, node, and switch configuration. The CE has to specify the number/type of frames, tty information, node definition/type, and switch definition/type.
2. Test frame power, control, and monitoring.
3. Gather Ethernet hardware addresses.
4. Launch SP Perspectives hardware GUI and LED/LCD display to monitor nodes status.
5. Network boot the nodes from the control workstation.
6. Check the results of node diagnostics and hardware configuration.
7. Test the switch fabric connections and switch clocking.
8. Check results of the `Estart` command.
9. Cleanup by removing test specific data.

The current version of CE diagnostics does not cover the S70, S7A, or S80 nodes, which will continue to be tested by a ThinkPad the same way it used to be tested before PSSP 3.2.

1.8.2.2 Packaging and installation

This component will follow the standard packaging practices. The following fileset will install the CE diagnostics tool:

```
ssp.cediag                3.2.0.0    C    SP CE Diagnostics
```

Files installed when installing CE diagnostics are shown in Figure 39.

```
ssp.cediag 3.2.0.0    /usr/lpp/ssp/cediag/cws/setup_sptest
                       /usr/lpp/ssp/cediag/node/loopled
                       /usr/lpp/ssp/optlevel.ssp.cediag
                       /usr/lpp/ssp/cediag/node
                       /usr/lpp/ssp/cediag/cws/user_notes
                       /usr/lpp/ssp/msgmaps/cediag.sptest.map
                       /usr/lpp/ssp/cediag/cws/user_info
                       /usr/lpp/ssp/cediag/cws/release
                       /usr/lpp/ssp/cediag/cws/sptest
                       /usr/lpp/ssp/cediag/cws/rcmd
                       /usr/lpp/ssp/cediag/cws/diag.sptest
                       /usr/lpp/ssp/cediag/cws/sptest.steps
                       /usr/lpp/ssp/cediag/cws
                       /usr/lpp/ssp/cediag/node/clktest
                       /usr/lib/nls/msg/en_US/cediag.cat
                       /usr/lib/nls/msg/en_US
                       /usr/lpp/ssp/cediag/cws/start_cediag
                       /usr/lpp/ssp/cediag/cws/recovery
                       /usr/lpp/ssp/cediag/node/diag.auto.odmadd
                       /usr/lpp/ssp/cediag/cws/sptest.stepsubs
                       /usr/lpp/ssp/cediag/node/startswap
                       /usr/lpp/ssp/cediag/node/diag.auto
                       /usr/lpp/ssp/cediag/cws/cleanup_sptest
                       /usr/lpp/ssp/cediag/cws/diag.client
                       /usr/lpp/ssp/cediag/cws/s1open
                       /usr/lpp/ssp/cediag/cws/diag.dsh
                       /usr/lpp/ssp/cediag/cws/sptest.subs
```

Figure 39. CE diagnostic files installation

This component is installed in the standard way. With new systems, a new CWS image will be created with pre-configured NIM and SPOT, along with test IP addresses and frame data to save time for the CE.

1.8.2.3 Dependencies

This release of CE diagnostics is intended to run only on SP control workstations that are supported on PSSP 3.2. The CWS should satisfy the following:

- Acceptable RS/6000 model (F50 in any of its three possible configuration, 43P150).
- Enough disk space and memory.
- Available serial port(s) to connect SP frames.
- Graphics adapter, monitor, keyboard, and mouse.

Software dependencies

CSS fsd flag — The SP Switch fault service daemon needs a parameter that would allocate the minimum buffers. When Estart is run from the CE Diagnostics with this flag, the fsd daemon is expected to fit into the 32 MB network boot image.

System Code — Previously, the cediags used private copies of Estart.sw and Eclock. In PSSP 3.2, the CE diags will only use the standard system code.

Results — Previously, the CE diagnostics filtered the results in out.top looking for relevant failures to display to the CE. In PSSP 3.2, it is expected that CSS will manage the output and send the CE information that css finds important for the CE to know, whether failures, debug information, or success.

1.8.2.4 Procedures

This section outlines the procedure for running CE diagnostics for a new system and for extending a system (new frame added to an existing system).

New system installation

Figure 40 contains information about new system installation.

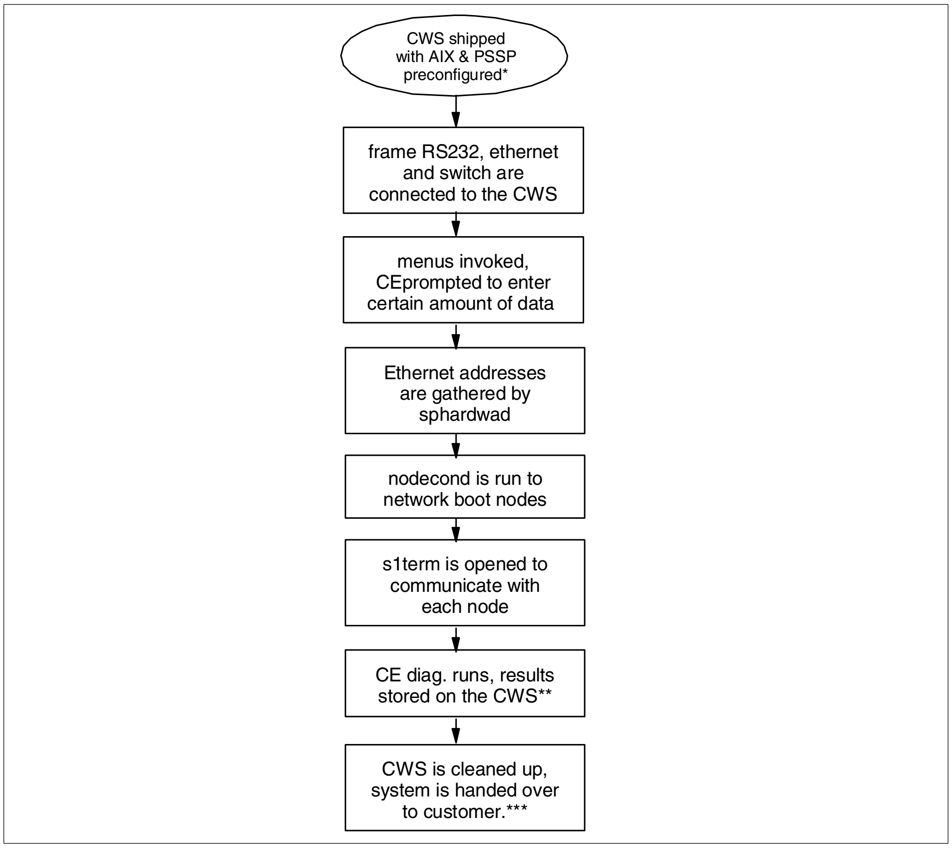

Figure 40. CE diagnostics procedure for new system

- * The image has SPOT built with SDR predefined with the frame and node information except ethernet hardware addresses. /etc/hosts is also preconfigured.
- ** CE diagnostics replaces the standard diagnostic program and runs when the node is booted.
- *** A new complete PSSP installation is required. Files affected by the cleanup script are /etc/hosts, NIM database, SDR system, and partition classes.

New frame added to existing system

Figure 41 outlines steps to take in order to add a frame to an existing system followed by some screen snapshots of the CE diagnostics tool.

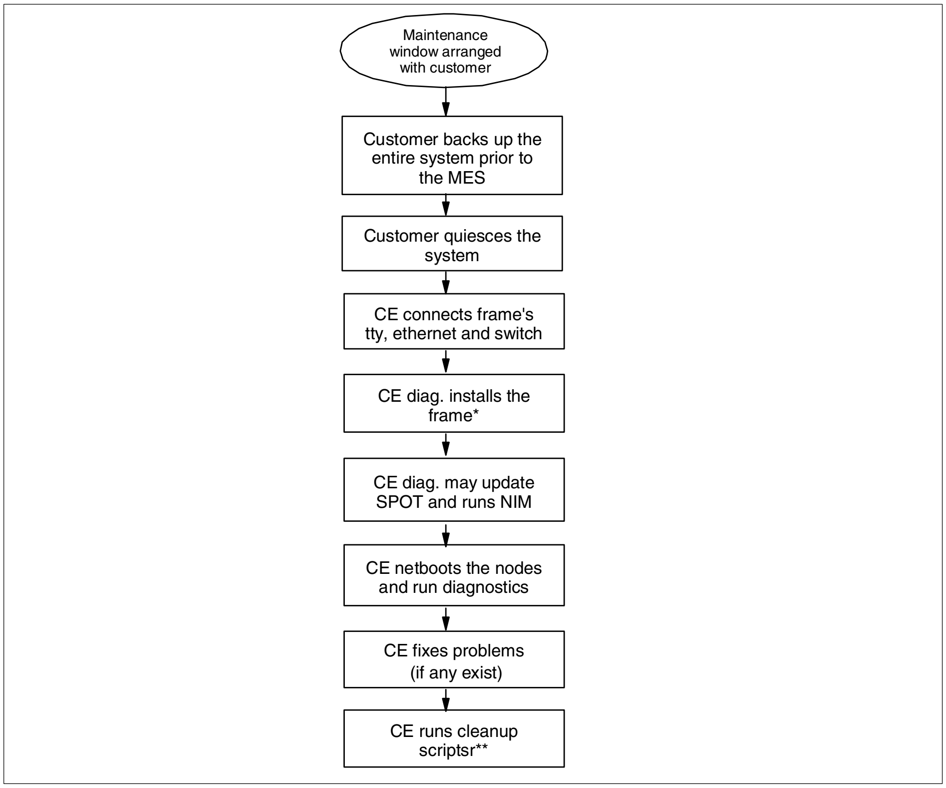

Figure 41. CE diagnostics procedure for an existing system

* CE diagnostics installs frame by running `spframe`, `spethernet`, `spadapters`, and `sphardware`.

**Again, the customer needs to make a complete PSSP reinstallation of the nodes in the new frame.

Figure 42 shows the output that results after you issue the following command: `/usr/lpp/ssp/cediag/cws/start_cediag`

```
Getting configuration information...

Getting configuration information...
Checking hardware definitions...

Checking prerequisites to run sptest...
Checking PSSP code installation and levels...
Checking NIM database basic definitions...

_________________________________________________________
            ***  SP System Installation Menu  ***
             Copyright 1994-1999 - IBM Corporation
Version 2.90
Current Config: Logical Frames: 1       Switch: SP_Switch

_________________________________________________________
 1) Frame Configuration
 2) Verify Frame Controls
 3) Gather Ethernet Hardware Addresses
 4) Start System Monitor
 5) Verify Processor Nodes
 6) Check Node Diagnostics
 7) Verify Switch Feature
 8) Check Switch Results
 9) Verify Switch Clocks
 10) Finish Frame

 a) Auto-continue until unsuccessful
 f) Finish
 h) Software Help        u) User Notes

----------------------------------------------------------
 Please enter "1-10 a f h u"(Default=1):
```

Figure 42. start_cediag command output

For example, choosing Option1 brings the output shown in Figure 43, where you can add a new frame, accept the current configuration, or add new information.

```
 Please enter "1-10 a f h u"(Default=1):1
Reading current frame configuration...

________________________________________________________

                Frame Definition Menu
 (Enter frame number to modify information for that frame.)


 Frame# Status           TTY_location Frame_type CWS_LAN_adapter
Node_IP_addr
 Switch_IP_addr
 ------ ---------------- ------------ ---------- ---------------
---------------
 ---------------

________________________________________________________
 1     Customer_use    00-00-S1-00  SP_Switch




 +/-) Scroll forward/backward
 a) Add new frame
 b) Back (no update)
 n) Next   (update)
 s) Start new system (Remove existing frame information)
----------------------------------------------------------
 Please enter "1 + - a b n s"(Default=a):
Please enter "1-10 a f h u"(Default=1):1
Reading current frame configuration...

________________________________________________________
```

Figure 43. Option 1 of start_cediag menus

Refer to the *Maintenance Information Manual* (MIM) for full details about the procedures and different screens.

1.8.2.5 Performance

Boot time varies from node to node according node type. It takes about five minutes to boot a thin node, then about five minutes to do the diagnostics; so, roughly it takes about one hour to complete diagnostics for 16 nodes-frames.

Wide and high nodes take longer to boot. It takes about five minutes to test the switch only, and about 10 minutes to test the frame base power.

Chapter 2. SP Switch support

The SP Switch2 and SP Switch2 adapter are the next step in high-performance communication technology available for the IBM SP system. The SP Switch2 board and adapter provide the means to exchange information at a very high data rate among a large number of high-performance SMP nodes. The SP Switch2 fabric can scale up to 256 compute nodes (16-way POWER3 SMP high * 256 nodes = 4096 processors). The SP Switch and SP Switch-8 are also supported in PSSP 3.2.

For more information about Hardware support, refer to *RS/6000 SP: Planning, Volume 1, Hardware and Physical Environment*, GA22-7280.

SP Switch2 system will be supported in the following phases:

- PSSP 3.2 GA release

  Single Switch System: Single Adapter, Single Port, Single Switch-plane

- Future PSSP releases

  Double Switch System: Dual Adapter, Single Port, Dual Switch-planes

  Quad Switch System: Dual Adapter, Dual Port, Quad Switch-planes

2.1 Requirements and limitations

PSSP 3.2 supports SP Switch, SP Switch-8, SP Swich2, SP Switch routers and SP-Attached servers. But, SP Switch2 hardware is only supported on POWER3 high nodes running PSSP 3.2 or later. If you use SP Switch2, you cannot have any other nodes or SP-attached Servers or SP Switch Routers. The system partitioning function is not supported in PSSP 3.2 when the SP Switch2 is installed.

Note

In an SP system with SP Switch2:

1. All nodes need the PSSP 3.2 (or later) level.
2. All nodes exist in the same, singular default system partition.

In an SP system with SP Switch, nodes can run at different levels of PSSP including PSSP V3.2.

The system can be partitioned.

If you have a partitioned SP system and plan to install the SP Switch2, then the SP system must be re-configured as a single system partition and migrated to PSSP 3.2 before the SP Switch2 can be installed. We recommend the following three-step for migrating from previous PSSP levels to PSSP 3.2 with SP Switch2:

1. Migrate CWS and nodes to PSSP 3.2.
2. Once all nodes are at PSSP 3.2, reconfigure the SP to a single system partition (if not already at a single system partition).
3. Install the SP Switch2 hardware.

2.2 SP Switch hardware

This section describes the new SP Switch2 hardware, which is supported only in PSSP 3.2 or later. This new hardware provides you higher bandwidths and higher reliability, availability, and serviceability (RAS).

2.2.1 RS/6000 SP Switch and adapter evolution

Figure 44 on page 77 shows the SP Switch system historical outline.

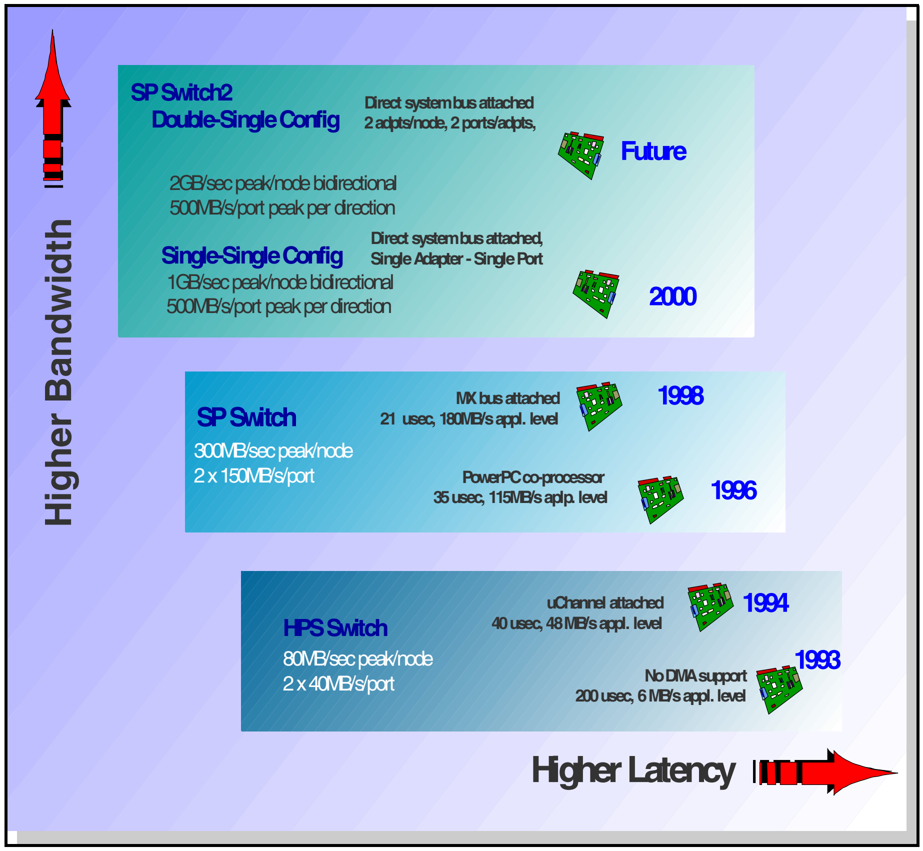

Figure 44. SP Switch subsystem evolution

First of all, you need to know about the types of switches and switch adapters. The adapters corresponding to these switches are shown in Table 3.

Table 3. Switch and switch adapter matrix

Switch Type	Adapter Type	Comments
HPS Switch (80MB/s peak/node)	HiPS Adapter-1(F/C 4017)	not supported in PSSP 3.2
	HiPS Adapter-2(F/C 4018)	
SP Switch (300MB/s peak/node) [SP Switch-8]	SPS Adapter (F/C 4020)	for MCA node
	SPS MX Adapter (F/C 4022)	for 332 MHz SMP nodes
	SPS MX2 Adapter (F/C 4023)	for POWER3 nodes

Switch Type	Adapter Type	Comments
SP Switch2 (2 * 500MB/s peak/port) bidirectional	SPS2 Adapter (F/C4025)	Single-Single for POWER3 SMP high nodes
	Future plan	Double-Single support

Note

- The 8-port SP Switch-8 (F/C 4008) provides switch functions for up to eight processor nodes.
- For upgrades to greater than eight node support, the SP Switch-8 is replaced by the 16 port SP Switch (F/C 4011). However, the 16 port switch is not supported in short frames (model 500 and F/C 1500).
- The SP-Attached Server and POWER3 High Nodes cannot be attached to the SP Switch-8.

High performance switch adapters are not compatible with any of the SP Switch adapters. They cannot coexist in the same system configuration.

2.2.2 SP Switch2 switch

Like the SP Switch, the SP Switch2 has 32 ports - 16 ports for switch-to-node connections and 16 ports for switch-to-switch connections. Management of the SP Switch2 network is the same as the SP Switch, using service packets over the switch plane. This section describes some of the most important hardware characteristics of the SP Switch2 hardware.

TOD synchronization

The SP Switch2 is designed with multiple clock sources in the system. There is no master clock unlike HPS or SP Switch. Therefore, instead of having clock selection logic, SP Switch2 switch has a separate oscillator for each switch chip. The TOD (Time Of Day) logic on the SP Switch2 has also been significantly redesigned to simplify both coding and system requirements as well as to improve the accuracy of the TOD across the system.

J-TAG interface

An additional interface, J-TAG interface, is added to the SP Switch2. This new interface will allow the supervisor to perform new functions, such as writing initialization data to the switch chips and reading error status information back from the switch chip.

Adaptive routing

To support the double-port, a number of routing enhancements were added in SP Switch2, namely *adaptive routing* and *multicast packets*. Adaptive routing will allow the switch chip to determine which output port to route the packet to based on the route nibble. Multicast packets give the switch the ability to replicate and distribute packets to a predefined groups of nodes.

2.2.3 SP Switch2 adapter

The SP Switch2 adapter encompasses many new hardware changes. They include:

- Up to two adapters. Each adapter contains two switch ports and can be placed in a single POWER3 SMP high node.
- Faster interface, data paths, and microprocessor.
- New data memory component.

Using multiple ports and adapters in a single node, along with the new SP Switch2 adapter, will increase the overall bandwidth of the switch adapter and decrease the latency of switch communications on the POWER3 SMP high node.

Hardware changes

The design of some chips are changed, and a new chip is also added. This section describes the new SP Switch2 chip design.

Figure 45 shows an SP Switch with MX adapter node overview.

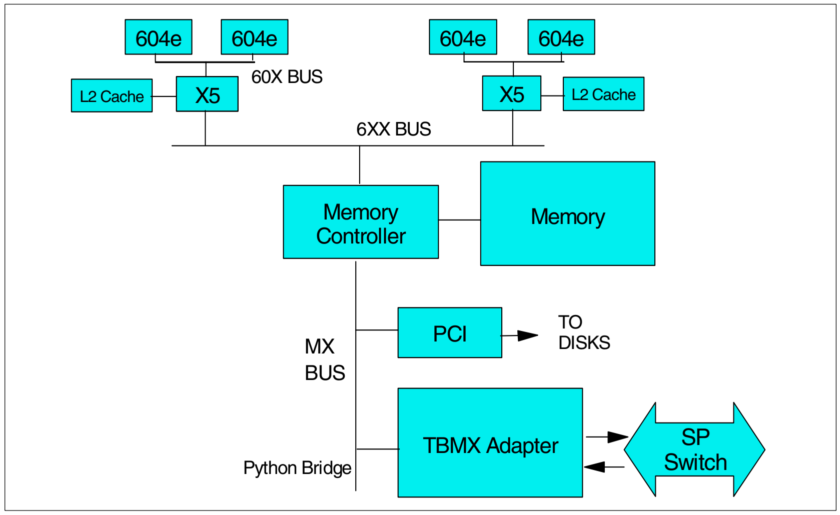

Figure 45. SP Switch MX adapter with 332 MHz SMP node

Figure 46 shows the new SP Switch2 adapter hardware structure.

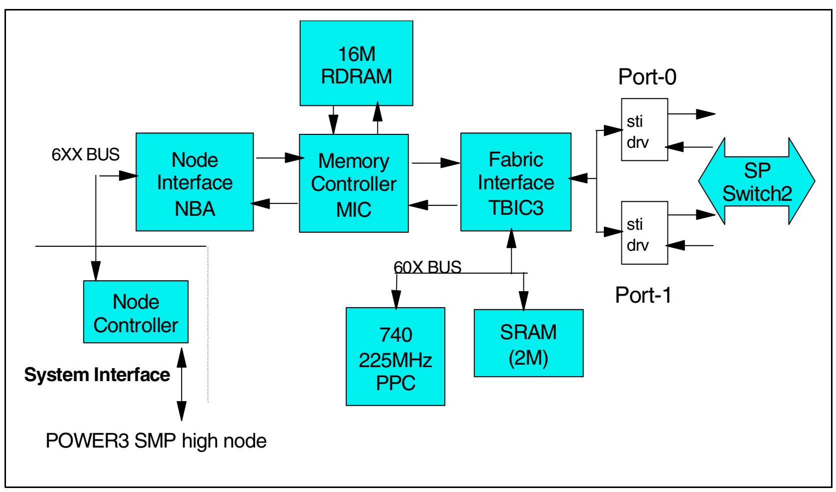

Figure 46. SP Switch2 adapter hardware structure

The SP Switch2 adapter connects to the system bus interface directly. This design solves the performance bottleneck caused by the BUS contention and enables switch traffic to proceed at higher bandwidths and lower latencies.

The TBIC3 chip moves data from the switch chip onto the adapter. Its function is very similar to those provided by TBIC on the TB3 adapter or the TBIC2 on the TB3MX adapter. The primary enhancements are its faster data transfer path (500 MBps) and the addition of packet reassembly hardware in the chip.

The NBA chip moves data between the adapter and POWER3 SMP node's 6xx bus. Its function is similar to what provided by MBA supplied on the TB3MX. The primary change is that it gives the adapter a memory type bus interface rather than I/O style used in previous adapters.

The MIC chip is responsible for either passing information between the TBIC3 and NBA or for giving access to the adapter's data memory component. This is a new function not available on previous adapters.

The 740 microprocessor executes the microcode on the adapter and is responsible for passing control information between the adapter and the software on the POWER3 SMP node for encoding or formatting packet headers, for building packet routing information, and for handling error conditions.

2.2.4 SP Switch2 configurations

This section describes the supported configurations in SP Switch2 environments. There is currently one supported SP Switch2 Single-Single configuration containing only high nodes.

SP Switch

Figure 47 shows the SP Switch configuration.

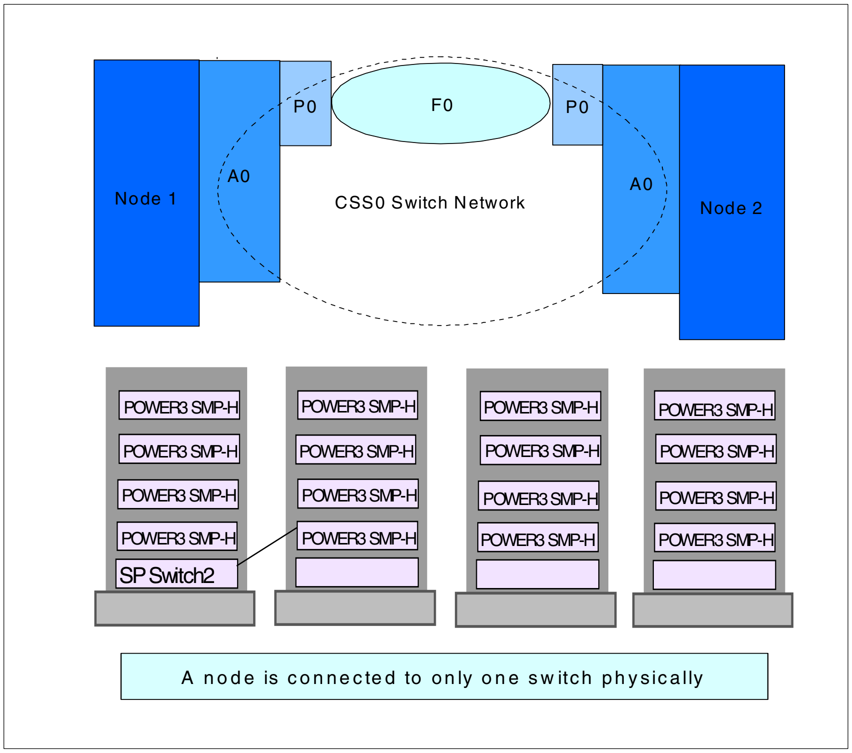

Figure 47. SP Switch high node switch configuration

In the SP Switch system, the nodes have been labeled with their relative switch node numbers, or switch port numbers, for the shared switch in the first frame.

SP Switch2 Single-Single

For SP Switch2, the configuration shown in Figure 47 is again a supported switch configuration and called an SP Switch2 Single-Single configuration. However, the switch port numbers are not predefined. In other words, the nodes may be attached to any node port of the switch in the first frame. Each node would have a single SP Switch2 adapter installed. An SP Switch2 adapter has two switch ports, and the first port on each adapter is to be attached to the switch in the first frame of the SP Switch2 system. The second port of each adapter is unused in this configuration.

SP Switch2 Double-Single

The second supported SP Switch2 switch configuration, called SP Switch2 Double-Single configuration, is arrived at by attaching the second port of each node to a switch placed in the second frame, resulting in the following supported switch configuration. The two switch ports should be on different adapters.

Figure 48 shows a sample Double-Single configuration.

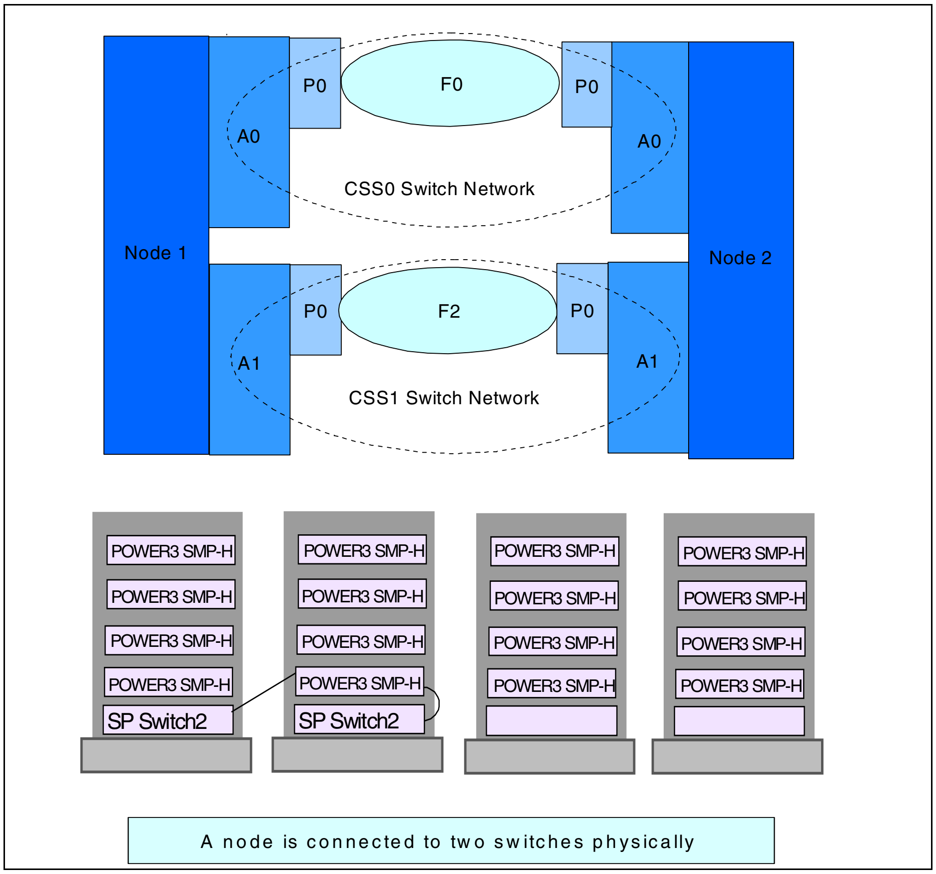

Figure 48. SP Switch2 Double-Single configuration

SP Switch2 Double-Double

In addition to the Single-Single and Double-Single, an SP Switch2 Double-Double configuration is supported, wherein a switch board is placed in the third and fourth frames of the high nodes, and the respective ports of a second adapter in each node are attached to those switches.

Figure 49 shows a sample Double-Double configuration.

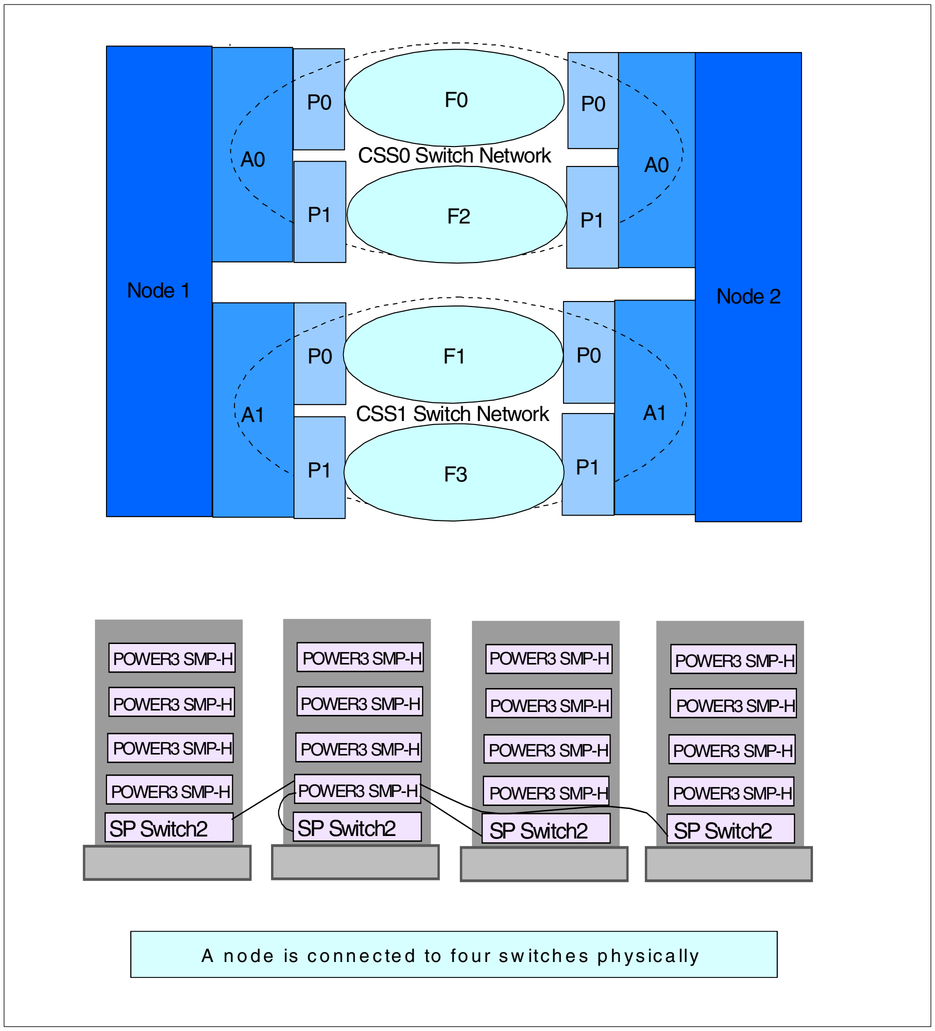

Figure 49. SP Switch2 Double-Double configuration

Note

- SP Switch2 Double-Single and Double-Double configurations are not supported in PSSP 3.2. These will be supported in future PSSP releases.
- We recommend that ports in same node should be connected to same numbered port on each switch board to avoid confusion about port numbers if you use Double-Single or Double-Double configurations.
- Single-Double” configuration will not be supported due to performance and RAS issues.

2.3 SP Switch software

This section describes the new and changed functions in the CSS and System Management components. The SP-attached server and the SP Switch router are also supported in PSSP 3.2.

2.3.1 CSS software enhancement

The CSS component is enhanced to manage the new switch type with up to four parallel switch planes, up to two adapters per node, and up to two ports per adapter. This new support of the new SP Switch2 and adapter coexists with support for the SP switch. In addition, to support for the new hardware, the CSS component is enhanced to improve the overall RASU of both the SP Switch and SP Switch2 fabrics.

The PSSP 3.2 switch code includes all the functions within the PSSP 3.1 release plus a subset of the RASU requirements that can be implemented with the hardware functionality of the SP Switch. The major switch RASU enhancement is to provide recovery from global hardware events, such as clock loss through the switch administration daemon. In PSSP 3.2, the `Eclock` command should only need to run when the switch undergoes changes, such as adding or removing a switch. Events, such as switches powering off or master oscillator failure, will automatically be recovered.

Prior to PSSP 3.2, the master oscillator failure causes a total switch outage that requires operator intervention for recovery. The automatic restart of the switch network on oscillator failures is supported for SP Switch base systems with four or less switches. Before PSSP 3.2, the `Eclock` topology files contained only one alternative clocking permutation. Now, there are more alternatives based on the number of switches available in the SP system.

The function of auto-restart of the fault service daemon on unfence is added. Since PSSP 3.1, you can use auto-unfence during node IPL or other reasons, but some times you may get a negative result from auto-unfence or `Eunfence` commands. This is caused by the fault service daemon not running. Now, PSSP 3.2 supports the restart of the fault service daemon, if required, before attempting to unfence a node.

Before PSSP 3.2, in the following case, you needed to restart the fault service daemon manually. In PSSP 3.2, the `Eunfence` command checks to see if the fault service daemon is running or not on the node, then restarts it automatically if required.

Figure 50 shows the automatic restart of the fault service daemon before attempting to unfence a node.

```
[root@sp6en0:/]# SDRGetObjects switch_responds \
node_number==13 switch_responds autojoin isolated

switch_responds autojoin     isolated
          1           1           0
[root@sp6en0:/]#
[root@sp6en0:/]# Efence 13
All nodes successfully fenced.
[root@sp6en0:/]#
[root@sp6en0:/]# SDRGetObjects switch_responds \
node_number==13 switch_responds autojoin isolated

switch_responds autojoin     isolated
          0           0           1
[root@sp6en0:/]#
[root@sp6en0:/]# dsh -w sp6n13 " ps -e | grep fault_service_Wo "
sp6n13:  15674      -  0:00 fault_service_Worm_RTG_SP
[root@sp6en0:/]# dsh -w sp6n13 " kill 15674 "
[root@sp6en0:/]# dsh -w sp6n13 " ps -e | grep fault_service_Wo "
[root@sp6en0:/]#
[root@sp6en0:/]# Eunfence 13
The fault-service daemon is not running on node sp6n13 -
 trying to bring it up with rc.switch using rsh.
"adapter/mca/tb3"
/etc/inittab entry specified as once for the fault service daemon.
All nodes successfully unfenced.
[root@sp6en0:/]#
[root@sp6en0:/]# SDRGetObjects switch_responds \
node_number==13 switch_responds autojoin isolated

switch_responds autojoin     isolated
          1           1           0
[root@sp6en0:/]# dsh -w sp6n13 " ps -e | grep fault_service_Wo "
sp6n13:  16108      -  0:00 fault_service_Worm_RTG_SP
[root@sp6en0:/]#
```

Figure 50. Auto-restart the fault service daemon example

SP Switch Support Changes

For SP Switch based systems of four or less switch boards, the daemon is enhanced to provide global recovery for hardware events. They include:

- Switches or frames powering down

  When a switch or frame is powered down, all switch operations necessary to change switch clocking and restart switches and nodes will be performed by the Switch administration daemon, including moving the primary and/or backup nodes.

- Clock failure or loss of clock input

  The Switch administration daemon will perform all the steps necessary to recover from a clock outage, which may include moving the location of the Master oscillator or clock redriving switch boards for the SP Switch-based systems. It may also restart nodes on the switch, pick new primary and backup nodes, and run `Estart` in all affected partitions.

- Switches or frames powering up

  The Switch Administration daemon will set up the clocks and `Estart` affected partitions when frames and/or switches power up.

- Invocation of `Eclock` performed by the system administrator

  The Switch administration daemon will perform all the steps necessary to set up switch clocking.

For more information, refer to *IBM Parallel System Support Programs for AIX: Administration Guide*, SA22-7348.

SP Switch2 support

In support of dual port per adapter and two adapters per node, a fundamental change in the fault service daemon occurred. A multi-threaded fault service daemon was needed to manage each port of each adapter as a secondary node switch endpoint. If you use a SP Switch2 system, the cssadm2 daemon runs on the control workstation instead of cssadm. It performs the same node recovery functions for SP Switch2, as does the cssadm daemon for SP Switch systems. The SRC subsystem name for this daemon is *swtadm2*. A new *emasterd* daemon runs in the control workstation, and it subscribes to information provided by the Event Management subsystem in order to monitor the health of the Master Switch Sequecing (MSS) node. The MSS node is the node that periodically re-sequences the time-of-day (TOD) signals on the SP Switch2. These subsystems are inactive on an SP Switch system.

For more information about multi-threaded daemons, refer to Section 2.4.4, “Fault service daemon” on page 95.

You can run the `lssrc` command on the control workstation, as shown in Figure 51, to verify whether these daemons are running or not.

```
[root@sp6en0:/]# lssrc -g  swt
Subsystem         Group            PID     Status
 swtadmd          swt                      inoperative
 swtlog           swt              14456   active
 swtadmd2         swt              4912    active
 emaster          swt              21156   active
[root@sp6en0:/]#
```

Figure 51. SP Switch daemons running on the control workstation

CSS structure changes

This section describes the behavior and enhancements of each CSS component.

- CSS SP Switch2 adapter microcode

  The adapter microcode on SP Switch2 adapters is enhanced to support two ports per adapter. In particular, for service packets, the originating port ID is passed through to switch management code. The behavior of the microcode is modified to allow for a more non-disruptive interaction between the protocols and the fault service daemon. It allows for local adapter error recovery to be handled and the switch routes to be maintained without direct handshaking with the protocols.

- CSS SP Switch2 device driver

  Access, manipulation, and error recovery of the CSS TBX adapters is done directly by the fault service daemon and the traditional device driver. The SP Switch2 adapter will be managed in a more abstracted and centralized manner. IOCTL calls and events are defined to allow access, manipulation, and error recovery, as well as interaction with the protocols.

  The device driver also provides a set of IOCTL calls to allow query and manipulation of the Connectivity Matrix (CM).

  For more information about CM, refer to Section 2.4.8.1, “Switch connectivity” on page 100.

- CSS SP Switch2 kernel extension

  Additional resources are needed by the fault service daemon to manage multiple adapter and multiple ports from the kernel extension. In line with the efforts in the device driver to smooth out the interfaces between events on nodes, adapters, and the protocols, new events are defined within the kernel extension.

Figure 52 shows the CSS structure.

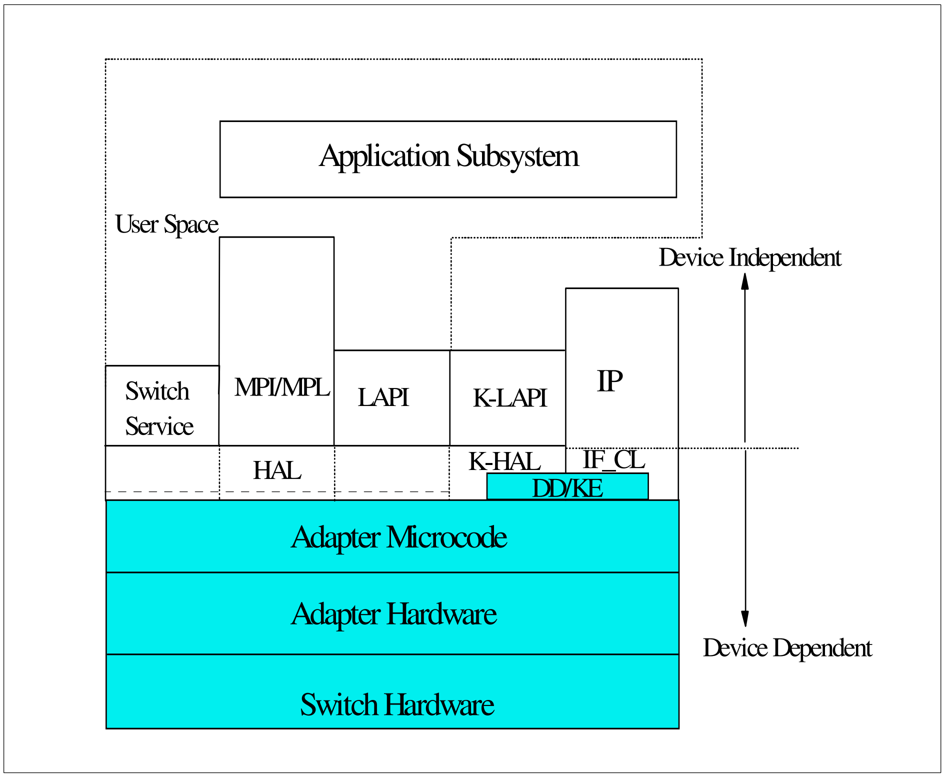

Figure 52. SP Communication SubSystem stack

- Hardware Abstraction Layer (HAL)

  For SP Switch2-based systems, HAL is enhanced to provide support for the switch service packet interface. This interface is used by the fault service daemon to initialize and provide recovery for the switch. In line with the efforts in the device driver and kernel extension to smooth out the interfaces between events on node, adapter, etc., and the protocols, new events are monitored by HAL.

- Switch API library

  For SP Switch2-based systems, the library call to read the TOD clock is enhanced to read the TOD over either of the node's adapters.

2.3.2 System management software enhancement

The PSSP System Management function is extended to support the installation, configuration, management, and monitoring of the SP Switch2 board and adapter hardware. It includes new infrastructure for monitoring the SP Switch2 board and adapters with the Perspectives GUI.

SDR changes

A key feature of the new support for the SP Switch2 is the internal representation of the switch and up to four distinct switch planes. This representation is accomplished by creating new classes and adding attributes to existing classes in the SDR. The new classes include: *Switch_plane* and *Switch_adapter_port.* The classes that are expanded with new/changed attributes include: *SP, Adapter, Switch*, and *switch_responds*.

The assignment of *switch_node_numbers* to node objects continues to be performed by the *SDR_config* script. When an SP Switch2 is installed, SDR_config uses a new algorithm to calculate and assign switch_node_numbers. With an SP Switch2, the “next available” switch_node_number is assigned to a node when the node object is created. The “next available” number is taken from the range of zero to 511 when an SP Switch2 is installed. The new algorithm is used to assign switch_node_numbers to all nodes. On a system with an SP Switch or no switch installed, the “old” algorithm is used to assign switch_node_numbers (actually, pre-assign them in the *Syspar_map* class).

The Syspar_map class is used and appears somewhat differently on SP Switch2 systems. On systems without an SP Switch2, the Syspar_map is used to “pre-allocate” switch node numbers for system partitioning. In this environment, every switch_port (node slot) is represented in the Syspar_map with the “used” attribute signifying which ports (node slots) are actually filled with nodes. On SP Switch2 systems, the Syspar_map contains entries only for nodes attached to the switch (that is, there is no pre-allocation of switch ports). This is because there is no hard connection between node number and switch node number. It is also because system partitioning is not supported on SP Switch2 systems.

For more informations about SDR classes, objects, and attributes, refer to *IBM Parallel System Support Programs for AIX: Administration Guide*, SA22-7348.

New commands

The CMI commands that are used to enter data into the SDR are updated to allow the customer to specify new information related to the SP Switch2 and SP Switch2 adapters, for example, `spadaptrs` and `splstdata`. A new `spswplane` command is added to specify the number of switch planes in use on the SP system. For the SP Switch2, a new `Emaster` command is used to display the MSS node. For SP Switch2, Ecommands are extended with a "-p" parameter when specific plane action is needed. A new `Emaster` command and this flag is valid only on systems with SP Switch2 hardware. If the "-p" parameter is not specified, the default is to perform the operation for all valid switch planes.

In PSSP 3.2, only a single plane (SP Switch2 Single-Single configuration) is supported; so, you can only use the "-p 0" option with Ecommands.

For more information about new commands, refer to *IBM Parallel System Support Programs for AIX: Command and Technical Reference*, SA22-7351.

2.4 SP Switch functional characteristics

This section describes some important functional changes on switch management behavior to support SP Switch2.

2.4.1 J-TAG switch chip initialization

This new interface to the SP Switch2 switch chips provides another way to get information into and out of the switch. What we discuss here is how this interface is used to enhance switch-to-switch mis-wire detection during pre-initialization and during the switch initialization.

At first, you need to know the way mis-wire detection in SP Switch switch initialization works:

1. Switch initialization runs when someone enters the `Estart` command. This causes the port management thread on the switch plane primary node to attempt initialization of the switch plane.
2. This initialization process uses the topology file information specified in the SDR to both label and explore the devices in the switch network.
3. The primary node starts what is called *a breadth first* search of the switch. It contacts the switch chip attached to its switch port, and once it has established communications, it sets the chip ID in the switch chip, based on its expected connection in the topology file. This switch chip ID will be returned to the primary node every time an Switch Error/Status packet is sent from the switch chip.
4. Initialization then does the same for each device attached to the current device. This process continues until all switch chip ports have been explored and verified, or an attached device gives an unexpected response (the wrong switch chip ID is returned).

This approach can lead to switch chips being mis-labeled. Once devices are incorrectly labeled, telling the user that a particular device is mis-wired can be very misleading.

Otherwise, by using the J-TAG interface on the SP Switch2 switch, the problem listed above can be avoided. This interface allows for the partial initialization of switch chips prior to initializing the switch network (`Estart`). This partial initialization consists of setting physical location information (frame and switch chip) in the switch chip. With the availability of this information, via the Switch Error/Status packet, switch initialization code can now positively identify and verify which switch chip it is trying to initialize.

The pre-initialization process will take place every time a switch board is powered on. This includes the following:

- An SP Switch2 switch board is powered on.
- Each switch chip on the board stores its location on the board in the last three bits of the chip identifier field (possible values are zero through seven). These three bits are not writeable through the J-TAG interface.
- Hardmon notices the presence of the *needed* bit and instructs the supervisor card to update the values for the chip identifier field on the switch chips for this board.
- The supervisor card then writes this information to each switch chip on the switch board.
- The SP Switch2 board chip saves this information in the switch chip identifier, minus the last three bits.

The chip identifier field is also updated whenever the configuration is changed. This is done in SDR_config script, which calls the `hmcmds setid` command to instruct hardmon to reset the information.

2.4.2 Switch plane initialization

The initialization of the SP Switch2 plane is very similar to what happens on the SP Switch plane but takes advantage of a number of the new SP Switch2 features. The changes that will be made to switch initialization flow between the two switch types are as follows:

- Node initialization is changed to accommodate the new flexibility supported for switch node numbering. An error/status packet returned from the node now contains the node's switch node number. This information is used by the primary node to build the node portion of the topology of the switch plane.

- Mis-wire detection is improved during switch initialization for the SP Switch2 board. The SP Switch2 board returns two pieces of information in its error status packet that will help this. The first is the “hardware identifier”, which is set every time the switch is powered on. The second is the addition of the “port received on” value, which lets the worm daemon know on which specific port the initialization packet was received.
- Phase 1 of switch initialization also uses the *Return on same path* feature available in the SP Switch2 initialization/read information packets. This packet allows the initialization code to count on the error/status packet response to return out the same port of the switch chip it received the packet on. By using this feature, the initialization time will be decreased in some cases. An un-initialized switch chip will now respond deterministically over the specified route.
- The new switch packet type, read information, will also be used by the secondary nodes to beacon the primary node when they want to join the switch network.

2.4.3 Merge route table and port multiplexing

The route table on an SP Switch2 adapter is different from the route table loaded on an SP Switch adapter. Since an SP Switch2 adapter has two ports through which data can be routed and which are connected to separate switch planes, the route table doubles in size. Routes will be generated in each port management thread for its switch plane. Four routes will be generated per switch port endpoint. The adapter services thread will then merge those two route tables into a single product to be downloaded into the respective adapter. The format/rules to be applied during the merge are:

1. The entries for a single destination node will be grouped together (via switch node number).
2. Eight entries will exist per node destination.
3. The entries will be grouped such that the outgoing port is alternated between entries, for example, route one uses port zero of the adapter, route two uses port one of the adapter, route three uses port zero of the adapter, and so on.
4. Any destination nodes that are only reachable through one port of the adapter will have all eight entries finned with entries for the port that can reach the destination node.

The resultant merged route table, processed sequential round robin by the microcode, will cause the message sent to a destination node to be multiplexed across the switch adapter’s ports and switch planes as well.

2.4.4 Fault service daemon

A multi-threaded fault service daemon is needed to manage each port of each adapter as a secondary node switch endpoint.

In this section, we discuss the behavior of the multi-threaded fault service daemon.

Figure 53 on page 96 shows the conceptual image of the multi-threaded fault service daemon structure and the interactions between the multiple elements needed to support a node with multiple adapters with multiple ports connected to multiple switch planes.

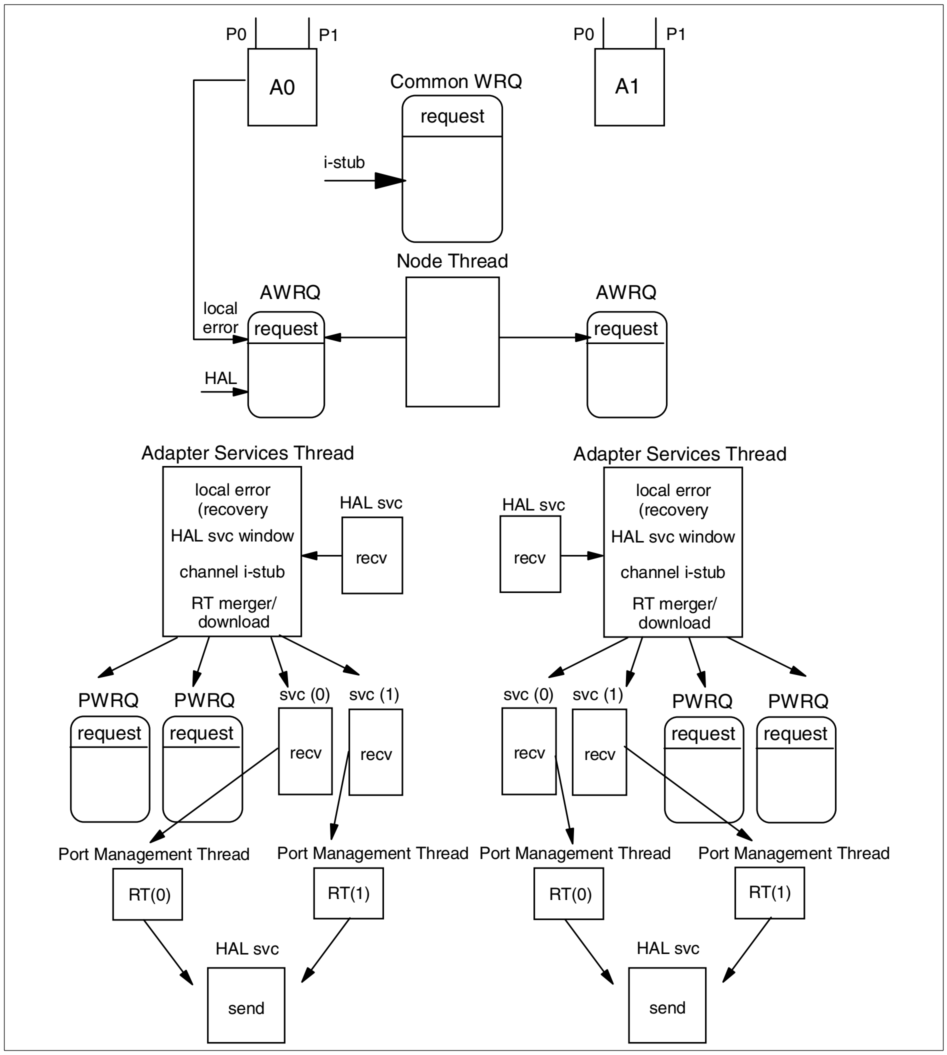

Figure 53. Multi-threaded fault service daemon

There are three types of threads as follows:

- **Node thread** - The node thread's main responsibility is handling work requests (for example, Ecommands) from interface routines and directing these requests to the appropriate adapter service thread's work request queue. The node thread also manages daemon initialization as well as any synchronization or shutdown that may be necessary among adapter services threads.

- **Adapter service threads** - The adapter service threads will be spawned as necessary for each of the installed adapters on the node. The functions performed by these threads are those associated with their respective adapters.

  They include:

  - Receiving service packets from the adapter (via HAL) and directing them to the correct port management thread.
  - Recovering from local adapter errors.
  - Passing work requests from its work request queue to the targeted port management thread's work request queue.
  - Merging the individual route tables of its respective port management threads and directing the resultant route table to be downloaded to its adapter.

- **Port management threads** - The port management threads will be spawned, as necessary, for each of the ports of the installed adapters on the node.The functions they perform are basically the core functions of the SP fault service daemon.

They include:

  - Switch plane initialization
  - Switch fault recovery
  - Route table generation
  - Switch connectivity status update

2.4.5 Event management

New resource variables are added for each of the new switch_responds attributes that record the switch responds value for the switch planes. A new class, *IBM.PSSP.SwitchResponse*, is created, and values are supplied by the IBM.PSSP.Switch resource monitor. The resource variable is called *IBM.PSSP.Switch_responds.state*, and it is indexed by the NodeNum and Switch Plane. So, a query to event management for an instance of the resource variable would be specific to NodeNum and Switch-Plane. These new resource variables are managed differently from the SP Switch variable, IBM.PSSP.Response.Switch.state. The variable in previous levels of PSSP is managed in the main Event Management daemon. The management of these new resource variables is managed by the IBM.PSSP.Switch resource monitor.

2.4.6 Topology Service (TS)

To support the SP Switch2, Topology Service (TS) changed in two points. One is its startup scripts (/usr/sbin/rsct/bin/hats and /usr/sbin/rsct/bin/topsvcs), and the other is the daemon (/usr/sbin/rsct/bin/hatsd).

The `spadaptrs` command is enhanced to add the specification of up to two SP Switch2 switch adapters per node in the SDR. Both of SP Switch2 switch adapters will have a unique IP addresses and netmasks. Based on these SDR objects, the *hats* startup script will build one or two switch heartbeat networks (instead of one being built prior to PSSP 3.2).

The daemon has two basic changes. It will first have to identify whether the connectivity matrix is present. And, it will then have to use the appropriate CM APIs to obtain the self-check bits for adapters that have the connectivity matrix. Querying the *SP_node_ready* bit will still be used for adapters that do not have the connectivity matrix. Other changes in the daemon's code include replacing some hard coded uses of /dev/css0 by appropriate uses of either /dev/css0 or /dev/css1. The adapter membership protocols will not need to change. These protocols assume that if all adapters can communicate with the Group Leader and with the upstream/downstream neighbors, then all adapters can communicate with all others. This assumption should mostly hold true with the port failure escalation mechanism. Without such a mechanism, port failures might have resulted in partial connectivity among adapters, which would lead to protocol instability and false/missing adapter notifications.

For more information about the connectivity matrix, refer to Section 2.4.8.1, "Switch connectivity" on page 100.

2.4.7 Group Service (GS)

To support multiple switch adapters in SP Switch2, the hagsglsm daemon is enhanced. The hagsglsm daemon works by subscribing through GS for the status of the local node's switch adapter and other clients of GS (for example, Event Management, RVSD, GPFS) determine the set of nodes on the switch. Prior to PSSP 3.2, the hagsglsm daemon can list only one switch adapter's status with the "HA_GS_CSS_MEMBERSHIP_GROUP" group. To support multiple adapters, a new group, "HA_GS_CSS_MEMBERSHIP_GROUP_1", is added. This group lists the nodes that have a working css1 adapter.

2.4.8 Logging

The SP Switch system uses some SP specific log files. Some logs reside on the CWS only, and some reside only on the nodes. Others reside on both. If you use an SP Switch, you can find almost all log files on /var/adm/SPlogs/css directory. In an SP Switch2 system, the location of log files are changed to support multiple adapters and multiple ports. Now, we have three level files. They include:

- Node level files
- Adapter level files
- Port level files

Figure 54 shows the image of a directory hierarchy.

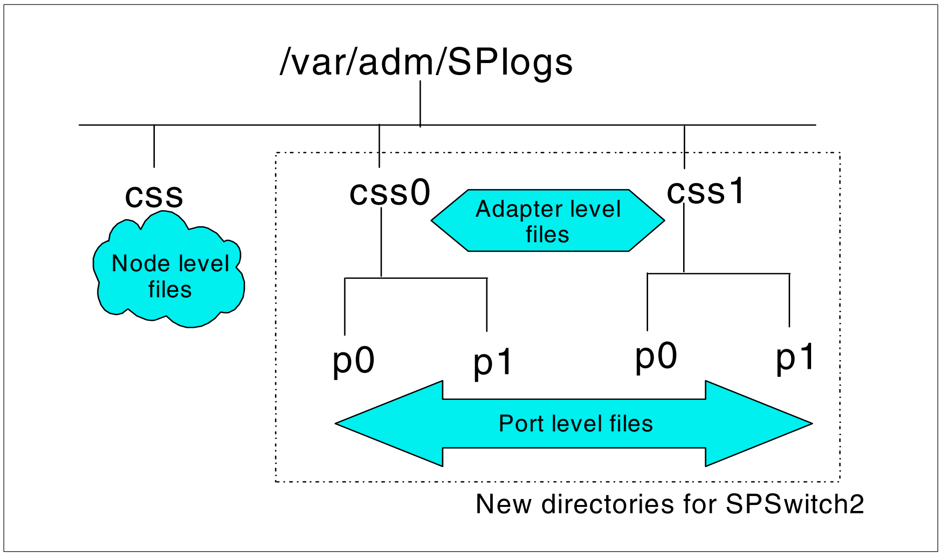

Figure 54. Switch log file directory hierarchy

- Node level files

  The files reside in the /var/adm/SPlogs/css directory, same as the SP Switch log.

  - daemon.log
  - rc.switch.log
  - css.snap.log
  - Ecommands.log

  - logevent.out
  - daemon.log (daemon.stderr and daemon.stdout merged into)
- Adapter level files

  The files reside in the /var/adm/SPlogs/css0 and /var/adm/SPlogs/css1 directory.
  - adapter.log
  - dtbx.trace
  - dtbx_failed.trace
  - css.snap.log
  - scan_out.log
  - scan_save.log
  - router.log (adapter thread level)
- Port level files

  Within the /var/adm/SPlogs/css0 and /var/adm/SPlogs/css1directories two separate directories, p0 and p1, exist, and the log files reside there.
  - flt
  - fs_daemon_print.flt (worm.trace file is absorbed into)
  - out.top
  - router.log
  - topology.data
  - css.snap.log
  - cable_miswire

The css.snap.log occurs at all the level, since css.snap was enhanced to support snaps at node, adapter, and port levels.

For more information about switch log files, refer to *IBM Parallel System Support Programs for AIX: Diagnosis Guide*, GA22-7350.

2.4.8.1 Switch connectivity

To reduce switch management's and other components reliance on the SDR for switch connectivity status and the advent of multiple switch planes drives changes to switch connectivity in PSSP 3.2 for SP Switch2 switch based systems, a new inter-component structure was created called the Switch Connectivity Matrix (CM). This structure is an N * M matrix representing switch connectivity outward from this node on a switch port basis, where N is the maximum number of nodes in the system, and M is the number of switch ports present on a node. The value of N and M are consistent across the system. When adapter level queries are made on systems with two SP Switch2 adapters with both ports being utilized, both ports must be connected for an adapter to be considered connected.

Table 4. Switch Connectivity Matrix

	A0		A1	
	P0	**P1**	**P0**	**P1**
N1	X	X	X	X
N2	X	X	X	X
N3	X	X	X	X
N4	X	X	X	X
-	X	X	X	X
-	X	X	X	X
Nm	X	X	X	X

Each block will represent the four possible switch ports on the node's adapters that this node could have connectivity to the indexed node. A block will exist for each node potentially on the system with the value of "X" if one is reachable though this switch adapter port and zero if it is not. A node's own block will contain the state of each of its switch ports. If a node's port is operational, then all node entries for that port will be zero as well.

The CM resides in a kernel memory. It is updated by the port management, adapter services and the node threads. The device driver will provide an API for other components and subsystems to create, delete, access, and query the CM.

Within the SDR, connectivity will be maintained on a switch plane basis (that is, switch_responds0, switch_responds1, switch_responds2, and switch_responds3).

2.4.9 Data flow control

All applications on an RS/6000 SP system can use the SP Switch network. Using the SP Switch network, you can profit low latency and high bandwidth communication methods. Two communication protocols are offered on the adapter:

- Standard TCP/IP communication through AIX sockets or message passing libraries
- Dedicated user space access via message passing libraries

This section describes the data flow on the SP Switch fabric.

At first, you need to know about the windows that send and receive data to and from the switch. There are three different types of windows:

IP window	Responsible for the IP communication among nodes
Service window	Manages configuration and monitoring of the switch network
User space window	Permits high-speed data communication among user applications

Each window has its own send and receive FIFOs and a set of variables that describe the status of its FIFOs, such as the position of first available to both windows and adapter microcode and that are used to properly transfer data to and from the window's FIFOs and the adapter.

In the SP Switch2, there are up to 16 user space windows available for each node. Using the SP Switch, you can use up to four user space windows even if the system is in PSSP 3.2.

2.4.10 User space data flow

The send FIFO consists of a Send Command FIFO and a Send Data FIFO. The following are the basic structures of a send FIFO:

- Send command FIFO in adapter SRAM
- Send data FIFO in pinned system memory
- Short messages sent as immediate data in command FIFO
- Send head pointer in pinned system memory
- Send tail pointer in adapter SRAM
- Protocol/window independent microcode field

The following are the basic structures of a receive FIFO:

- Receive FIFO in pinned system memory
- Receive head pointer in adapter SRAM
- Receive tail pointer in pinned system memory
- Protocol/window independent microcode operations

Figure 55 shows software node structure.

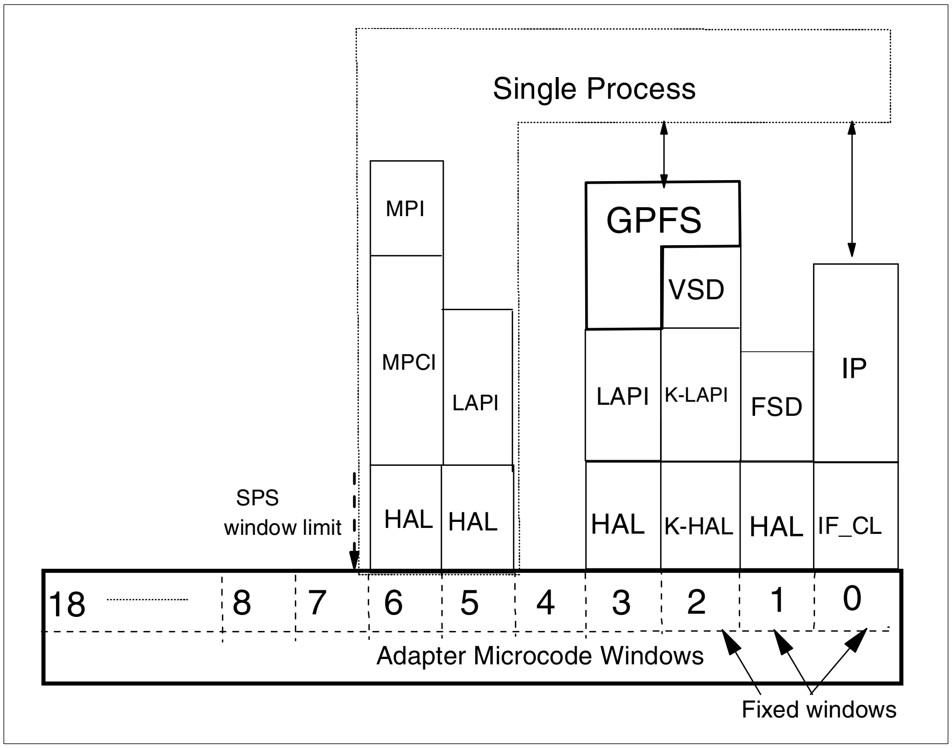

Figure 55. Window assignment

2.4.11 IP data flow

The IP send interface consists of *M-bufs*, *interface Cluster Buffers*, and a send command FIFO. The following are the basic structures of a send FIFO:

- Send command FIFO in adapter SRAM
- Command "select" operation field
- Small datagrams (less than 22 Bytes) are in M-bufs at the IP layer
- Small datagrams are sent as immediate data in command FIFO at the IF_CL layer
- Medium datagrams in M-bufs chain at the IP layer
- Medium datagram in M-bufs chain are assembled into a cluster buffer
- Send interface cluster buffer in pinned system memory
- Protocol/window independent microcode operations

The followings are the basic structures of a receive FIFO:

- Receive command FIFO in pinned system memory
- Receive head pointer in adapter SRAM
- Receive tail pointer in pinned system memory
- Receive local cluster buffers RD-RAM
- Receive system cluster buffers in pinned system memory
- Protocol/window independent microcode operations

2.4.12 Device memory allocation

The total device memory, reserved for SP Switch2 adapters' User Space windows used as interface network FIFO buffers, is specified as an element (win_poolsize) in the DDS structure for the SP Switch2 Device Driver along with other node-specific configuration parameters. This information is contained in the ODM data base. In previous releases of PSSP, only spoolsize and rpoolsize can be changed with the `chgcss` command.

This command is enhanced to include the driver configuration parameters listed in Table 5.

Table 5. Device Driver's DDS structure configuration data (SP Switch2)

Parameter	Description	Default
win_poolsize	total window memory pool size	128 MB
win_maxsize	maximum window memory size	16 MB
win_minsize	minimum reserved window memory	1 MB

The *maximun_logical_windows* is five for SP Switch systems and 17 for SP Switch2 systems. The default values listed bellow are specified as part of the *ODM*'s default settings:

- win_poolsize
  - Minimum value = maximum_logical_windows * win_minsize
  - Maximum value = 256 MB (80 MB for SP Switch)
  - Default value = 128 MB (80 MB for SP Switch)
- win_maxsize
  - Minimum value = 256 KB
  - Maximum value = win_poolsize (for SP Switch, no greater than 16 MB)
  - Default value = 16 MB

- win_minsize
  - Minimum value = 256 KB
  - Maximum value = win_poolsize / maximum_logical_windows
  - Default value = 1 MB

For more information on the `chgcss` command usage, refer to *IBM Parallel System Support Programs for AIX: Command and Technical Reference*, SA22-7351.

You can run the `lsattr` command on the SP Switch2 system to verify the current adapter values. Figure 56 shows the current adapter values in our SP system.

```
[c190n01:/]# lsattr -E -l css0
adapter_memory 0xe0000000       adapter memory address         False
rambus_memory  0x10c0000000     RAMBUS  memory address         False
bus_mem_addr   0x04000000       Bus memory address             False
win_poolsize   134217728        Total window memory pool size  True
win_maxsize    16777216         Maximum window memory size     True
win_minsize    1048576          Minimum window memory size     True
int_priority   3                Interrupt priority             False
int_level      32               Bus interrupt level            False
spoolsize      2097152          Size of IP send buffer         True
rpoolsize      2097152          Size of IP receive buffer      True
khal_spoolsize 524288           Size of KHAL send buffer       True
khal_rpoolsize 524288           Size of KHAL receive buffer    True
adapter_status css_ready        Configuration status           False
diags_prog Diagnostic program           True
ucode_version  1                Micro code version             True
ucode_name     /etc/microcode/col_ucode Micro code name        True
window         AVAIL AVAIL AVAIL AVAIL AVAIL AVAIL AVAIL AVAIL AVAIL
                       AVAIL AVAIL AVAIL AVAIL AVAIL AVAIL AVAIL AVAIL
```

Figure 56. Sample command to verify the current adapter values

If you use SP Switch2, the default value of the spoolsize and rpoolsize attributes is 2097152 (2 MB). This is different from SP Switch adapters default values (524288). These values are set by the /usr/lpp/ssp/css/colony.add file during installation of the ssp.css fileset.

We show a sample change window setting with SP Switch systems in Figure 57. In this sample, we reduce the win_poolsize from 80 MB to 64 MB and then check the ODM win_poolsize entry.

```
[c187n09:/]# lsattr -E -l css0
bus_mem_addr   0x04000000        Bus memory address              False
int_level      0xb               Bus interrupt level             False
int_priority   3                 Interrupt priority              False
dma_lvl        8                 DMA arbitration level           False
spoolsize      524288            Size of IP send buffer          True
rpoolsize      524288            Size of IP receive buffer       True
adapter_status css_ready         Configuration status            False
win_poolsize   83886080          Total user device memory        True
win_maxsize    16777216          Maximum per-user device memory True
win_minsize    1048576           Minimum per-user device memory True
window         VSD AVAIL AVAIL AVAIL AVAIL Adapter window ownersTrue

[c187n09:/]#chgcss -l css0 -a win_poolsize=67108864
chgcss: attribute win_poolsize value = 67108864.
chgcss: attribute win_maxsize value = 16777216.
chgcss: attribute win_minsize value = 1048576.

[c187n09:/]# lsattr -E -l css0 | grep win
win_poolsize   67108864          Total user device memory        True
win_maxsize    16777216          Maximum per-user device memory True
win_minsize    1048576           Minimum per-user device memory True
window         VSD AVAIL AVAIL AVAIL AVAIL Adapter window ownersTrue

[c187n09:/]# odmget -q 'name=css0 and attribute=win_poolsize' CuAt
CuAt:
        name = "css0"
        attribute = "win_poolsize"
        value = "67108864"
        type = "R"
        generic = "DU"
        rep = "n"
        nls_index = 15
```

Figure 57. Sample change window setting with SP Switch systems

You do not need to change them unless you need to, which should be based on your RS/6000 SP system's switch communication and workload demands. If you decide you need to consider making changes, refer to SP tuning information at the following URL: `http://www.rs6000.ibm.com/support/sp/`

16Way MUSPPA

The SP Switch2 supports a maximum of 16 concurrent User Space processes to use the POWER3 SMP high node, which is a 16-way SMP, effectively. The SP Switch2 adapter has 2 MB of SRAM, which supports having 16 real hardware windows. The CSS software design has the further flexibility of supporting “variable” Multiple User Space Processes per node [Adapter] (MUSPPA). If you use SP Switch, you can use up to four MUSPPA.

This section describes the SP Switch2 in PSSP 3.2 MUSPPA design.

- For SP Switch2 in PSSP 3.2, the number of adapters has no relationship to the number of User Space processes supported. Both one adapter and two adapter SP Switch2 configurations support up to a maximum of 16 User Space processes.
- To support the “variable” MUSPPA is an important aspect. The number of User Space windows can be in the range of zero to N. The maximum of N is four for SP Switch adapter nodes and 16 for SP Switch2 adapter nodes. The maximum number of windows available (to LoadLeveler) for User Space jobs depends on the use of windows by other subsystems (for example, GPFS).
- Variable MUSPPA has a maximum of N User Space processes per node (aggregate of all jobs). The maximum is obtained if each process uses only one protocol (MPI or LAPI). If a process uses both MPI and LAPI, the potential maximum is only N/2 processes per node.

Example

- An SP Switch2 node has windows available for LoadLeveler’s use.
- An application using MPI and LAPI protocols is run on the node.

The maximum number of parallel tasks (or processes) that can be run on this node is eight.

2.4.13 Adapter window assignment

The IP window has a fixed window assignment of window zero, and Service has a fixed window assignment of window one. Refer to Figure 55 on page 103 for window assignment information. For compatibility reasons, these assignments match the fixed assignments on the SP Switch systems. The total number of device windows supported by the adapter is obtained by the Device Driver directly from the adapter after the device is opened. The SP Switch2 supports 19 windows: Two fixed, one pre-reserved for K-LAPI, and up to 16 available for MUSPPA use. If you use an SP Switch, you can use up to four MUSPPA because the SP Switch supports only seven windows.

In PSSP 3.2, new attributes were added to the `chgcss` command to manage window numbers. Through this command, you can reserve or release the windows for your applications. In PSSP 3.2, a reservation on behalf of VSD using KLAPI will be made on each node, therefore, obtaining logical window zero for kernel use. This can be undone by the `chgcss` command with the RELEASE option as shown in the following sample in Figure 58.

```
[root@sp6n01:/]# chgcss -l css0 -a window=cmd:query
VSD AVAIL AVAIL AVAIL AVAIL
[root@sp6n01:/]# chgcss -l css0 -a window=cmd:release/id:VSD
AVAIL AVAIL AVAIL AVAIL AVAIL
[root@sp6n01:/]# chgcss -l css0 -a window=cmd:query
AVAIL AVAIL AVAIL AVAIL AVAIL
```

Figure 58. Sample chgcss command

If the VSD/KLAPI window is explicitly released, it can be used by any other client, kernel, or user space. However, on SP Switch systems, only one window can be reserved as a kernel window type due to microcode limitations.

Note

On SP Switch2, 16-way MUSPPA and GPFS/LAPI are mutually exclusive if VSD/KLAPI is configured.

For more information about VSD/KLAPI, refer to Chapter 3, “IBM Virtual Shared Disk” on page 119, and, for GPFS/LAPI, refer to Chapter 4, “GPFS Release 3” on page 143.

2.5 SP Switch RAS concept

For the SP Switch system, a switch adapter or port failure will result in the node’s loss of connectivity to the switch. In case of a switch adapter or port failure, users can use another network instead of switch the network. But, it needs some considerations about performance and application design. In the future, if you have a four switch-plane SP Switch2 system, the communication subsystem provides redundant node-to-node connectivity. Each node contains two separate and independent switch adapters. Each adapter contains two switch ports. Each switch port connects to a separate and independent switch-plane. Each switch-plane provides a multiple switch chip connecting any two nodes. We discuss the SP Switch2 RAS concept and availability in this section.

2.5.1 Aggregate IP address concept

This section describes the concepts needed to ensure that a single failure in the SP Switch2 subsystem does not cause failure of a node, multiple nodes, or failure of the PSSP software subsystems (for example, GPFS or VSD). Both SP Switch2 Double-Single (Figure 48 on page 83) and Double-Double (Figure 49 on page 84) provide two logical switch networks derived from the fact that there are two (switch) network adapters.

The addition of a third IP address that aggregates the addressing of the two SP Switch2 adapters on a single node is provided. This aggregate IP address is optional and must be explicitly assigned by the administrator. To support the aggregate IP address, new SDR classes, CMI commands, and new SMIT panels will be supported.

Multi-link device driver

First of all, we need to discuss the pseudo device driver. The CSS provides two independent instances of the IP network interface driver (IF_CL), one for each SP Switch2 adapter. Each of the switch networks is associated with a unique IP address; that is, each node has two IP switch addresses. There is no physical or logical connection between these two networks. In addition to these two IP drivers (and addresses), a third pseudo IP driver will be provided to give a multi-link/aggregate of the two real IP devices. IP messages sent using this aggregate address will be striped across both adapters at the datagram's boundaries.

The working name for the IP aggregate/multi-link pseudo device is "ml0". If a node is customized to have the ml0 device by the administrator, the rc.switch routine will query the ODM and determine the configured IP devices during node initialization. For two adapter SP Switch2 systems that have the multi-link device configured, the `ifconfig` command will be invoked three times to load and initialize IP devices css0, css1, and ml0. The ml0 pseudo device initialization will query the ODM to determine which real IP devices (css0, css1) to use from the aggregate device and their associated IP addresses. New ODM entries will have the following form:

- associated device name: (name = ml0)
- IP address: (attribute = agnetaddr)
- Netmask: (attribute = agnetmask)
- update_interval: (attribute = aginterval)
- update_threshold: (attribute = agthreshold)
- Underlying device information: (attribute = agglist)

Multi-Link Route Table

The multi-link IP driver contains a *Multi-Link Route Table (MLRT)*. This table contains entries for each node in the system, and each entry contains the parameters associated with each destination node in the MLRT entry as shown in Figure 59.

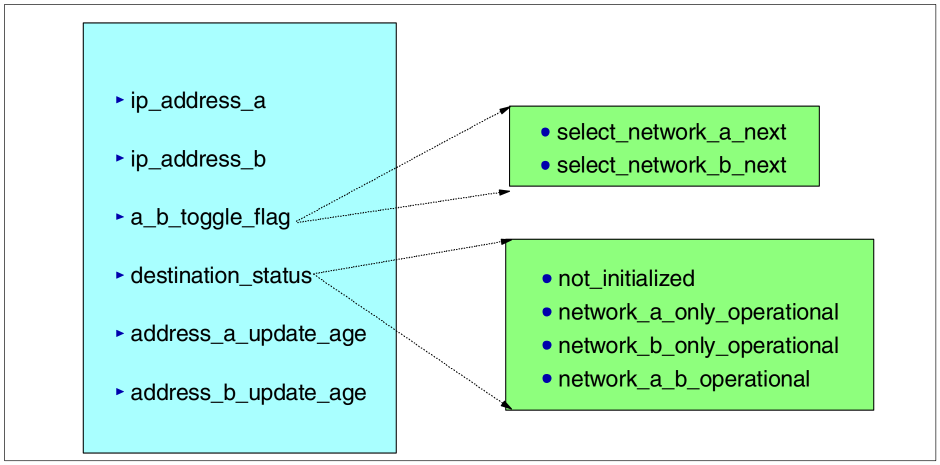

Figure 59. Multi-Link route table

“ip_address_a” and “ip_address_b” are the IP addresses of the switch adapter device (css0, css1) on the destination node. The “a_b_toggle_flag” is used to select the destination IP address when both network_a and network_b are operational as indicated by the “destination_status”.

When only one of two networks is operational between the source and destination node, the “destination_status” is used to determine the destination IP address. All datagrams, including the IP message, will be sent using the single operational network. If neither network is operational between the source and destination node, the multi-link device driver will drop the outbound IP datagram.

The contents of the MLRT are filled in using the multi-link *hello protocol.* At periodic intervals (“update_interval”), the multi-link driver sends multicast/broadcast messages out its local switch networks. Between hello protocol updates, the existing MLRT is used for outbound messages, and updates to the MLRT happen asynchronously. When a hello protocol update arrives, it is used to update the MLRT, for example, if the new update advertises that one of the sender’s interfaces is no longer available, the MLRT is updated to reflect the change. Whenever an update arrives, the “update_age” field for the sending interface in the MLRT entry is set to “update_threshold”. In the fullness of time, the update_interval timer triggers the multi-link driver to send an update. The multi-link driver will look at the states of its local IP interfaces and broadcast/multicast their states in a hello protocol update. It will walk through the MLRT, decreasing the update_age fields for each of the interfaces. If any of the update_age fields reaches zero, that link is presumed dead, and the MLRT “destination_status” field is updated to remove the link.

Multi-link message flow

In this section, we show the multi-link IP message flow, depicted in Figure 60, as a simple, two-node example.

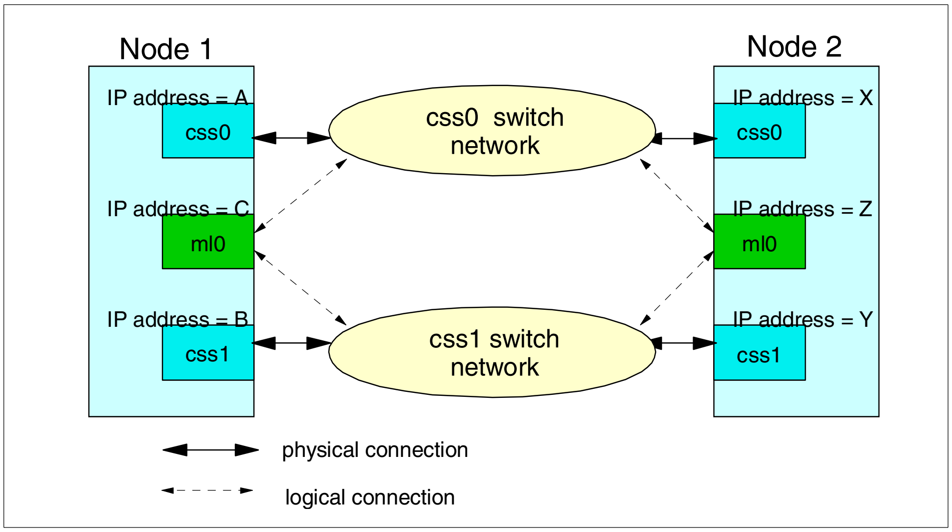

Figure 60. Simple, two node example

An IP message is sent from Node 1 to Node 2 using a destination IP address of Z. The normal AIX IP send protocol proceeded to the point where the function *ip_output* is called. ip_output is processed at the IP routing layer. Using the IP route table for the subnet associated with address Z, the IP routing layer passes control to the multi-link interface driver for device ml0. The ml0 driver uses address Z to index into its MLRT. Using the “destination_status”, it determines that both “network_a_b_operational”. Then, it finds the “a_b_toggle_flag” set to “select_network_b_next”, and the IP address “ip_address_b” has the value of Y.

At this point, the ml0 driver calls the function ip_output with Y as the destination address. In this second invocation of ip_output, the route table associated with Y’s subnet is used. This results in control being passed to the css1 device driver. At Node 2, the message (containing destination IP address Z) will be processed by the css1 device driver and is simply passed up to the IP layer. IP protocol processing will deliver the message to the IP socket associated with address Z.

Sample failover behavior

In this section, we show an example of a striping/fail-over behavior.

Figure 61 shows a two switch network SP system having multiple forms of node connectivity.

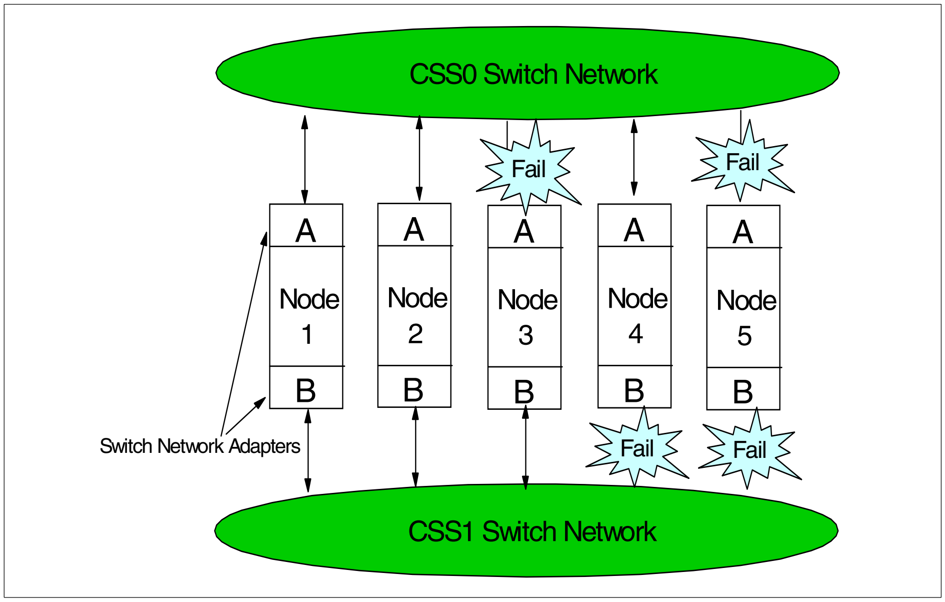

Figure 61. Multiple switch network striping and failover

- When an IP message is sent from Node -1 to Node -2, and both nodes have two fully operational adapters (A, B), consecutive datagrams will be sent in the pattern: Adapter-A, adapter-B, adapter-A, adapter-B, and so on.
- When an IP message is sent from Node -2 to Node -3, and one of the two nodes has one non-working adapter, all IP datagrams between Node -2 and Node -3 will be sent using a single adapter (that is, there will be no striping). For example, assume adapter-A on Node -3 is broken, all datagrams are sent using adapter-B. So, the pattern, in this case, is adapter-B, adapter-B, adapter-B, and so on. Users of this third address will see no loss of IP connectivity/availability between Node -2 and Node

-3. There will be a reduction in bandwidth capacity between these two nodes. The bandwidth capacity between Node-2 and Node-1 will be unaffected by the adapter failure on Node -3; messages from Node -2 to Node -1 will continue to be striped using both adapter-A and adapter-B.

- If Node -3 has a working adapter-B and a non-working adapter-A, and Node -4 has a working adapter-A and a non-working adapter-B, IP messages cannot be sent from Node -3 to Node -4. The simple reason for this is that there is no physical switch connection between Node -3 and Node -4. The third IP address continues to be available on both of these nodes, but Node -3 and Node -4 cannot exchange messages. Both Node -3 and Node -4 can exchange IP messages with either Node -1 or Node -2.
- If Node -5 has two non-working adapters, the third address will not be available (neither will either of the two IP addresses of the two independent switch networks).

2.5.2 Switch route failures

Under normal conditions, packets are spread across four route paths per destination node for each adapter switch port. A switch route is comprised of one to five switch chip connections depending on the physical location and switch port connection of the source/destination node pair. For the majority of node pairs in larger system configurations, there exist greater than four possible route paths. When a switch connection fails, a switch fault notification is given to the Switch Fault Service daemon. The Service function contacts the appropriate adapter and issues a command for the adapter to suspend the send processing of all client windows (except the Service window). Next, the Service function determines the current state of the switch and re-compiles the list of available route paths for each destination node for each switch port. Four available routes are selected for each route port pair. If four unique routes are not available, the available routes are replicated to fill in the set of four routes. The updated route table is loaded onto the adapter. The Service event ends by issuing a second command to the adapter microcode to resume normal client window processing. The upper layer protocols do not have a direct awareness of this class of switch fault event.

2.5.2.1 Adapter port failures

In SP Switch2 systems, if you have more than two switch adapter ports per node, a switch adapter failure doesn’t result in the total node’s loss of connectivity to the switch. This section describes the behavior when the adapter port failures occurred.

- Single switch-plane

For single switch-plane SP Switch2 systems, an adapter switch port failure will result in the node's loss of connectivity to the switch. The fault service function will give notification to the Device Driver to close all adapter windows. For User Space windows, the DD does a post of the HAL notification thread. If the protocol registered an error handler, HAL will pass control to this error handler. Otherwise, HAL will take no action. The protocol error handler needs to call specific function, which closes the adapter window. If this window is not closed within a specified time interval, the Device Driver will send a terminating signal to the process, which owns the window.

- Dual switch-planes

  For dual switch-plane SP Switch2 systems, a port failure will result in the loss by the node of connectivity to one of two switch-planes. The Switch Fault Service will give notification to the Device Driver to notify window users that the Connectivity Matrix has been updated. The Device Driver notifies HAL of the Connectivity Matrix event by posting HAL notification thread. HAL will query the Connectivity Matrix and determine which adapter number had the failure. HAL will then exclusively use the (single) adapter number on all of the nodes. Both MPCI and LAPI's reliability mechanisms will recover any lost packets. Maximum bandwidth capacity will be reduced by half. If both adapters on a node fail, the DD will notify all window clients to close the window. This will result in the node's loss of connectivity to the switch.

- Quad switch-planes

  The quad switch-plane system node behavior is that of two separate dual switch-plane node connections. The adapter single port failure will be escalated into an adapter failure. The Dual Switch-plane operation as described above is followed.

2.5.3 Adapter Failures

In SP Switch2 systems, if you have two adapters per node, a switch adapter failure doesn’t result in the total node’s loss of connectivity to the switch. This section describes the behavior when the switch adapter failures occurred.

- Single Switch-plane

  For single switch-plane SP Switch2 systems, an adapter failure will result in the node's loss of connectivity to the switch. The behavior is same as a switch port failure.

- Dual and quad switch-planes

For dual and quad switch-plane SP Switch2 systems, a single adapter failure simply means that nodes can only be reached via the one operational adapter. The fault service function will give notification to the DD that the Connectivity Matrix has been modified. In response to this events the DD will notify each window client that the Connectivity Matrix has been modified. The SP Switch2 IP design provides an optional third IP device/address. This third IP address provides transparent fail-over from single adapter failures. The concept of the third IP address is also called an aggregate IP address.

For more information about aggregate IP address, see Section 2.5.1, “Aggregate IP address concept” on page 109.

2.5.4 Switch failures

A total single switch plane failure is analogous to a port failure occurring on all nodes throughout the entire system.

- Single switch-plane

  For single switch-plane systems, a switch-plane failure results in the lose of all communications.

- Dual switch-planes

  For two switch-planes system, a single switch-plane failure results in the route table on every adapter having routes specified using a single adapter port. The protocols do not have a direct awareness of the failure. The system's total bandwidth capacity will be reduced by half. Any lost packets will be recovered by upper layers of the protocols that employ packet reliability mechanisms.

- Quad switch-planes

  For four switch-planes systems, a single switch-plane failure results in the route table for the associated adapter on every node having routes specified using a single adapter port. The protocols do not have a direct awareness of the failure. The system's total bandwidth capacity will be reduced by a forth. Any lost packets will be recovered by upper layers of the protocols that employ packet reliability mechanisms.

2.6 SP Switch concurrent repair

The SP Switch hardware already has high reliability, but the concurrent repair is not supported. Now, SP Switch2 supports the concurrent repair.

Over 70 percent of the SP Switch2 system's components can be repaired while the switch is operating. This function will reduce the unscheduled system maintenance time and improve system availability.

Each occupied switch port in the SP Switch2 contains an interposer card (F/C 4032). Interposer cards can be changed or added while the switch is operating. Any unused switch ports must have blank interposer cards (F/C 9883) installed, which prevent contamination of the connector and ensure proper cooling air flow.

2.6.1 Hardware design

SP Switch2 components are designed for 100 percent of concurrent repair. Table 6 shows the concurrent repair capabilities of the SP Switch2.

Table 6. Concurrent repair

Failure component	Concurrent repair
Switch Power	(N+1) Hot swappable
Switch Supervisor Cards	Hot swappable
Intermediate Switch Boards	(N+1) Hot swappable
Switch Boards	Only directly attached node effected
Interposer Cards	Hot swappable
Cables	Hot swappable
Switch Adapters	Single Node Outage Only
Supervisor & Switch Adapter	On-Line Code Patch

2.6.2 SP Switch versus SP Switch2

The SP Switch2 system has a large advantage of the hardware maintenance. SP Switch subsystem hardware already has high reliability. Moreover, The SP Switch2 adapter‘s unscheduled outages is reduced by 10 percent, and the SP Switch2's unscheduled outages is reduced by 30 percent.

Figure 62 shows the comparison of the SP Switch versus SP Switch2 hardware maintenance ratio.

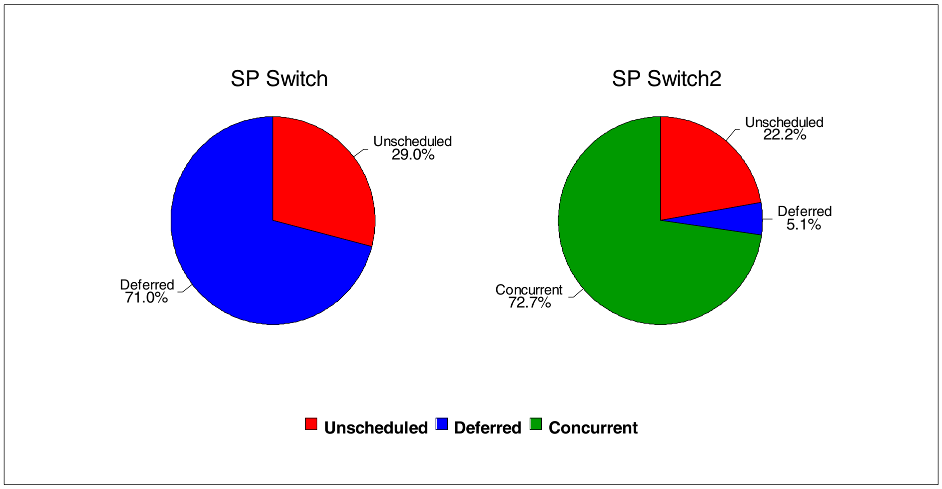

Figure 62. Comparison of SP Switch versus SP Switch2 (repair ratio)

In SP Switch2 systems, over 70 percent of the failures do not cause a service outage. Table 7 shows that all repairs are now concurrent with the system operation.

Table 7. Comparison of SP Switch versus SP Switch2 (system Impact)

Failure Module	**SP Switch Impact**	**SP Switch2 Impact**
Supervisor card	Machine may power off Can't get switch state Deferred repair	Machine keep running Can't get switch state Fault tolerated and concurrent repair
An interposer card	Connected node can't communicate with the switch Unscheduled node outage Deferred repair	Connected node can't communicate with the switch Unscheduled node outage Concurrent repair
An oscillator	Master oscillator - Sys failure Redrive - Partial Sys failure Unscheduled system outage Immediate repair	Same as loss of a switch chip or same as loss of four ports or same as loss of four interposer cards Unscheduled node outage Concurrent repair
A switch chip	Loss of four ports Assumed to be unscheduled incident	Loss of four ports Assumed to be unscheduled incident

Failure Module	SP Switch Impact	SP Switch2 Impact
A fan	Fault tolerated	Fault Tolerated Concurrent repair
Power card	Fault tolerated Deferred repair	Fault tolerated Concurrent repair

Chapter 3. IBM Virtual Shared Disk

This chapter begins by introducing IBM Virtual Shared Disk (VSD) and IBM Recoverable Virtual Shared Disk (RVSD) concepts for the purpose of background reading. For the more experienced reader, we introduce Kernel Low-Level Application Program Interface (KLAPI) and Concurrent Virtual Shared Disk (CVSD), which are new in IBM VSD 3.2.

For more detailed information, refer to *IBM Parallel System Support Programs for AIX: Managing Shared Disks*, SA22-7349.

IBM Virtual Shared Disk

IBM Virtual Shared Disk is the software that enables nodes in the RS/6000 SP to share disks with the other nodes in the same system partition.

A *Virtual Shared Disk* is a logical volume that can be accessed not only from the node it belongs to, but also from any other node running the VSD software in the system partition.

A *VSD Server* is a node that owns a number of VSDs. It reads and/or writes data to VSDs as requested by client nodes and transfers data back usually via the SP switch.

A *VSD client* node is a node that requests access to VSDs. A node can be both a VSD server and a client node simultaneously.

IBM Recoverable Virtual Shared Disk

If a VSD server node fails, access to data on all VSDs that it owns is lost. In order to avoid this scenario, we implement RVSD and twin-tailed or loop cabling between nodes.

The RVSD concept is to allow not only one node (the VSD server primary node) to have access to a set of VSDs, but also a second node (the VSD server secondary node) in case one of the following fails:

- VSD server primary node
- Switch adapter
- Disk adapter
- Disk or network cable

RVSD provides protection against node failure by subscribing to Group Services. When a node fails, RVSD is informed by Group Services.

If the failed node is the VSD server primary node, RVSD will have the VSD server secondary node take over the disk subsystems from the primary node and become the server for those VSDs while the primary node is unavailable.

Twin-tailed or loop cabling of the disk subsystem between nodes is needed in order to provide an alternate path to the disk subsystem from the VSD server secondary node.

With RVSD, the disk subsystem becomes highly available since you can have continuous access to the VSDs, even when the VSD server primary node is unavailable.

The amount of time required to failover to a secondary server is dependent on the number of volume groups that must be varied online, the number of virtual shared disks that make up the volume group, and whether the volume groups need to be re-imported due to configuration changes that have occurred on the primary server.

Note

RVSD uses the notion of quorum, the majority of the VSD nodes, to cope with communication failures. If quorum is not met, the VSD will move to **STOPPED** state, and the server list will be blank.

3.1 Kernel low-level Application Programming Interface

The KLAPI protocol has been introduced to meet the growing performance requirements of IBM General Parallel File System for AIX (GPFS) and VSD.

The overhead of GPFS has been dominated by multiple data copies. Studies have concluded that elimination of a data copy in GPFS is required to reduce the overhead of GPFS and improve GPFS node throughput. With many large RS/6000 SP systems running GPFS with large amounts of data, these performance enhancements are a necessity.

With data being written, and read becoming increasingly larger, the VSD transport services have been improved through KLAPI. KLAPI is an efficient transport service for communication between clients and servers and has been developed to aid VSD. KLAPI is a zero copy transport protocol that supports fragmentation and reassembly of large messages, packet flow control, and recovery from lost packets. Message sizes are as large as GPFS block sizes. Packet flow control is introduced to prevent switch congestion when many nodes send to one, for example, in the VSD environment. This helps reduce the VSD retries, which can significantly impact performance.

KLAPI provides transport services to kernel subsystems that need to communicate via the SP Switch. KLAPI semantics for active messages are similar to those for user space LAPI.

Refer to Section 6.1, “LAPI” on page 181, for more detailed information regarding LAPI.

3.1.1 Layering

Just as in user space, Kernel Hardware Abstraction Layer (KHAL) provides an abstraction of the SP Switch adapter for KLAPI. The semantics of many KHAL functions are similar to user space HAL functions. However, KHAL must also have cross memory descriptors specified in addition to memory pointers when copying data to or from internal FIFOs. Interrupt handling and the open and close functions require the most significant changes. KHAL adds support for super packets (for example, large contiguous packets that require adapter fragmentation and reassembly functions) and zero copy through the use of direct DMA.

Most of the HAL open and close functions are implemented in the CSS kernel extension for both user space and kernel. Like IP, the adapter windows for KHAL use a partition table where a logical task is equal to a switch node number.

In user space, HAL uses a background thread for interrupt handling, while a separate kernel process is used for this purpose in KHAL. This kernel process is awakened when a timer expires, when new packets arrive, and when interrupts are enabled or there is a fault condition that KHAL must handle.

KLAPI reuses much of LAPI and shares the same code base. KLAPI uses kernel locking services in place of the pthread functions it uses in user space. KLAPI must also use cross memory services since KLAPI clients do not share the same address space as the kernel interrupt process. Additionally, KLAPI makes use of send completion handler rather than counters as the primary mechanism for determining message completion status.

Figure 63 shows how these components are layered within an application call.

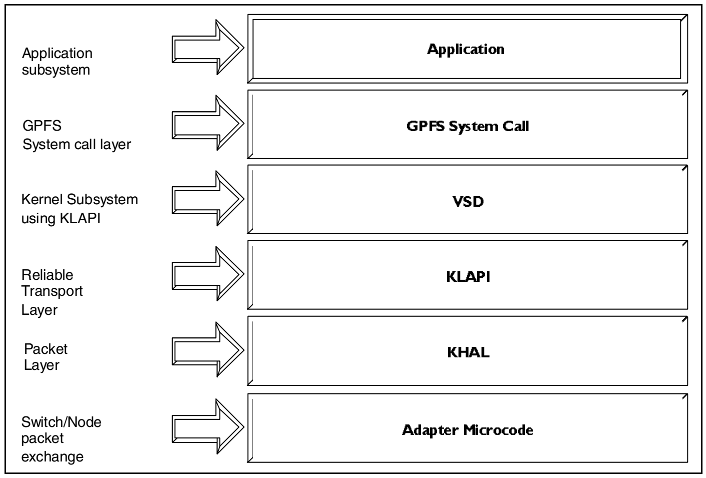

Figure 63. Software stack for an application using GPFS with KLAPI turned on

3.1.2 Functionality with VSD/GPFS

In releases prior to PSSP 3.2, IBM's VSD used IP for transport services. It also shares buffers with the SP Switch interface. To exploit KLAPI in PSSP 3.2, IBM VSD utilizes KLAPI transport services and provides its own buffer management on the server side. For GPFS, IBM VSD manages direct DMA to the GPFS buffer cache using Kernel LAPI services.

To utilize KLAPI transport services, IBM VSD replaces calls to its own send and receive functions with calls to the KLAPI active messaging interface. This includes using LAPI_Kamsend and providing header handlers and completion handlers for receiving data.

KLAPI provides a zero copy transport with message fragmentation and reassembly, packet flow control, and recovery from packets lost in the network as part of its transport service. IBM VSD must provide message flow control to balance the use of server resources and manage recovery from down networks and down nodes. KLAPI provides services to aid in the recovery from down networks and nodes.

IBM VSD provides its own buffer management on the server side. This requires IBM VSD to allocate its own set of pinned kernel buffers and use KLAPI services to prepare these buffers for direct DMA.

From Figure 64, we can see where we avoid extra copies with KLAPI. On the left hand side, we can see that one copy is between the application buffer, specified on the read or write call, and the GPFS buffer cache. The other copy occurs in VSD-client between the GPFS cache and CSS IP interface managed cluster buffer. The CSS adapter then uses DMA to access this cluster buffer. On the right hand side, using KLAPI, the data copy in VSD is avoided. To eliminate this copy, DMA must be used between communication adapters and the GPFS buffer cache.

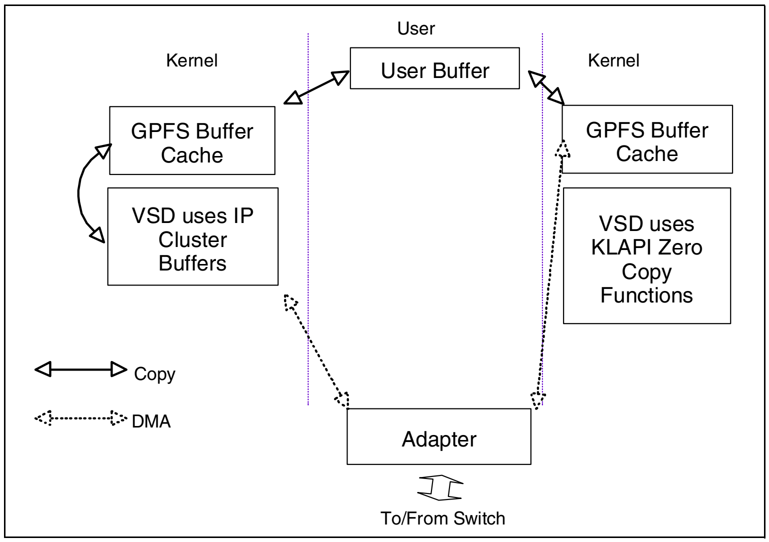

Figure 64. Comparison of GPFS/VSD data flow using KLAPI vs. IP

3.1.2.1 Read Flow

When a read call is made from a GPFS/VSD client, the client prepares and posts a buffer for DMA, thus, obtaining a ptag. Then, it sends a read request with buffer *ptag* to the VSD Server. On the Server, the VSD LAPI completion handler starts I/O read and saves the *ptag*. The VSD I/O handler then sends data with saved *ptag.* Once completed, the DMA data is passed to the Client from the VSD buffer, which then writes to the subsystem buffer. The VSD LAPI completion handler will thus initiate I/O done processing.

Figure 65 shows the steps involved when a read call is made from within GPFS/VSD when KLAPI is enabled.

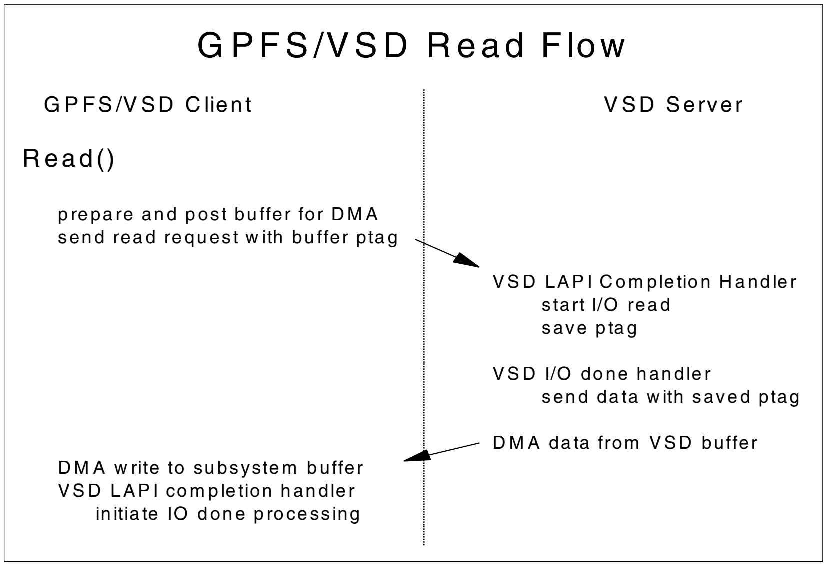

Figure 65. GPFS/VSD Read Flow using KLAPI

3.1.2.2 Write flow

When a write call is initiated from a GPFS/VSD client, it prepares and posts a buffer for DMA and sends a write request to the VSD Server. On the Server, VSD sends zero copy LAPI_zGet request to the Client, which then sends data using DMA back to the Server, where the VSD LAPI completion handler starts the I/O write. In this case, the ptag is internally generated by the KLAPI subsystem.

Figure 66 shows the steps involved when a write call is made from within GPFS/VSD when KLAPI is enabled.

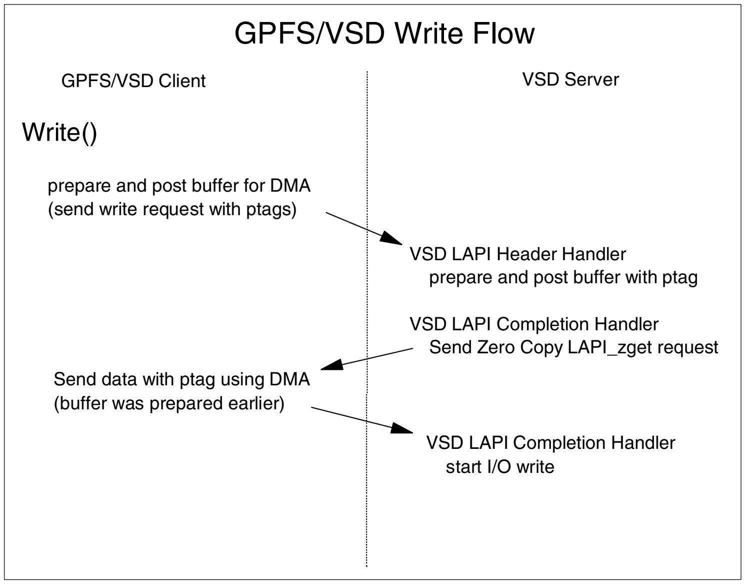

Figure 66. GPFS/VSD Write Flow using KLAPI

3.1.3 Command updates

With IBM Virtual Shared Disk Version 3 Release 2 supporting the use of KLAPI, several commands have been updated to take effect of this change.

3.1.3.1 ctlvsd

Sets the operational parameters for the IBM Virtual Shared Disk subsystem on a node.

```
ctlvsd -l on|off
```

- Enable or disable KLAPI on a node. The initial state is to have KLAPI disabled. This command can be used only after the device driver is unloaded.

Figure 67 shows KLAPI being enabled on a VSD designated node.

```
[root@sp6n05:/]# ctlvsd -l on
css0KLAPI=Enable
[root@sp6n05:/]#
```

Figure 67. Example of KLAPI being enabled for VSD on a node

Figure 68 shows KLAPI being disabled on a VSD designated node.

```
[root@sp6n05:/]# ctlvsd -l off
css0KLAPI=Disable
[root@sp6n05:/]#
```

Figure 68. Example of KLAPI being disabled for VSD on a node

3.1.3.2 lsvsd

Displays configured VSDs and their characteristics.

`lsvsd -i`

- This will show “KLAPI addresses” as soon as there has been communication between those nodes.

Figure 69 shows an example of this command being run.

> **Note**
>
> This command only gives the KLAPI address for the VSD designated node you run it from. The other designated nodes will show their IP address.

```
[root@sp6n05:/]# lsvsd -i
node              IP address
  5             KLAPI[   4]
  7             192.168.16.7
 10             192.168.16.10
[root@sp6n05:/]#
```

Figure 69. Example of lsvsd -i

3.1.3.3 statvsd

Displays IBM Virtual Shared Disk device driver statistics of a node.

statvsd

- DMA space shortages - new statistic.

The heading has been changed to incorporate the KLAPI interface.

Figure 70 shows the output from the new version of statvsd.

```
[root@sp6n05:/]# statvsd
VSD driver (vsdd): KLAPI interface:     PSSP Version:3  Release: 2

         9 vsd parallelism
     61440 vsd max IP message size
         0 requests queued waiting for a request block
         0 requests queued waiting for a pbuf
         0 requests queued waiting for a cache block
         0 requests queued waiting for a buddy buffer
       0.0 average buddy buffer wait_queue size
         0 rejected requests
         0 rejected responses
         0 rejected no buddy buffer
         0 rejected merge timeout.
         0 requests rework
         2 indirect I/O
         0 64byte unaligned reads.
         0 comm. buf pool shortage
         0 DMA space shortage
         2 timeouts
retries: 3 3 3 3 3 3 3 2 2
        25 total retries
Non-zero Sequence numbers
 node#        expected        outgoing   outcast?       Incarnation: 0
     7              13              77

 2 Nodes Up with zero sequence numbers: 5 10
[root@sp6n05:/]#
```

Figure 70. Example of statvsd

3.1.4 KLAPI performance

An important consideration of using KLAPI with GPFS/VSD is how it compares against user space (this is comparing raw KLAPI performance versus LAPI user space performance). KLAPI can be considered in the non-zero copy case and the zero copy case. The three most important performance areas to consider are latency, bandwidth and CPU utilization. From testing done, the following conclusions have been arrived at:

Latency: Comparable to user space

Bandwidth:

- Non-zero copy case: Less than user space due to avoidance of floating copy point
- Zero copy case: Much better than user space, limited only by the interconnect speed

CPU utilization:

- Non-zero copy case: Comparable with user space
- Zero copy case: Much better than user space

So we can see that implementing KLAPI provides better performance than using user space.

Note

KLAPI in PSSP 3.2 will not support zero copy functions on micro channel nodes. Nodes with SP Switch adapters will still see performance benefits from flow control; however, they will not see CPU utilization benefits.

3.1.5 Benefits of implementing KLAPI

With the introduction of KLAPI into the GPFS/VSD hierarchy, and with systems becoming larger and data being written and read growing all the time, there are large benefits to using KLAPI rather than IP.

- KLAPI provides an efficient, reliable transport layer for kernel based subsystems with performance far superior than IP.
- GPFS/VSD incur two copies on each end for each data transfer using IP.
- KLAPI provides zero copy functions.
- Provides a one-sided programming model for kernel-based subsystems, which is typical in client/server types of communication.
- Flow control on writes because of the GET model.

3.1.6 Guidelines

When implementing KLAPI, there are certain guidelines that the user should take into consideration. These guidelines are intended to give the user directions in which they can use the protocol:

- KLAPI is not supported for use by Oracle.
  - As a new protocol, KLAPI should only be turned on for technical markets. This is a testing limitation only.
- VSD use of KLAPI requires RVSD.
- KLAPI can only be turned on/off only when VSD is not active.
- VSD use of KLAPI always uses buddy buffers (even for small requests).

3.1.7 Coexistence

IBM VSD 3.2, enabling the use of KLAPI, will continue to support IP protocol. Thus, nodes running IBM VSD 3.2 *can* interact with nodes running older versions of IBM VSD. With this interaction, one can migrate single nodes to IBM VSD 3.2. KLAPI protocols *can only* be used between nodes with VSD 3.2 installed.

3.2 IBM Concurrent Virtual Shared Disks

IBM Virtual Shared Disk includes concurrent disk access, which allows you to use multiple servers to satisfy disk requests by taking advantage of the concurrent disk access environment supplied by AIX. In order to use this environment, VSD uses the services of Concurrent Logical Volume Manager (CLVM), which provides the synchronization of LVM and the management of concurrency for system administration services.

Concurrent disk access extends the physical connectivity of multi-tailed concurrent disks beyond their physical boundaries. You can configure volume groups with a list of Virtual Disk Servers. Nodes that are not locally attached have their I/O distributed across these servers.

In the example shown in Figure 71, Nodes 1 and 2 are not attached to any disk to access disk1. You can use Nodes 3 or 4. You can access Disk 2 through Node 5 only (or Node 4 when Node 5 fails).

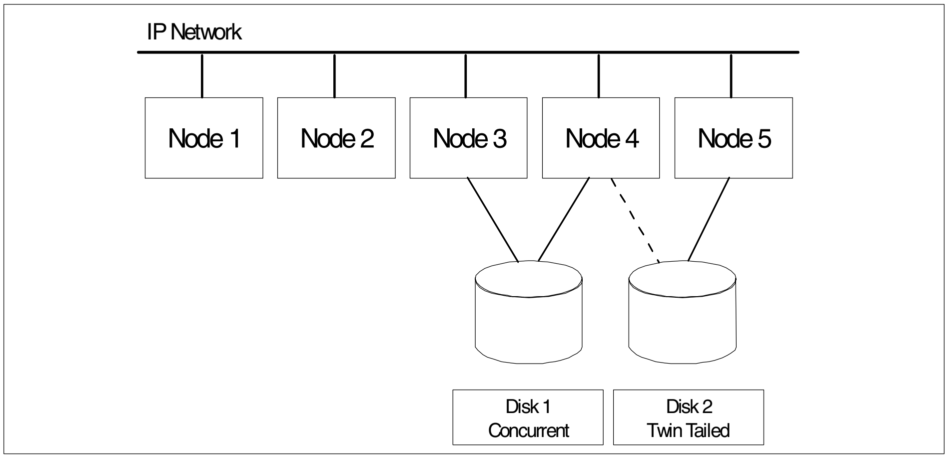

Figure 71. A Concurrent Virtual Shared Disk IP implementation

When you are using IBM CVSD, recovery from node failure is faster because the failed node is marked as unavailable to all other nodes, and its access to the physical disk is fenced. This procedure is faster than the recovery procedure followed in the twin-tailed environment. An additional benefit from multiple VSD servers is that disk services can be spread across multiple servers.

3.2.1 Guidelines

With the introduction of CVSD, there are certain guidelines that we suggest you follow and adhere to. These are for the intention of error free planning and to prevent misconceptions.

- Only 2-way CVSD is supported.
  - This is a current limitation due to CLVM.
  - There is no mirror write consistency. We suggest using RAID 5.
- CVSD is only supported for SSA.
- CVSDs do not support VSD caching.
- IBM High Availability Cluster Multi-Processing for AIX (HACMP) and VSD both use CLVM. If both are installed, only one product is permitted to provide concurrent disk access.

3.2.2 Configuring

Once a server node is defined as a part of a cluster, CVSD starts the clvmd daemon on the server node via the `vsdnode` command. Installation of CVSD makes changes to inittab to start clvmd, if the node is part of a cluster, so that clvmd comes up when the node is rebooted. CVSD has a library named 'libclstr.a' which provides the interface between CVSD and CLVM. The interfaces are mainly required to provide node numbers to CLVM as needed.

3.2.2.1 Planning

It is vitally important to plan your VSDs before creating/defining/upgrading them. Please refer to *IBM Parallel System Support Programs for AIX: Managing Shared Disks*, SA22-7349, Chapter 2, Installing the Shared Disk Management Components of PSSP.

Important considerations are:

- Naming convention used for VSDs, volume groups, and clusters.
  - Make sure these are easily distinguishable and have meaning.
- How many VSDs, what size, which twin-tailed disk(s) will they use?
  - What are the VSDs going to be used for? GPFS? What are the requirements? Do you have enough disks? Are you going to set up RAID?
- When migrating, ensure that you will run the `rvsdrestrict` command.
  - This is important to make sure that all nodes in a partition are running with the same version of VSD. All nodes in a partition do not have to be running with the same version of VSD. However, if you intend to use KLAPI, ensure all nodes are running VSD 3.2.

3.2.2.2 Creating/defining VSDs

Like VSD, CVSD has paths to do the following:

- Using `createvsd`.
- Using `vsdvg` and `definevsd`

1. `createvsd` will create the concurrent volume group and logical volume(s), activate them on all of the concurrent server nodes specified, define the global volume group as concurrent capable, and define CVSDs.

2. `vsdvg` will define the volume group as concurrent capable. Therefore, all the VSDs defined in that global volume group will be CVSDs.

Note:

A user will *not* be able to create concurrent volume groups if the node was not defined as part of a cluster group using the `vsdnode` command.

3.2.2.3 Starting the VSDs

`resumevsd` supplies a list of server nodes to the VSD driver for doing I/O. If the volume group is not varyon'd on a server node, then `resumevsd` will fail as it does with previous versions.

Note:

CVSDs **do not** support cache. When you create a CVSD, it automatically selects No Cache.

3.2.2.4 CVSD and RVSD

RVSD scripts have been changed with the implementation of CVSD.

1. Node Down:

   Normally, when a node goes down, the volume groups served by that node are varied online to the backup node (if defined and available), and I/O to the VSDs is resumed to the backup node. In the case of CVSD, the volume group will already be online to another server; so, the I/O can simply be resumed to the active server. To handle false failures and maintain data integrity, CVSD will fence off the disks (using SSA fencing) so that there will be no I/O on the node that went down.

2. Node Reintegration:

   Normally, when a primary node comes back up, the volume groups are varied back online to it, and I/O is resumed to the primary node. In the case of CVSD, when any CVSD server comes back up, the volume group will be concurrently varied online to it, and I/O will be resumed using all the active CVSD servers.

Note:

In both cases, all of the VSDs served by the new server node will be made active on the new server node. And, the new node will be unfenced if it was fenced so that the new node can access the SSA disks.

3.2.2.5 Scenario

Here, we set up a test scenario to show how CVSD is of benefit.

One CVSD, called vsdname1n5, has a server list of two nodes, sp6n05 (node 5) and sp6n07 (node 7). One node, sp6n10 (node 10), is an additional VSD client. The aim is to stop rvsd on one of the server nodes to show the change made to the server list.

In Figure 72, we show the output of `lsvsd -l` from node 07, the VSD client. This has a server list of nodes 5 and 7.

```
[root@sp6n07:/]# lsvsd -l
minor  state server lv_major lv_minor vsd-name
option      size(MB)      server_list
 1      ACT    7      37       2      vsdname1n5
nocache        300         7,   5
 3      ACT    5       0       0      gpfs1vsd
nocache       8672         5
 4      ACT    7      39       1      gpfs2vsd
nocache       8672         7
 5      ACT    5       0       0      gpfs3vsd
nocache       8672         5
 6      ACT    5       0       0      gpfs4vsd
nocache       8672         5
[root@sp6n07:/]#
```

Figure 72. Initial output from lsvsd -l

Now, we stop the RVSD subsystem on node 5 to simulate a node down scenario. In Figure 73, we run `ha.vsd stop` to stop the RVSD subsystem.

```
[root@sp6n05:/]# ha.vsd stop
0513-044 The rvsd Subsystem was requested to stop.
ha.vsd: Tue Feb  8 13:23:00 EST 2000 Waiting for 19116 to exit.
ha.vsd: Tue Feb  8 13:23:05 EST 2000 19116 has exited.
[root@sp6n05:/]#
```

Figure 73. Stopping RVSD on node 5

We then query the VSD output on node 7 to see the new server list.

Figure 74 shows the output from `lsvsd -l` on node 7. There is now only node 7 in the server list, but the VSD is still ACTIVE.

```
[root@sp6n07:/]# lsvsd -l
minor  state server lv_major lv_minor vsd-name
option      size(MB)      server_list
 1      ACT    7      37       2      vsdname1n5
nocache        300         7
 3      ACT    7      38       1      gpfs1vsd
nocache       8672         7
 4      ACT    7      39       1      gpfs2vsd
nocache       8672         7
 5      ACT    7      40       1      gpfs3vsd
nocache       8672         7
 6      ACT    7      41       1      gpfs4vsd
nocache       8672         7
[root@sp6n07:/]#
```

Figure 74. The output from lsvsd -l on node 7 for node down scenario

We now bring the RVSD subsystem back up on node 5 and reintegrate the server node. We can do this by running the `ha.vsd start` command shown in Figure 75.

```
[root@sp6n05:/]# ha.vsd start
0513-059 The rvsd Subsystem has been started. Subsystem PID is 14394.
[root@sp6n05:/]#
```

Figure 75. Restarting RVSD on node 5

We now check to see if this has been reintegrated successfully by querying the VSD status on node 7.

Figure 76 shows that this has been successful since the server list for the CVSD is back to being 5 and 7.

```
[root@sp6n07:/]# lsvsd -l
minor  state server lv_major lv_minor vsd-name
option      size(MB)      server_list
 1      ACT    7      37       2      vsdname1n5
nocache        300         7,   5
 3      ACT    5       0       0      gpfs1vsd
nocache       8672         5
 4      ACT    7      39       1      gpfs2vsd
nocache       8672         7
 5      ACT    5       0       0      gpfs3vsd
nocache       8672         5
 6      ACT    5       0       0      gpfs4vsd
nocache       8672         5
[root@sp6n07:/]#
```

Figure 76. The output from lsvsd -l on node 7

From this example, we are able to see that the VSD remains ACTIVE while a server node is unavailable. This is a significant enhancement and reduces VSD down time.

3.2.3 System Data Repository changes

With the introduction of CVSD, some changes have been made to the way the System Data Repository (SDR) stores the information for VSDs.

3.2.3.1 New SDR Class VSD_Cluster_Info

A new table VSD_Cluster_Info is used to store the cluster information.

VSD_Cluster_Info contains the following fields:

- node_number
- cluster_name
- CVSD_node_number
- cvgs_defined

Figure 77 shows how the table looks.

```
[root@sp6n05:/]# SDRGetObjects VSD_Cluster_Info
node_number  cluster_name CVSD_node_number cvgs_defined
          5 VSDCluster              0            1
          7 VSDCluster              1            1
[root@sp6n05:/]#
```

Figure 77. Example of VSD_Cluster_Info

3.2.3.2 New fields in VSD_Global_Volume_Group

There are two new fields in the table VSD_Global_Volume_Group:

- server_list.
- vsd_type.

Figure 78 shows an example of the table and its new fields.

```
[root@sp6n05:/]# SDRGetObjects VSD_Global_Volume_Group
global_group_name local_group_name primary_node secondary_node
eio_recovery recovery     primary_ts   secondary_ts server_list
vsd_type
itsovsdvgn5cvsd itsovsdvg            5 ""                  0
0 ""           ""          5:7          CVSD
gpfs1gvg     gpfs1vg               5           7           1
0 389a12880347bce2 389a12880347bce2 0            VSD
gpfs2gvg     gpfs2vg               7           5           1
0 389a12bd170d10c8 389a12bd170d10c8 0            VSD
gpfs3gvg     gpfs3vg               5           7           1
0 389a12cc0ebc8ea3 389a12cc0ebc8ea3 0            VSD
gpfs4gvg     gpfs4vg               5           7           1
0 389a12db1ad872a3 389a12db1ad872a3 0            VSD
[root@sp6n05:/]#
```

Figure 78. Example of VSD_Global_Volume_Group

3.2.4 CVSD external commands

With the introduction of CVSD, many of the VSD-specific commands have been changed to take effect of this.

Refer to *IBM Parallel System Support Programs for AIX: Command and Technical Reference*, SA22-7351 for further information and complete syntax of commands.

3.2.4.1 vsdnode

vsdnode enters IBM Virtual Shared Disk information for a node, or series of nodes, into the SDR:

```
vsdnode ...[cluster_name]
```

[cluster_name] has been added.

A cluster name must be specified for server nodes that will be serving CVSDs. The cluster name can be any user provided name. A node can only belong to one cluster. When you have a CVSD environment, the two servers must both specify the same cluster name.

3.2.4.2 vsdvg

vsdvg defines a VSD global volume group.

The syntax has been changed to:

```
vsdvg ...{-l list_of_servers local_group_name | local_group_name
primary_node [secondary_node]}
```

The `-l` option has been introduced to define the list of servers for CVSD. If there are more than one servers, it would imply that this global_volume_group is a concurrent volume group.

3.2.4.3 createvsd

Creates a set of VSDs, with their associated logical volumes, and puts information about them into the SDR:

```
createvsd ...[-t vg_type] -n {node_list | ALL}
```

- `-t VSD | CVSD`. The default is VSD.
- `-n node_list`
  - For VSD: [Primary/Secondary]:hdisk_list
  - For CVSD: [Server1/Server2]:hdisk_list

3.2.4.4 lsvsd

lsvsd displays configured VSDs and their characteristics:

```
lsvsd
```

- `-l` lists the server_list for CVSD. The new field is added to the end.

Figure 79 shows an example of the output from `lsvsd -l`.

```
[root@sp6n05:/]# lsvsd -l
minor  state server lv_major lv_minor vsd-name
option      size(MB)      server_list
 1      ACT    5      39       2      vsdname1n5
nocache        300         7,   5
 3      ACT    5      40       1      gpfs1vsd
nocache       8672         5
 4      ACT    7       0       0      gpfs2vsd
nocache       8672         7
 5      ACT    5      42       1      gpfs3vsd
nocache       8672         5
 6      ACT    5      43       1      gpfs4vsd
nocache       8672         5
[root@sp6n05:/]#
```

Figure 79. An example of lsvsd -l

3.2.4.5 vsdatalst

vsdatalst displays IBM Virtual Shared Disk subsystem definition data from the SDR:

`vsdatalst -g`

- Additionally displays the server_list for CVSD.

Figure 80 shows an example of the output from this command.

```
[root@sp6n05:/]# vsdatalst -g
      VSD Global Volume Group Information
                                                Server Node Numbers
Global Volume Group name        Local VG name   primary      backup
eio_recovery     recovery     server_list
vsd_type
------------------------------- -------------- -------      ------
------------     --------    ------------------------------------
--------
gpfs1gvg                    gpfs1vg             5        7
1          0     0                                    VSD
gpfs2gvg                    gpfs2vg             7        5
1          0     0                                    VSD
gpfs3gvg                    gpfs3vg             5        7
1          0     0                                    VSD
gpfs4gvg                    gpfs4vg             5        7
1          0     0                                    VSD
itsovsdvgn5cvsd              itsovsdvg           5        0
0          0     5:7                                  CVSD
[root@sp6n05:/]#
```

Figure 80. Example of vsdatalst -g

`vsdatalst -c`

- Displays the cluster information.

Figure 81 shows an example of the output from this command.

```
[root@sp6n05:/]# vsdatalst -c
       Cluster Table
Node Number    cluster_name
-----------   -------------------
     5         VSDCluster
     7         VSDCluster
[root@sp6n05:/]#
```

Figure 81. Example of vsdatalst -c

3.2.4.6 updatevsdnode

updatevsdnode changes IBM Virtual Shared Disk subsystem options in the SDR.

```
updatevsdnode
```

- -c would change the cluster to which the node belongs.
- NONE would remove the node from the cluster.

3.2.4.7 resumevsd

resumevsd activates an available VSD:

```
resumevsd -p | -b
```

- Both flags are not valid for CVSD:

```
resumevsd -l server_list
```

- Is used to pass the server_list to the driver.

3.2.4.8 defvsd

defvsd designates a node as either having or using a VSD.

It now makes CVSD and cache mutually exclusive.

3.2.5 CVSD internal commands

Some SSA internal commands are available to aid the administration of CVSDs.

> **Note**
>
> These commands are for reference only. They are intended for IBM Service Personnel only. They could be disruptive to the system.

SSA Fencing allows you to add (or remove) the system identified by the Node_Number to the list of system denied access to all the Logical Volumes belonging to the Concurrent Volume Group identified by Local_VG_Name.

- Only root user can issue commands.
- Local_VG_Name is a valid Concurrent VG for the node.
- Node_Number is the same for the SP environment and the SSA Cluster Network.
- (SDRGetObjects Node node_number == lsattr -El ssar -a node_number)

```
fencevg -v -n
```

- Fences all the disks of the volume group from the node specified by the node number.

`unfencevg -v -n`

- Unfences all the disks of the volume group from the node specified by the node number.

`lsfencevg -v`

- Lists the nodes fenced for the volume group.

Figure 82 shows the output from the `lsfencevg -v` command.

```
[root@sp6n05:/]# lsfencevg -v itsovsdvg
Found 0 Fenced Nodes for SSA disk /dev/hdisk9
[root@sp6n05:/]#
```

Figure 82. An example of lsfencevg -v

3.2.5.1 IBM Virtual Shared Disk Perspective

Several enhancements have been made to the IBM Virtual Shared Disk Perspective (`spvsd`) in support of the new virtual shared disk functions.

Support for CVSDs

Changes have been made to some of the notebooks and dialog boxes to reflect how you create and display CVSDs. These reflect the command updates already discussed.

An example of this is when you create a CVSD. A dialogue box appears reminding you to reboot the server nodes to allow concurrent access to be enabled.

VQuery RVSD Subsystem

This is a new feature in the IBM Virtual Shared Disk Perspective. This allows you to gather data about the current status of the RVSD subsystem.

From the nodes pane, select:

Actions...

Query RVSD Subsystem.

This produces the same output as the command line version:

`ha.vsd query`

Figure 83 shows the output from this command.

```
[root@sp6n05:/]# ha.vsd query
Subsystem         Group            PID     Status
 rvsd             rvsd             19116   active
 rvsd(vsd): quorum= 6, active=1, state=idle, isolation=member,
            NoNodes=9, lastProtocol=nodes_joining,
            adapter_recovery=on, adapter_status=up,
            RefreshProtocol has never been issued from this node,
            Running function level 3.2.0.0.
[root@sp6n05:/]#
```

Figure 83. Output from ha.vsd query

3.3 Reliability, availability, and serviceability

Here, we list the reliability, availability, and serviceability improvements that are included in IBM Virtual Shared Disk Version 3 Release 2.

VSD no longer runs in interrupt mode:

- Provides better sharing of system resources.
- Allows dynamic memory allocation.
- Allows device driver to safely use more system functions.

Self tuning:

- The values on the `vsdnode` command, “VSD request count” and “rw request count”, are ignored but kept for compatibility. These items previously set aside pinned kernel memory that can now be obtained dynamically as needed. (Buddy buffers still work as previous.)

Locking Improvements

RVSD logs have been combined into one log, /var/adm/ras/vsd.log

Error report entries improved.

Configuration changes:

- The device driver can be configured without having VSDs already defined.
- Must issue `ucfgvsd VSD0` to unload device driver (that is, the device driver stays loaded after `ucfgvsd -a`).

Chapter 4. GPFS Release 3

IBM General Parallel File System for AIX (GPFS) provides file system services to parallel and serial applications running on the RS/6000 SP. GPFS allows users shared access to files that may span multiple disk drives on multiple SP nodes.

The GPFS file system compared with other types of file systems on the RS/6000 SP, provides:

- Improve system performance
- Assures file consistency
- Increases data availability
- Enhances system flexibility
- Simplifies administration

IBM General Parallel File System for AIX (GPFS) is an enterprise file system for AIX and has the following features:

- Scalability
  - File size - greater than other file systems
  - File system size - greater than other file systems
  - I/O rate - very high
- Data manageability
  - Back-up/archive - by applications, such as TSM (Tivoli Storage Manager)
  - Provides DMAPI interface with Data Management applications
- Availability/failure Survivability in case of failure of:
  - Node
  - Disk
  - Software
  - Communication adapter
  - Disk adapter
- Standards compliance

IBM General Parallel File System for AIX (GPFS) is the response to customer needs and requests. The range of customers is very large.

- Customer applications that require fast, scalable access to large data. These applications could be serial or parallel reading or writing:
  - Seismic data processing
  - Weather forecast
  - ASCI nuclear simulation applications that serve data to visualization engines and other analytical processes
- Environments with very large data, especially when single file servers (such as NFS) reach capacity limits:
  - Digital library file serving
  - Access to large CATIA file sets
  - Large files for Business Intelligence applications
- Customers applications requiring greater job throughput via improved workload and disk capacity balancing:
  - Large aggregate scratch space for commercial or scientific applications
  - Internet serving of database content to users with balanced performance
- Customers needing file systems that survives failures.

In the next sections, we discuss the new features, migration, coexistence and compatibility, DMAPI support, DFS interoperability, and new programming interfaces of GPFS Version 1 Release 3.

4.1 What's new in Release 3?

The new features of IBM General Parallel File System for AIX (GPFS) Release 3 consist of performance, reliability, usability, scalability, security and standards enhancements, or a series of performance and functional improvements.

4.1.1 Terminology changes

The usability of Release 3 is reflected in some terminology changes that will give customer the possibility to use some well known terms. In Release 3, these changes will be used consistently.

The term *Stripe Group* has been replaced by the term *File System*.

The term *GPFS configuration* has been replaced by the term *GPFS nodeset*. A GPFS nodeset is a group of nodes that all run the same level of GPFS and operate on the same file system. There can be multiple GPFS nodesets per SP partition.

The term *stripe group panic* has been replaced by the term *file system forced unmount*. A file system forced unmount is the feature of GPFS that maintains reliability and consistency by forcing an unmount of the file system when an application or event generates an operation that threats user data.

4.1.2 Scalability enhancements

In GPFS Release 3, additional block sizes of 512 KB and 1024 KB are now supported for file systems. These larger block size values may allow you to improve performance by matching RAID stride sizes.

Possible maximum values for file size and file system size have been relaxed. The introduction of multiple levels of indirection allows file sizes up to largest supported GPFS file systems size. The maximum indirection level supported by IBM Service is three.

The maximum file system size supported by IBM Service has increased to 9 TB. However, the value of a file systems size is a testing limit, and the real limit is supposed to be much higher.

With use of multiple levels of indirection, GPFS now internally controls the values for indirect block size and i-node size. System control of these values allows for more effective caching of i-nodes and may improve the performance of some applications.

In GPFS Release 3, the i-node for newly created GPFS file systems is 512 bytes and is no longer configurable. The -I *IndirectSize* and -i *InodeSize* options have been removed from the `mmcrfs` command. However, older file systems, created under previous releases, are supported under Release 3.

The maximum number of files has increased because the i-node file is larger.

The maximum number of file systems that may exist within a GPFS nodeset is 32.

The maximum number of disks in a GPFS file system is 1024.

4.1.3 Administration rework

Some rework and some changes had to be done in GPFS Release 3 to make the administration more flexible and easier to use. Some new commands had to be added to improve file system consistency. Most GPFS administration tasks can be performed using either SMIT menus or by entering command strings from GPFS nodes or from the control workstation (CWS).

4.1.3.1 SDR changes

GPFS commands save configuration and file system information in files in the system data repository (SDR). These files are accessible from any node in the local partition. In GPFS Release 3, the format of the SDR files has changed. The main objective of these changes is to reduce the traffic with the SDR files and, consequently, increase the performance of the GPFS file systems.

In GPFS Release 3, no SDR files are changed during the install process, but new files are created to reflect the SDR changes. The necessary conversions are handled automatically by the GPFS administration commands each time a command is issued.

If you have a previous release GPFS file system, and you migrate to Release 3, old style SDR files are maintained to allow reversion until they are explicitly deleted.

In GPFS Release 3, rules for SDR access have also changed. It is not possible anymore to edit the SDR files with `vi`, instead you should use the `mmchconfig` command.

If you want to modify the SDR files, you must have the required credentials for SDR access. For further information about security access, see Section 1.7, “Security management” on page 40, and *IBM Parallel System Support Programs for AIX: Administration Guide*, SA22-7348.

4.1.3.2 Commands changes

The `mmcrvsd` command is now available to create virtual shared disks for use by GPFS. The `mmcrvsd` command creates VSDs in the form used most often by GPFS (one VSD per LV per physical disk). It is suggested that this command be used to create virtual shared disks for the `mmcrfs`, `mmadddisk`, `mmchdisk`, and the `mmrpldisk` commands.

A new command, the `mmdefragfs` command, is now available to reduce fragmentation, thereby potentially increasing the number of full free blocks available to the file system. This defragmentation utility is very useful for users of small files.

The `mmshutdown` command is now available to cleanly unmount all GPFS file systems and shutdown GPFS either on a single node or across a set of nodes. Starting from GPFS Release 3, this is the correct way to stop GPFS services.

The `mmchdisk` command has been enhanced to allow all disks belonging to a file system to be resumed or started at the same time. To make this possible, the `mmchdisk` command now has a -a flag to restart all disks.

The `mmcrfs`, `mmadddisk`, and `mmrpldisk` commands no longer create and configure virtual shared disks. The `-i` *IndirectSize* and `-i` *InodeSize* option parameters have been removed from the `mmcrfs` command, and it is no longer possible to modify the i-node size or the indirect block size. GPFS now internally controls these values. However, older file systems, created under previous releases, are supported under GPFS Release 3.

The `mmadddisk`, `mmchdisk`, `mmdeldisk`, `mmrpldisk`, `mmrestripefs` commands have been enhanced to allow the user to specify which nodes of a GPFS nodeset will participate in the restripe of a file system. If it is not specified, the default is to use all nodes. These commands can be used to do a parallel restripe and will improve the speed of disk offload.

The `mmaddnode`, `mmchconfig`, `mmconfig`, `mmcrfs`, `mmdelnode`, and `mmlsnode` commands are now executable from any node running GPFS or from the control workstation. Each of these GPFS commands now has a -C *nodeset_id* option that allows you to designate which GPFS nodeset the command is to act upon. If this option is *not* specified when issued from a node, the command will act upon the nodeset to which the issuing node belongs. This option *must* be specified when the command is issued from the CWS, or it will fail.

Because file system names are unique across nodesets, the `mmadddisk`, `mmchdisk`, `mmchfs`, `mmchmgr`, `mmdefragfs`, `mmdeldisk`, `mmdelfs`, `mmdf`, `mmfsck`, `mmlsdisk`, `mmlsfs`, `mmlsmgr`, `mmrestripefs`, and `mmrpldisk` commands can be executed from any node running GPFS or from the CWS.

The `mmcheckquota` command has been expanded to allow rebuilding of a restored copy of the user and group quota files for a GPFS file system.

In GPFS Release 3, the *priority* parameter on the `mmconfig` and `mmchconfig` commands is no longer used. If upon initialization of the GPFS daemon this parameter is encountered in an old *mmfs.cfg* file, it will be silently ignored.

The *mallocSize* parameter is no longer used. The `mmconfig` command, the `mmchconfig` command, and the */usr/lpp/mmfs/samples/mmfs.cfg.sample* file reflect this. If upon initialization of the GPFS daemon this parameter is encountered in an old *mmfs.cfg* file, it will be silently ignored. If you upgrade to GPFS Release 3, the behavior of existing file systems will be preserved as long as both the *mallocSize* and *maxFilesToCache* parameters were changed proportionally in the older version of GPFS.

GPFS creates a number of cache segments on each node in the SP system. The amount of cache is controlled by three parameters:

- pagepool - The amount of pinned memory reserved for caching data read from disk.
- maxFilesToCache - The total number of different files that can be cached at one time.
- maxStatCache - This parameter sets aside additional memory to cache attributes of files that are not currently in the regular file cache.

The number of i-nodes cached is controlled by the *maxFilesToCache* parameter. The number of i-nodes for recently used files is constrained by how much the *maxFilesToCache* parameter exceeds the current number of open files in the system. However, you may have open files in excess of the *maxFilesToCache* parameter.

A stat cache entry is about 128 bytes, which is significantly less memory than a full i-node, and this value may be changed via the *maxStatCache* parameter. The default value is *4 x maxFilesToCache*.

The GPFS administrator controls the size of these caches through the `mmconfig` and `mmchconfig` commands.

All commands' internal code has been reviewed to be more efficient and to be performance oriented.

In GPFS Release 3, the GPFS File System Manager may be migrated to a specific node.

4.1.4 Security enhancements

In PSSP 3.2, a new authentication and authorization method is possible to use, and this method is DCE. GPFS Release 3 has the capability to exploit this method.

If PSSP is configured to use DCE security, GPFS will authenticate itself to the other nodes in the nodeset using DCE. This requires correct setup in the PSSP security environment.

GPFS Release 3 also exploits the SP Security Services Library routines that are included in the PSSP software.

GPFS Release 3 accesses the SDR and uses `sysctl` in the execution of some commands. For further information on SDR access, see *IBM Parallel System Support Programs for AIX: Administration Guide*, SA22-7348.

To perform most of GPFS administration tasks, you should have:

- Root authority - This is required to perform all GPFS administration tasks except those with a function limited to listing GPFS operating characteristics or modifying individual file attributes.
- Kerberos authentication - This is required to perform most GPFS administration tasks. If you do not have Kerberos authentication, issue the `k4list` command. If your authentication method for SP Security Services has been set to DCE, `dce_login` authentication is required. For further information, see the *IBM Parallel System Support Programs for AIX: Administration Guide*, SA22-7348 and *IBM Parallel System Support Programs for AIX: Command and Technical Reference*, SA22-7351.
- If you are performing GPFS administration tasks from the CWS on a multi-partitioned system, the SP_NAME environment variable must be properly set. It is suggested that you use a separate window for each partition, setting the environment variable accordingly. For further information, see the *IBM Parallel System Support Programs for AIX: Administration Guide*, SA22-7348.

4.1.5 Performance improvements

One of the major reasons for the existence of the GPFS file systems is the need for performance in the SP environment. The most important features of GPFS Release 3 are performance improvements.

4.1.5.1 MPI-IO support

The Message Passing Interface (MPI) standard defines a set of I/O interfaces for use by parallel programs solving a single problem and sharing data. GPFS Release 3 provides a series of extended interfaces that will allow the MPI layer to specify characteristics of its data access and allow GPFS to optimize its access to data. In Release 3, we have the following extensions for MPI support:

- Specification of reuse patterns for data. This will allow specification of such patterns as accessed once, multiple use, read followed by update. This will allow GPFS to allocate space more effectively.

- Locality of reference for access. A process may specify that it will access data only within a specified region of the file, will access all of that region exclusively within the parallel application, and will access the region in a strided fashion.
- Specification of a list of nodes that will access the file.

All of these specifications apply to a single open file and persist for the duration of the set of open files. These are not persistent attributes of a file. GPFS will use any hints supplied to optimize performance. However, it will not allow any violation of the access pattern promised in the hints to compromise normal file system semantics for this or any other user of the same data. Operations that are not consistent with the hints may cause a slowdown of operations for all users of the file.

The requirement for using MPI-IO is to do a better job for collective I/O exploiting applications using MPI-IO. Release 3 allows MPI-IO layer and its exploiting applications to specify hints on their access patterns and effectively operate on those hints.

MPI-IO requirements are interpreted as exploiting certain hints that are available at the MPI-IO layer to optimize performance. These hints can be exploited best in the areas of prefetching (or not prefetching) data and acquisition of locks in a way that causes minimal conflicts between MPI-IO agents that have declared their intents and the retention of data that is declared as being likely to be reused or not reused. As described in the MPI-IO standard, GPFS will treat this information as hints to be used within the context of whatever else is operating in the system. Some of these hints will originate at the MPI application level, and some will originate in the MPI layer based on buffer size, stride, and number of nodes. For more information regarding hints and directives, see Section 4.5, “New programming interfaces” on page 159.

A series of programming interfaces were added to GPFS Release 3 that allow the MPI layer of an application to specify characteristics of its data access and allow GPFS to optimize its access to the data. GPFS data shipping is used by MPI-IO to avoid token management overhead. Token manager has been changed to allow multiple tokens to be acquired in a single message.

Using GPFS Release 3 with MPI-IO will optimize I/O throughput for GPFS file systems, and performance of open/close/stat operations on files will increase up to 5-10 times.

4.1.5.2 GPFS LAPI exploitation

The default communication protocol for communication between the nodes in a GPFS nodeset is Transmission Control Protocol/Internet Protocol (TCP/IP). However, you can specify the alternate use of Low-Level Application Programming Interface (LAPI) during the configuration (`mmconfig -P LAPI`) or when changing the configuration (`mmchconfig comm_protocol=LAPI`) of your GPFS system (see the *IBM General Parallel File System for AIX: Guide and Reference*, SA22-7452, for complete information on the GPFS administration commands).

GPFS exploits the Low-Level Application Programming Interface (LAPI) function of the Parallel System Support Program (PSSP). Using LAPI instead of TCP/IP will improve the token manager traffic and the performance of communication between the GPFS daemons. Also, it will reduce latency of file system daemon communications (token, data shipping, and others) by a significant factor (estimated at 2-4 times). It will also help reduce CPU utilization in that it will use the new, non-polling LAPI wait function.

However, it is important to understand what is required when choosing LAPI as the communication protocol. A switch adapter window must be reserved on each node in the GPFS nodeset. To successfully reserve an adapter window for use by GPFS, *all* LoadLeveler jobs must be stopped prior to issuing either the `mmconfig` command or the `mmchconfig` command. If LoadLeveler is not stopped, then the chance of reserving the same adapter window on all nodes for its use is not good. And, if the same adapter window can not be reserved on each node for use by GPFS, the GPFS commands will fail.

After reserving an adapter window on each node, the window cannot be used by any application (even LoadLeveler) until it is released. The adapter window is released only when the node is deleted from the GPFS nodeset. To delete the node, you must stop the GPFS daemon on all the nodes in the nodeset and specify the -c option on the `mmdelnode` command.

In GPFS Release 3, it is now possible to use KLAPI protocol on the VSD exploitation of this protocol. PSSP 3.2 now provides Kernel versions of both HAL (Hardware Abstraction Layer) and LAPI (Low-level communication API) products, and these versions are KHAL and KLAPI, respectively. For further information regarding KLAPI functionality, see Section 3.1, “Kernel low-level Application Programming Interface” on page 120.

4.1.5.3 Small files and metadata performance

In GPFS Release 3, there are significant performance improvements in small files and metadata handling. These improvements are reflected in the following:

- Prefetch of i-nodes on patterns, such as the `ls -l` command
- Implementation of a stat cache for repeated access to stat data
- More effective flush of i-nodes to avoid synchronous disk operations
- More effective management of the GPFS log to avoid log full conditions
- Faster file create
- Improved token management, including prefetch of tokens

The `mmcrfs` command has been speeded up by a great deal. The `mmcrvsd` command runs parallel VSD creates and is restartable.

4.1.5.4 Large file performance

Large file performance is mostly a function of the processors, the VSD, and the disks. Significant improvements here include:

- VSD KLAPI
- Newer processors
- SP Switch 2
- 10000 RPM disks with disk caching

GPFS KLAPI eliminates the problem of VSD overruns (writes, no reads) that existed in previous releases.

4.2 Migration, coexistence, and compatibility

The following sections highlight hardware and software requirements, migration, coexistence, and compatibility considerations.

4.2.1 Hardware and software requirements

The GPFS Release 3 runs on a system with the following hardware requirements:

- RS/6000 SP
- SP Switch or SP Switch 2
- Sufficient disk capacity to support the GPFS file systems

The following software products are required for GPFS Release 3:

- AIX 4.3.3.14 or later - Provides basic operating system and the routing of file system calls requiring GPFS data
- PSSP 3.2 or later - Provides the IBM Virtual Shared Disk components, the Group Services components, IBM Recoverable Virtual Shared Disk components, and the SP Security Services Library routines
- GPFS 1.3 filesets - Provides the main software components for GPFS

For complete hardware and software requirements, see *IBM General Parallel File System for AIX: Installation and Tuning Guide*, GA22-7453.

4.2.2 Migration considerations

Due to the optimization in the token manager function in GPFS 1.3, it is required that all nodes that use a given file system are at the same level of GPFS. GPFS Release 3 will also require that all nodes upgrade to the new release at the same time.

Some new functions in GPFS 1.3, such as DMAPI or multiple level of indirection, create data structures that are not recognized by previous GPFS releases.

In short, the procedure for migration to GPFS Release 3 is the following:

- Save the *mmfs.cfg* file. Some configuration options are available in Release 3 that are not available in older releases. This should be done if you want to go back to previous releases.
- Stop GPFS on all nodes.
- Install the new GPFS code on all nodes in a nodeset.
- Reboot all the nodes in a nodeset. This will activate the new code.
- Run GPFS Release 3 code using the previous releases disk images. No changes will be made to the disk that previous releases cannot interpret. If you create a new file system, this file system cannot be read by previous releases.
- When ready to make the change to GPFS Release 3 permanent, run the `mmchfs -V` command, which will enable the newer functions.

Until the `mmchfs -V` command is run, it will be possible to install older releases over GPFS Release 3 and reboot the nodes using the same file systems. Any attempt to do this after running the `mmchfs -V` command will require re-creation of the file systems and restoration of saved copies of the data. For further information regarding migration, see *IBM General Parallel File System for AIX: Installation and Tuning Guide*, GA22-7453.

In GPFS Release 3, if you want to use the Concurrent Virtual Shared Disk function with an existing GPFS file system, you must first create a new virtual shared disk specifying the concurrent option via either the `mmcrvsd` command or the procedure outlined in *IBM Parallel System Support Programs for AIX: Managing Shared Disks*, SA22-7349. You may then replace the old, non-concurrent virtual shared disk via the `mmrpldisk` command.

4.2.3 Coexistence considerations

GPFS Release 3 changes the locking semantics that control access to data, and, as a result, all the nodes should be at the same level. It will not be possible for Release 3 and another older release to coexist in the same nodeset.

In GPFS Release 3, all nodes in a GPFS nodeset must be in the same partition. A GPFS file system may only be accessed from within a single SP partition. Note that you can use a GPFS file system from another partition by mounting it via NFS or DFS.

In order to use LAPI as the communication protocol, all nodes in the GPFS nodeset must use LAPI. There is no coexistence between LAPI and TCP/IP.

4.2.4 Compatibility

All applications that run with older releases of GPFS will continue to run with GPFS 1.3.

File systems created under older releases of GPFS may continue to be used under GPFS 1.3

However, once a file system created under older releases of GPFS has been explicitly changed to GPFS 1.3 by issuing the `mmchfs -V` command, the disk image can no longer be read by an older release of GPFS.

4.3 DMAPI support

The Data Management Application Programming Interface (DMAPI) is the implementation of the X/Open Data Storage Management standard (XDSM) for the General Parallel File System (GPFS). It is intended to allow vendors of storage management applications, such as TSM, to provide Hierarchical Storage Management (HSM) functions for GPFS file systems.

The HSM solution has been adopted due to the large number of requests from SP customers to store more data in the file system than they have in disk storage. The general idea is that the files that are less frequently used are migrated to lower performance storage (usually tape) and recalled to disk upon access without operator intervention.

The Data Management Application Programming Interface (DMAPI) for GPFS allows the Data Management (DM) application:

- To monitor events associated with a GPFS file system or with an individual file
- To manage and maintain file system data - backup, archive, and data migration to and from library tapes without user intervention

All mandatory DMAPI functions, and most optional functions that are defined in the XDSM standard, are implemented in DMAPI for GPFS. GPFS Release 3 has implemented all optional functions required by Tivoli Storage Manager (TSM).

GPFS funnels all DMAPI events to a single node because there is no known DM application that would be capable of parallel processing events.

Figure 84 shows the GPFS DMAPI interaction.

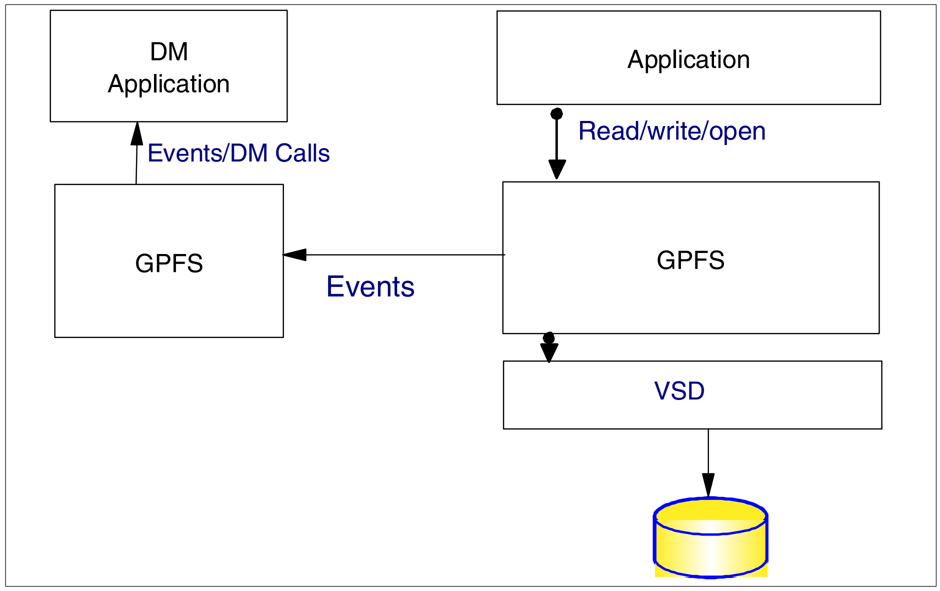

Figure 84. GPFS DMAPI interaction

The XDSM standard is mainly intended for a single-node environment. Some of the key concepts in the standard, such as sessions, event delivery, mount and unmount, DM attributes, quota, and failure and recovery, are not well defined for a multi-node environment, such as GPFS.

The failure model in XDSM is intended for a single-node system. There are two types of failure:

- DM application failure - The DM application has failed, but the system works normally.
- Total system failure - The file system has failed, all non-persistent DMAPI resources are lost, and the DM application itself may or may not fail.

In the DMAPI for GPFS, we have compliance with the enhanced DMAPI failure model, in order to support recoverability of GPFS. The failure model for GPFS is different from a single-node environment, such as being assumed in the XDSM standard.

The simplistic XDSM failure model is inadequate for GPFS. Being a multi-node environment, GPFS may fail on one node but survive on other nodes. This type of failure is called *single-node failure* (or partial system failure).

GPFS is built to survive and recover from single node failure, without meaningfully affecting file access on surviving nodes. The types of failure that DMAPI for GPFS can survive are:

- Single node failure
  - Session node failure
  - Source node failure
- Session failure and recovery
- Event recovery
- Loss of access rights
- DM application failure

DMAPI must be enabled individually for each GPFS file system. If the file system was created with a release of GPFS earlier than GPFS 1.3, the file system descriptor must be upgraded before attempting to enable DMAPI by using the `mmchfs -V` command. For further information regarding DMAPI for GPFS implementation, see *IBM General Parallel File System for AIX: Data Management API Guide*, GA22-7435.

4.4 DFS interoperability

Distributed File Service (DFS) interoperability requires DFS for AIX Version 3 Release 1 or later.

Starting from GPFS Release 3, GPFS and DFS for AIX may interoperate to support the export of a GPFS file system to DFS clients. A GPFS file system may be exported from one node using the DFS export protocol. After export, normal access to the file system may proceed from the SP nodes or DFS client nodes.

A GPFS file system exported via DFS is a secure export mechanism (which NFS is not) to support RS/6000 SP and GPFS users.

To export a GPFS file system via DFS, you have to:

- Create and mount the GPFS file system.
- Ensure DCE and DFS are properly configured and running. If the file system to be exported was created under a level of GPFS that does not

support DFS exporting, you must first migrate the file system using the `mmchfs` command with the -V option.

- Export the GPFS file system as a `nonlfs` file set.

GPFS has extended ACL capabilities when exporting via DFS. These capabilities include administering ACLs via DFS ACL commands and supporting extended DFS style ACLs that include the addition of foreign cell entries and additional permissions. You should use DFS ACL commands when administering DFS ACLs and GPFS ACL commands when administering GPFS ACLs.

When exporting a GPFS file system via DFS, additional ACL entry types are possible. These ACL entry types are created when using DFS ACL commands on a GPFS file system object. When exporting a GPFS file system via DFS, ACLs are affected by the default cell.

After DFS exporting a GPFS file system, you may access it from either DFS clients or other GPFS nodes. The other GPFS nodes do not usually utilize DFS or DCE in any form. As such, user access from these nodes to files or directories must be considered local cell access.

In addition to the object ACL protecting the file or directory, DFS ACL commands allow you to specify initial object creation and initial container creation ACLs. These ACLs specify the initial ACL that will be inherited by a new file or directory. GPFS utilizes only one form of ACL inheritance, referred as the default ACL. If DFS style ACLs are no longer necessary for your GPFS file system, they may be removed.

DFS server and client implementations differ in their ability to support very large files. DFS may, therefore, limit your ability to read or write very large files that GPFS can have.

Utilizing the DFS server with a GPFS file system can alter the underlying GPFS cache needs. The DFS server must hold GPFS resources based on a cumulative activity of all DFS clients accessing the server. For performance and scalability reasons, DFS does not aggressively release held resources. Therefore, it is suggested that a DFS exported node should be configured with the maximum data cache and file open cache possible.

GPFS file system exported via DFS requires that DFS obtain GPFS tokens to support its own tokens. This may put a stress on token server storage because of the large number of tokens required. In addition, concurrent update access to the same set of files from a DFS client and a GPFS application will be slow because of the locking required. It is suggested that heavy use of DFS export be avoided at times when maximum GPFS parallel efficiency is desired.

GPFS Release 3 file systems exported via DFS can be managed by a Data Management application as Tivoli Storage Manager (TSM). When using DMAPI to manage the data of a DFS exported file system, the configuration parameter, *dmapiEventTimeout*, can be used to control the blocking of DFS file operation threads that are waiting for a response from a DMAPI event. To avoid deadlocks with DFS locking mechanisms, the Data Management application should not invoke POSIX functions on a DFS exported file system.

Before unmounting a DFS exported GPFS file system, detach the DFS aggregate.

4.5 New programming interfaces

GPFS Release 3 supports additional programming interfaces that may enhance the performance and capability of programs.

GPFS programming interfaces now support both 32-bit and 64-bit applications. In order to use the 64-bit version of the GPFS programming interfaces, you must recompile your code using the appropriate compiler options.

Programs requiring exact modification or access times for files may use some of these interfaces.

MPI-IO uses some of these interfaces and should be considered also when you write an application.

4.5.1 Exact file status implementation

GPFS is designed so that most applications written to the X/Open standard for file system calls can access GPFS data with no modification.

However, there are some exceptions: applications that depend on exact reporting of changes to the mtime, ctime and atime fields returned by the stat() call may not work as expected. For example, the exact file status is needed by some data management applications. Providing exact status of the file as the default would be costly because it would require the revocation of a lot of tokens.

The delayed update of the information returned by the stat() call also impacts system commands, such as `du` or `df`, which display disk usage. The data reported by such commands may not reflect changes that have occurred since the last sync of the file system. For parallel systems, a sync does not occur until all nodes have individually synchronized their data. On a system with no activity, the correct values will be displayed after the sync daemon has run on all nodes.

GPFS now provides gpfs_stat() and gpfs_fstat() calls, which are identical to stat() and fstat() except that they have stronger locks and acquire exact mtime and ctime. These stronger locks may impact performance of parallel applications; so, they are not the default. These fields are guaranteed accurate when the file is closed.

You have to use the *libgpfs.a* library when linking the executables to reach these calls.

4.5.2 ACLs and attributes

GPFS Release 3 adds a series of calls:

- **gpfs_getacl()** and **gpfs_putacl()** - These are used for setting and retrieving the access control information for a GPFS file.
- **gpfs_fgetattrs()** and **gpfs_putfattrs()** - These are used for setting and retrieving all the extended file attributes.

These are calls for the saving and restoring of ACLs by backup products such as Tivoli Storage Manager (TSM). Use of the second form will pick up any extended file attributes that might be assigned to a file in the future as well as ACLs.

You have to use the *libgpfs.a* library when linking the executables to reach these calls.

4.5.3 File access patterns

GPFS attempts to recognize the pattern of accesses that an application makes to open a file and optimizes its behavior accordingly. For example, GPFS can recognize sequential reads and, therefore, prefetch blocks in advance of when they are required by the application. However, in many cases, GPFS does not recognize the access pattern of an application or cannot optimize its data transfers. In these situations, performance may improve if the application explicitly discloses aspects of its access pattern to GPFS via the gpfs_fcntl() library call.

The gpfs_fcntl() library call provides a mechanism for providing hints about access patterns that allow GPFS to optimize disk access and lock patterns.

The gpfs_fcntl() library call allows application programs to pass two classes of file access information, giving GPFS an opportunity to improve throughput and latency of file system requests:

- Hints
- Directives

Hints allow an application to disclose its future accesses to GPFS. Hints are always optional. Adding or removing hints from a program, even incorrectly specified hints, will never alter the meaning of a program. Hints can only affect the performance of an application. However, GPFS is free to silently ignore a hint if system resources do not permit the hint to be processed.

Access range hints tell GPFS that access to a file will be confined to specified ranges of the file and that locks and prefetch should be confined to these ranges. Also, you can use hints to specify completion of use of the file and to allow resources to be freed quickly.

Hints are intended for use by MPI-IO, which ships I/O for specific ranges to specific nodes. This provides hints on intent to read or write these ranges.

In contrast, *directives* are stronger than hints. They may affect program semantics and must be either carried out by GPFS or return an error.

To communicate hints and directives to GPFS, an application program builds a data structure in memory and then passes it to GPFS. The header of hints and directives that follow it are defined as C structures.

4.5.4 Data shipping

In GPFS Release 3, a new mechanism called *data shipping* has been introduced. Data shipping is a mechanism for coalescing small I/Os on a single node for disk efficiency. It's mainly targeted at users who do not use MPI.

Data shipping is specifically targeted at parallel applications that do fine-grain write sharing on files in a GPFS file system.

Data shipping violates POSIX semantics and affects other users of the same file, and, therefore, it should be used only when fine grain write is specially needed.

4.5.5 Memory mapped file capabilities

GPFS Release 3 has support for the following memory mapped file capabilities as described in the X/Open 4.2 standard:

- mmap - Maps a region of a file into an area of the users storage
- munmap - Removes this mapping
- msync - Forces changes in the shared segment to disk

The rationale for implementing these calls is application enablement not an expectation that byte level sharing of files across multiple node should perform well. Therefore, it is acceptable to backend the VMM with a pager that essentially does reads and writes of the file that has been mapped. It is essential that data integrity be maintained; so, appropriate locking must be implemented.

The motivation for mapped file support is the enablement of applications that depend on this support. It is not the integration of the existing GPFS buffer cache support into the memory management capabilities of AIX, which would be a far larger effort.

Users of a memory mapped file who wish to share data at a fine grain will have to synchronize their data at each point where they want to externalize their changes to other users. This is not a mechanism to share segments, such as can be done within a single instance of AIX.

Memory mapped file support is implemented as a backend pager for the AIX Virtual Memory Manager (VMM). This pager has the following characteristics:

- It supports VMM calls for the mapping and unmapping of files. This involves setting up the client segment.
- It supplies pages of files in response to VMM page ins. In order to have a page mapped in a client segment, the node must hold a token on the byte range represented by that page. Any token revoke requires that GPFS invalidate the page in the page table using VMM services. Further access to the mapped page will require a page fault, a reacquire of the token, and a pre-fetch of the page. As a result of this, fine-grained write sharing of mapped files may not perform well and should be avoided. It will, however, work correctly.
- It pages out pages as requested by the VMM. Since the semantic requires that pages being paged out reside on persistent storage, these pages must be written to disk in response to this function.

GPFS Release 3 has support for the *shmat* memory mapped file capability as described in AIX. Release 3 has also support for the *fsize* user process resource limit as defined by *ulimit*.

Chapter 5. LoadLeveler 2.2

IBM LoadLeveler for AIX is a job management system that allows users to run more jobs in less time by matching the jobs' processing needs with the available resources. LoadLeveler schedules jobs, and provides functions for building, submitting, and processing jobs quickly and efficiently in a dynamic environment. LoadLeveler 2.2 has new and enhanced features, mostly in scheduling, GUI support, process tracking, and DCE security.

This chapter introduces new enhancements of LoadLeveler. For more detailed information, refer to *IBM LoadLeveler for AIX: Using and Administering*, SA22-7311.

5.1 Scheduling enhancements

LoadLeveler has been enhanced to meet the requirements of advanced users for a more flexible, accurate, efficient job scheduling mechanism that also has an easy-to-use interface. For this purpose, LoadLeveler adds new scheduling concepts: consumable resources, geometry specification, and blocking.

5.1.1 Consumable Resources

The LoadLeveler scheduler can now schedule jobs based on the availability of specific resources by defining them as consumable resources.

5.1.1.1 New keywords in the configuration file

There are two new keywords for consumable resources in the configuration files: SCHEDULE_BY_RESORCES and FLOATING_RESOURCES.

The SCHEDULE_BY_RESOURCES keyword has a list of consumable resource names as input and specifies which consumable resources to be considered by the LoadLeveler schedulers when dispatching jobs. Each consumable resource name may be an administrator-defined alphanumeric string or may be one of the reserved words, for example, ConsumableCpus, or ConsumableVirtualMemory. These resources are either floating resources or machine resources.

The FLOATING_RESOURCES keyword has a list of resource names and their quantities as input and specifies which consumable resources are available collectively on all of the machines in the LoadLeveler cluster. Any resource specified for this keyword that is not already listed in the SCHEDULE_BY_RESOURCES keyword is ignored.

Figure 85 shows an example of consumable resources defined in the configuration file.

```
#
# Consumable Resources
#
SCHEDULE_BY_RESOURCES = ConsumableCpus Donuts Balloons FloatingLicenseX
FLOATING_RESOURCES = Balloons(2) FloatingLicenseX(5)
```

Figure 85. Consumable resources defined in a LoadL configuration file

5.1.1.2 New keywords in the administration file

There are two new keywords in the LoadLeveler administration file for consumable resources: resources and default_resources.

The resources keyword belongs to the machine stanza. It has a list of resource names and their quantities as input and specifies the quantities of the resources initially available in the machine. There are pre-defined resource names for resource keywords: ConsumableVirtualMemory, ConsumableMemory, and ConsumableCPUs. Figure 86 shows an example of consumable resources defined for a machine.

```
mach4: type = machine
       adapter_stanzas = k10n01_en0 k10n01_css
       machine_mode = interactive
       resources = ConsumableCpus(2) Donuts(2) licenseB(3)
```

Figure 86. Consumable resources defined in a machine stanza

The default_resources keyword is for the class stanza. It has a list of resource names and their quantities as input, and specifies the quantities of the resources required by each task running in this job class. Figure 87 shows an example of consumable resources defined for a class.

```
picnic:  type = class
        priority = 0
        cpu_limit = unlimited
        job_cpu_limit = unlimited
        max_processors = -1
        default_resources = Donuts(2) Balloons(2) ConsumableCpus(2)
```

Figure 87. Consumable resources defined in class stanza

5.1.1.3 New Keywords in the job command file

The resources keyword in the job command file has a list of resource names and their quantities as input and specifies the quantities of the resources required by each task in this job. Figure 88 shows an example of a job command file that requires consumable resources.

```
#!/usr/bin/ksh
# @ job_type = parallel
# @ initialdir = /u/symoon
# @ executable = $(initialdir)/mpitest
# @ input = /dev/null
# @ output = $(executable).out
# @ error = $(executable).err
# @ class = symoon_class
# @ task_geometry={(5,2) (1,3) (4,6,0)}
# @ network.MPI = css0,shared,US
# @ resources = ConsumableMemory(100 mb) Balloons(1)
# @ queue
```

Figure 88. Consumable resources defined in a LoadL job command file

5.1.2 Geometry specification

LoadLeveler allows users to group tasks of a parallel job to run together on the same node, possibly with a different number of tasks on different nodes.

5.1.2.1 New keyword in the job command file

The task_geometry is a new keyword in the job command file for custom geometry. It is used to describe how to assign specific tasks in the same group. In this example, the task_geometry keyword groups seven tasks to run on three nodes:

```
# @ task_geometry = {(5,2)(1,3)(4,6,0)}
```

Each number in the example above represents a task ID in a parallel job. Each set of parenthesis contains the task IDs assigned to one node. Although the task_geometry keyword allows for a great deal of flexibility in how tasks are grouped, a user cannot specify the particular nodes that these groups run on; the scheduler will decide which nodes will run the specified groupings.

Figure 89 shows how the tasks are allocated by the task_geometry. It is easy to notice that the figure shows only one of six possible cases of permutation with three different groups.

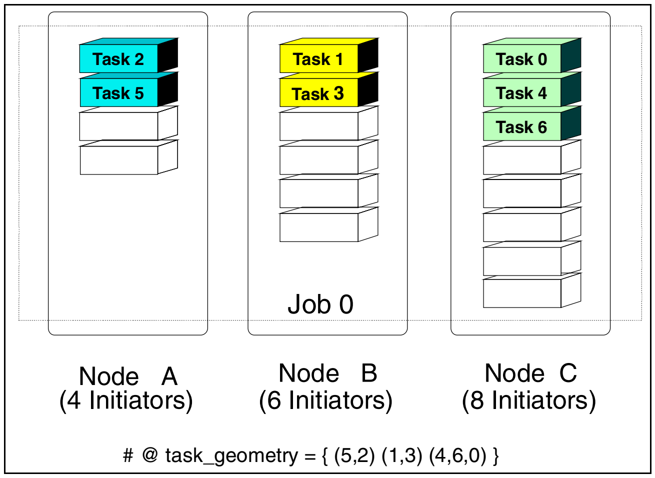

Figure 89. Tasks assigned by geometry specification

5.1.3 Blocking

Tasks can be allocated to nodes in groups (blocks) of the specified number (blocking factor). Blocking specifies that tasks be assigned to nodes in multiples of the blocking factor.

5.1.3.1 New keyword in the job command file

The blocking is a new keyword in the job command file. It has a positive integer as input. The blocking keyword must be specified along with the total_tasks keyword. The syntax is as follows:

```
# @ blocking = 4

# @ total_tasks = 17
```

This specifies that a job's tasks will be assigned in blocks of four tasks. LoadLeveler assigns tasks in blocks first and then those in the remainder.

Figure 90 illustrates how a blocking factor of four would work with 17 tasks.

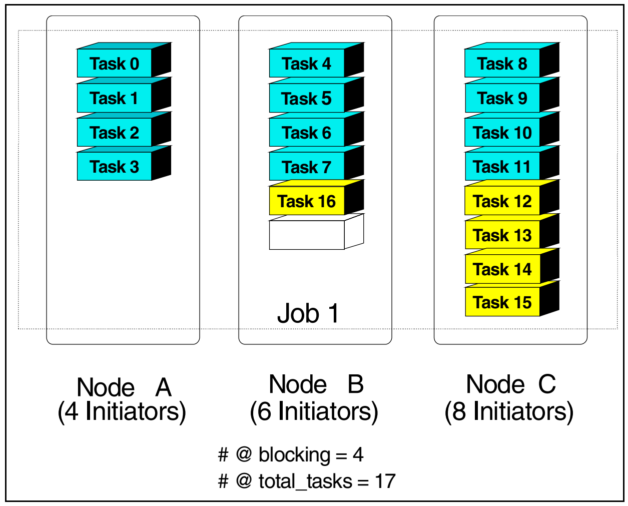

Figure 90. Tasks assigned by blocking

If blocking is set to 1, as a special case, LoadLeveler will spread the tasks over as many nodes as possible. Figure 91 shows an example how tasks are allocated with blocking=1.

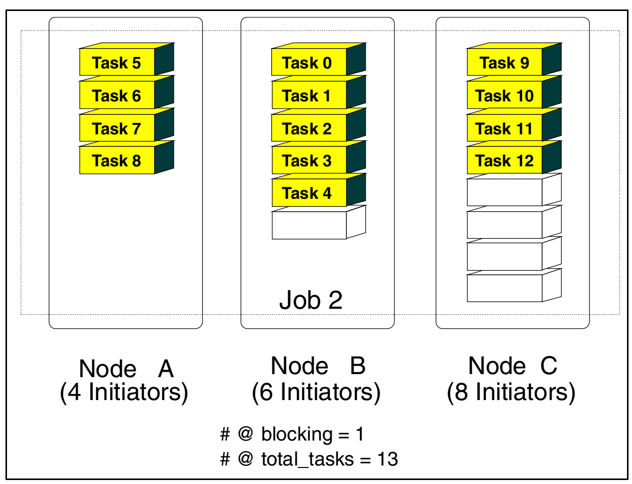

Figure 91. Tasks assigned by blocking=1

The blocking factor can also be a string, “unlimited”. Then, tasks are assigned to the each node until it runs out of initiators, at which time tasks will be assigned to the node that is next in the order of priority. Unlike with other task assignment keywords, the assignment function prioritizes nodes primarily by how many initiators each node has available, and secondly on their MACHPRIO expressions (beginning with the highest MACHPRIO value). Unlimited blocking is the only means of allocating tasks to nodes that does not prioritize machines primarily by MACHPRIO expression. This method allows a user to allocate tasks among as few nodes as possible. Figure 92 is an example for unlimited blocking assignment.

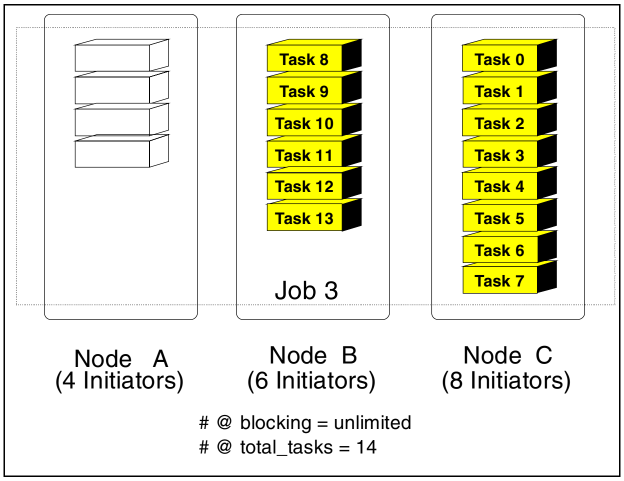

Figure 92. Tasks assigned by unlimited blocking

5.2 API changes

The Job Control API has been renamed in this release. This API is now called the Workload Management API and adds a new subroutine, ll_control. The ll_control subroutine provides most of the functions that are available through the standalone commands: `llctl`, `llfavorjob`, `llfavoruser`, `llhold`, and `llprio`. For example, a user can execute a standalone command, such as ‘`llctl start`’, by calling the ll_control subroutine with a control operation LL_CONTROL_START.

For complete description of the ll_control subroutine, refer to *IBM LoadLeveler for AIX: Using and Administering*, SA22-7311.

5.3 Performance and Limit Enhancements

There have been a lot of performance enhancements in this release of LoadLeveler. The following are important performance enhancements:

- Job scheduling time was reduced.
- LoadLeveler daemons are now running multi-threaded.
- Multiple processor capability was improved.
- Command response time has been improved.

LoadLeveler 2.2 can now handle up to 16 user-space tasks per node and 4096 tasks per single job with the SP Switch2 while supporting up to four tasks per node and 1024 tasks per job with the SP Switch as in LoadLeveler 2.1. Table 8 summarizes current limits in maximum number of parallel tasks.

Table 8. Limits of maximum number of tasks

Switch Type	SP Switch2		SP Switch	
Protocol	**US**	**IP**	**US**	**IP**
Maximum tasks per node	16	N/A	4	N/A
Maximum tasks per job	4096	2048	1024	2048

Note

In fact, LoadLeveler does not impose any architectural limits on the number of parallel tasks per node. There is no limit to the number of IP tasks that can run on a node. The user space tasks are limited by the number of adapter windows available for the switch adapter and the number of adapter windows each task requires (for example, a task using just MPI needs one window, a task using MPI and LAPI requires two windows). The CSS component of PSSP is the one that limits the number of adapter windows. There are four windows available with the SP Switch and sixteen windows with the SP Switch2 (See 2.4.9, “Data flow control” on page 101).

5.4 LoadLeveler TaskGuide

In order to provide users and administrators with more simple and powerful job scheduling tools, GUI implementation is quite effective and important. To provide more enhanced GUI support for the LoadLeveler administrator, the LoadLeveler TaskGuide has been added to this release of LoadLeveler.

The LoadLeveler TaskGuide provides a sequence of GUI panels that a LoadLeveler administrator steps through to accomplish LoadLeveler tasks regarding modification of LoadLeveler Configuration values related to scheduling. The TaskGuide format provides the same look and feel as other AIX/PSSP components; so, LoadLeveler administrators can use it very easily.

5.4.1 Installation of the LoadL TaskGuide

Installation of LoadLeveler TaskGuide is very simple. It can be done by installing the LoadL.tguides fileset. To install and run LoadLeveler TaskGuide, the Java runtime module, Java.rte.bin 1.1.6 or above, is also required.

5.4.2 Starting the LoadL TaskGuide

The LoadLeveler TaskGuide is started by selecting the **Admin** -> **Config Tasks** menu from the Machines sub-window of the xloadl window, as shown in Figure 93 on page 173, or by executing the invoking script lltg in the command line.

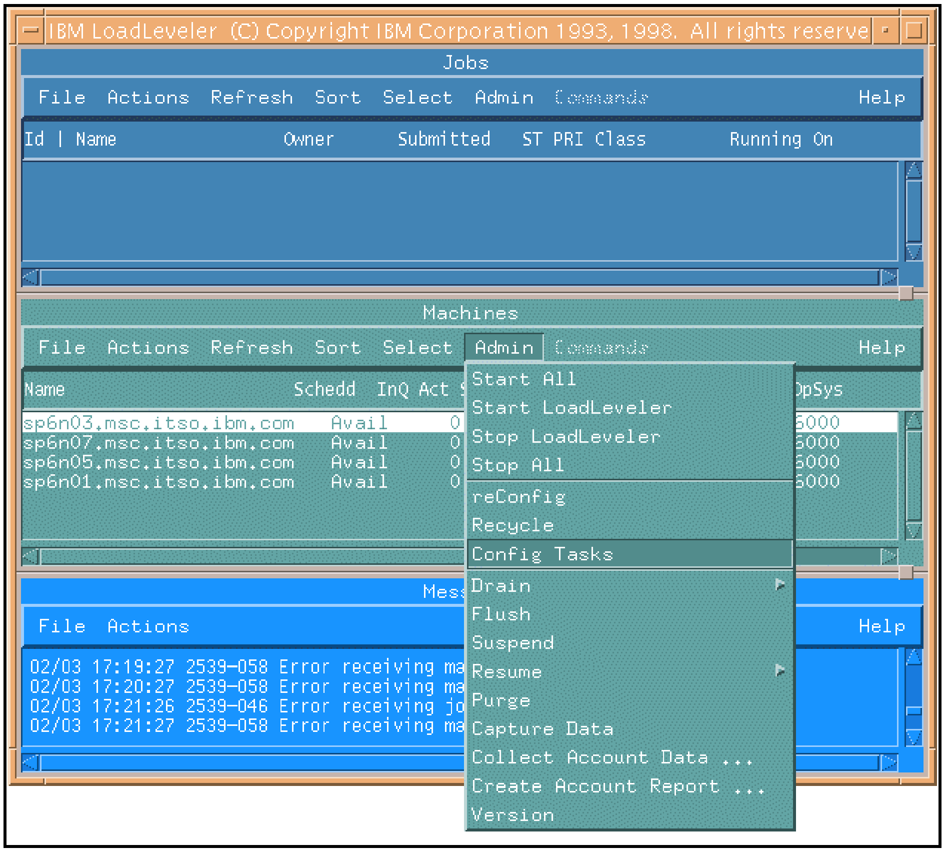

Figure 93. Starting LoadLeveler TaskGuide from xloadl window

Figure 94 shows the starting window of LoadLeveler TaskGuide, from which the administrator can select and execute a configuration task for scheduling.

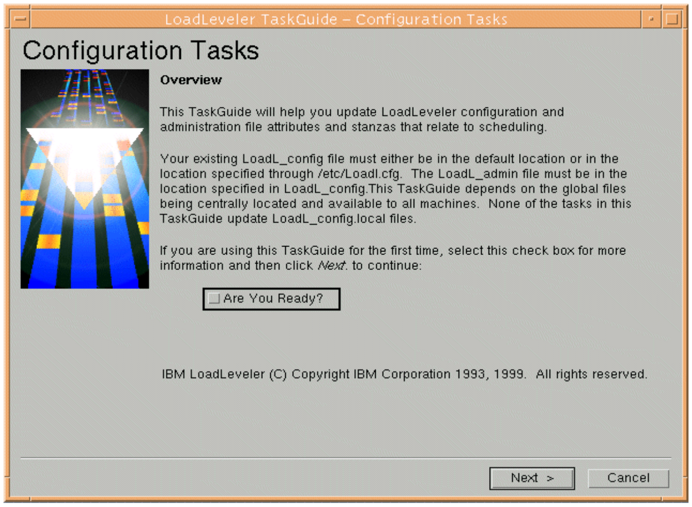

Figure 94. Starting window of LoadLeveler TaskGuide

5.4.3 Using the LoadL TaskGuide

Five configuration tasks are now available in the LoadLeveler TaskGuide, shown in Figure 95, to update stanzas or attributes in LoadLeveler configuration and/or administration files.

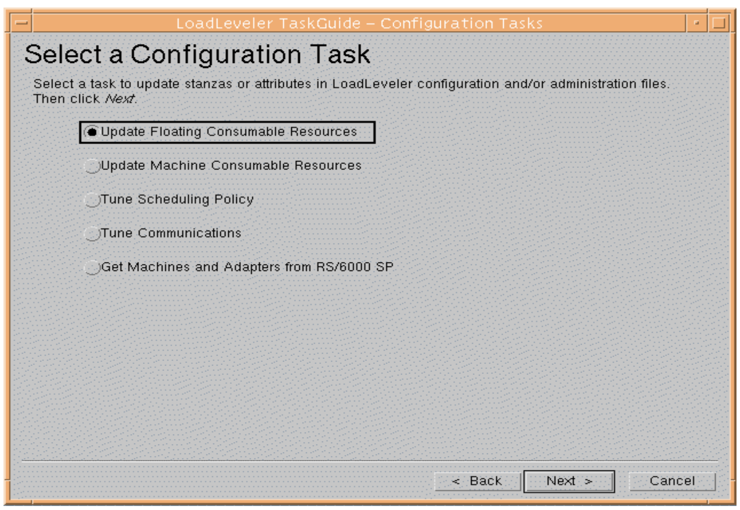

Figure 95. Task Selection Panel in LoadLeveler TaskGuide

The following is an explanation of each one of the options available at this level.

Update floating consumable resources
This menu allows the following tasks for floating consumable resources configuration for the LoadLeveler cluster:

- View floating resources and counts
- Add new floating resources and control whether they go in the schedule_by_resources list
- Edit floating resources count
- Delete floating resources

Update machine consumable resources
This menu allows the following tasks for machine consumable resources configuration for the LoadLeveler cluster:

- View machines, their resources, and count on that machine
- Add new machine resources and control whether they go in the schedule_by_resources count
- Edit machine resources count
- Delete machine resources

Tune scheduling policy
This menu sets the following scheduling parameters in the configuration file:

- Scheduler type
- Length of negotiation cycle
- Action on reject
- Schedule by resources
- System priority
- Machine priority

Tune communications
This menu changes the time interval of various LoadLeveler communication settings.

Get machines and adapters from SP
The administrator can now specify, easily and safely, the basic information about machine and adapter stanzas in the LoadL_admin file by using LoadLeveler TaskGuide. This menu executes the following tasks:

- Get machine and adapter information for a partition
- Present it for review and update
- Merge it into the LoadL_admin file

Figure 96 shows an example of machine and adapter information merged to LoadL_admin file by TaskGuide.

```
sp6n01: type = machine
        adapter_stanzas = sp6sw01 sp6n01
        spacct_excluse_enable = false
        alias = sp6sw01

sp6sw01: type = adapter
        adapter_name = css0
        network_type = switch
        interface_address = 192.168.16.1
        interface_name = sp6sw01
        switch_node_number = 0
        css_type = SP_Switch_MX_Adapter

sp6n01: type = adapter
        adapter_name = en0
        network_type = ethernet
        interface_address = 192.168.6.1
        interface_name = sp6n01
```

Figure 96. Information merged to LoadL_admin file by LoadLeveler TaskGuide

5.5 Enhanced Parallel Environment features in LoadLeveler

In general, parallel programming incurs many types of overheads that are not directly related to development and execution of user programs. Such overheads can prevent users from fully utilizing the systems and their own resources. IBM Parallel Environment for AIX (PE) deeply depends on LoadLeveler's resources management and scheduling capability to run parallel jobs; so, parallel environment features in LoadLeveler are very important. The following two sections describe some of the enhanced parallel environment features of LoadLeveler.

5.5.1 Process tracking

When a job terminates, its orphaned processes may continue to consume or hold resources, thereby degrading system performance or causing jobs to hang or fail. Process tracking is an option to track all the processes created by a job and allows LoadLeveler to cancel any processes (throughout the entire cluster) left behind when a job terminates.

5.5.1.1 New keywords in the configuration file

There are two new keywords used in specifying process tracking in the configuration file.

PROCESS_TRACKING

If set to TRUE, this keyword ensures that when a job is terminated, no processes created by the job can continue running. This keyword is in the LoadLeveler global configuration file. By default, PROCESS_TRACKING is set to FALSE.

PROCESS_TRACKING_EXTENSION

This keyword specifies the location of the directory that contains the kernel extension binary, LoadL_pt_ke, for process tracking. This keyword is for the local or global configuration files. If the PROCESS_TRACKING_EXTENSION is not supplied, then LoadLeveler will search the default directory, /u/loadl/bin.

5.5.2 Job Command File for interactive POE jobs

A LoadLeveler job command file, itself, can now be used for node allocation for interactive POE jobs. LoadLeveler job parameters can be specified in the job command file for interactive POE jobs, though not all keywords are applicable (for example, dependency, hold, and others). The following list shows some of LoadLeveler functionalities that can be used for interactive POE jobs:

- Task geometry
- Blocking factor
- Machine order
- Consumable resources
- Memory requirements
- Disk space requirements
- Machine architecture

A new POE option, the MP_LLFILE environment variable or -llfile flag, is used to specify the LoadLeveler job command file to be used for an interactive POE job. For example, the following example runs an interactive POE job *myprogram* with the conditions specified in a LoadLeveler job command file, /u/symoon/lljob.cmd:

```
poe myprogram -llfile /u/symoon/lljob.cmd
```

A LoadLeveler job command file can be used with or without a host list file. If a LoadLeveler job command file is specified in conjunction with a host list file, the task assignment keywords, such as node, task_per_node, total_tasks, and so forth, are ignored, and the specific nodes listed in the host list file will be requested from LoadLeveler.

5.6 Enhanced support for DCE security

LoadLeveler now fully supports DCE security features. Key features of DCE include the ability to authenticate users' identities, authorize users and programs to use LoadLeveler's services, and delegate user credentials to the starter process.

DCE maintains a registry of all DCE principals that have been authorized to log in to the DCE cell. In order for LoadLeveler daemons to login to DCE, DCE accounts must be set up, and DCE key files must be created for these daemons.

5.6.1 Configuring DCE security for LoadLeveler

When LoadLeveler is configured to exploit DCE security, most of its interactions with DCE are through the PSSP security services API. For this reason, PSSP should be configured first for DCE security services before configuring LoadLeveler for DCE.

For detailed information on steps used for configuring LoadLeveler for DCE, refer to *IBM LoadLeveler for AIX: Using and Administering*, SA22-7311.

The following sections discuss new keywords and commands of LoadLeveler configuration for DCE.

5.6.1.1 New keywords in the LoadLeveler configuration file

These new keywords should be specified in the LoadLeveler global configuration file:

DCE_ENABLEMENT = TRUE

DCE_ADMIN_GROUP = *LoadL-admin*

DCE_SERVICES_GROUP = *LoadL-services*

DCE_ENABLEMENT must be set to TRUE to activate the DCE security features of LoadLeveler.

DCE_ADMIN_GROUP defines a DCE group *LoadL-admin* for LoadLeveler administration. The *LoadL-admin* group should be populated with the DCE principals of users who are to be given LoadLeveler administrative privileges.

DCE_SERVICES_GROUP defines a DCE group *LoadL-services* for LoadLeveler services. The *LoadL-services* group should be populated with the DCE principals of all the LoadLeveler daemons that will be running in the current cluster.

5.6.1.2 New command for LoadLeveler configuration for DCE

The lldcegrpmaint is a new command to support LoadLeveler configuration for DCE. Usage of this command is as follows:

```
lldcegrpmaint config_pathname admin_pathname
```

where *config_pathname* is the path name of the LoadLeveler global configuration file, and *admin_pathname* is the path name of the LoadLeveler administration file. This command extracts the names of the DCE groups associated with the DCE_ADMIN_GROUP and DCE_SERVICES_GROUP keywords from the LoadLeveler configuration file. It will create these groups if they do not already exist. This command also adds the DCE principal of all the LoadLeveler daemons in the LoadLeveler cluster defined by the administration file to the group specified by the DCE_SERVICES-GROUP keyword.

Chapter 6. Parallel programming

Many new and enhanced features are included in IBM Parallel System Support Programs for AIX (PSSP) 3.2 and IBM Parallel Environment for AIX (PE) 3.1 to support parallel programming on the IBM RS/6000 SP.

This chapter highlights new enhancements in communications Low-level Application Programming Interface (LAPI), Message Passing Interface (MPI), Dynamic Probe Class Library (DPCL), and LightWeight Corefile (LWCF).

6.1 LAPI

This section briefly describes introductory concepts of LAPI and new enhancements, such as LAPI vectors. For more detailed and practical information, refer to *IBM Parallel System Support Programs for AIX: Administration Guide*, SA22-7348 and *Scientific Applications in RS/6000 SP Environments*, SG24-5611.

6.1.1 What is LAPI?

The Low-level Application Programming Interface (LAPI) is a non-standard interface designed to provide low latency, interrupt handling, and minimal CPU utilization in terms of communication over the SP Switch or SP Switch 2. Unlike in traditional MPI message passing (MPI-1), the LAPI semantic is unilateral, that is, one process initiates a LAPI operation, and the completion of the operation does not require any other process to take some complementary action. The LAPI library provides the functions PUT, GET and a general active message function that allows programmers to supply extensions by means of additions to the notification handlers. Those LAPI functions can either be invoked as C, C++, or Fortran calls, and corresponding header files are provided for these languages.

6.1.2 Why use LAPI?

There are many reasons why LAPI should be considered for a high-performance communication protocol over SP Switch or SP Switch 2:

- Performance - LAPI provides a small set of interfaces to write parallel programs requiring high-performance and reliable communication, especially for users for whom performance is more important than code portability.
- Flexibility - LAPI's one-sided communication model provides flexibility. Also, LAPI provides a more primitive interface (than either MPI or IP) to

the SP Switch or SP Switch 2, thus giving the programmer the choice of how much additional communication protocol needs to be added.

- Extendibility - LAPI supports programmer-defined handlers that are invoked when a message arrives. Programmers can customize LAPI to their specific environments.
- PSSP Strategy - There is also a strategic direction for design of communication software layers on top of the SP Switch and SP Switch 2. The strategic direction of LAPI, regarding PSSP itself, is to be the efficient transport layer of choice for all user space based subsystems and libraries over the SP Switch and SP Switch 2. While LAPI has many advantages, and may be suitable for developing all kinds of application protocols, the strategy does not suggest that all protocols will be moved to LAPI. IBM General Parallel File System for AIX (GPFS) is a good example to show how the LAPI protocol can be exploited for high performance communication and I/O (see Section 4.1.5.2, “GPFS LAPI exploitation” on page 151).

6.1.3 Where is LAPI?

The LAPI is packaged with PSSP, included in the ssp.css fileset. The IBM Parallel Environment for AIX product is also required to run LAPI applications. LAPI programs are started by the Parallel Operating Environment (POE), much the same way as MPI programs. In fact, LAPI and MPI calls can coexist within the same program or the same parallel job.

6.1.4 LAPI concepts

LAPI is a non-blocking and asynchronous communication protocol. The following sections are descriptions on important concepts of LAPI.

6.1.4.1 Origin and target

Origin denotes the task that initiates a LAPI operation (PUT, GET, or active message). Target denotes the other task whose address space is accessed.

6.1.4.2 Blocking and non-blocking calls

A blocking procedure is one that returns only when the operation is complete, and there are no restrictions on the reuse of user resources.

A non-blocking procedure is one that may return before the operation is complete and before the user is allowed to reuse all the resources specified in the call.

6.1.4.3 Completion of communication operation

A communication operation is considered to be complete, with respect to the buffer, when the buffer is reusable.

A PUT is complete with respect to the origin buffer when the data has been copied out of the buffer at the origin and may be overwritten. A GET is complete with respect to the origin buffer when that origin buffer holds the new data that was obtained by GET.

Communication behaviors

Two communication behaviors support two different definitions of "completion":

- In *standard* behavior, a communication operation is defined as complete at the origin task when it is complete with respect to the *origin* buffer; it is complete at the target task when it is complete with respect to the *target* buffer.
- In *synchronous* behavior, a communication operation is defined as complete at the *origin* task when it is complete with respect to both the origin buffer and target buffer. It is complete at the *target* task when it is complete with respect to the target buffer.

LAPI defines both standard and synchronous behaviors for PUT operations, but it defines only synchronous behavior for GET operations. Note that these behaviors are programmed through the origin counter, target counter, and completion counter.

6.1.4.4 Signaling completion of communication operations

LAPI uses counters to signal the events associated with non-blocking communication calls. The specified counter is incremented whenever the event associated with the counter occurs.

The LAPI library updates the user specified counters when a particular event with which the counter was associated has occurred. The user can check on the status of a particular event by polling on the value of the appropriate counter. If a user wants to wait on a particular event rather than check if it has occurred, LAPI provides an interface to wait on the counter until the counter reaches the specified value. PSSP 3.2 introduces a new call to help minimize the polling for counters and reduce the CPU usage. Instead of using the LAPI_Waitcntr function, which is a polling call, the user can use the new LAPI_Nopoll_wait function, which will sleep until counter value is reached.

6.1.4.5 Message ordering and atomicity

Two LAPI operations that have the same origin task are considered to be ordered with respect to the origin if one of the operations starts after the other has completed at the origin task. Similarly, two LAPI operations that have the same target task are considered to be ordered with respect to the target if one of the operations starts after the other has completed at the target task. If two operations are not ordered, they are considered concurrent.

6.1.4.6 Error handling

If an error occurs during a communications operation, the error may be signaled at the origin of operation, or at the target, or both.

6.1.4.7 Progress

All LAPI operations are unilateral by default and can complete successfully or fail, independent of the actions of other tasks.

6.1.5 LAPI architecture

In order to understand the LAPI architecture, it is helpful to take an overview of the SP Communication subsystem (CSS) stack shown in Figure 52 on page 90. More specifically, Figure 97 shows the LAPI protocol stack.

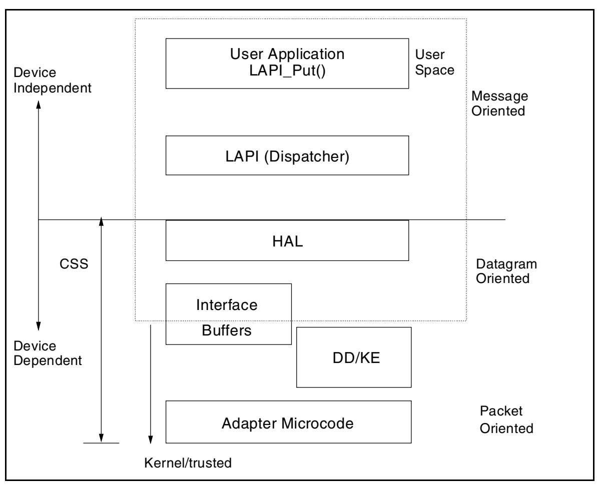

Figure 97. LAPI protocol stack

The LAPI layer interacts with the SP Switch (2) through the hardware abstraction layer (HAL) in which only the minimal hardware functionality is encapsulated. HAL is responsible for sending and receiving packets through the SP Switch (2) and also ensures that a task can only communicate with another task of the same parallel job by authentication.

The LAPI dispatcher in the LAPI layer deals with the sending of messages, arrival of messages, and the invocation of handlers. The LAPI dispatcher can be executed by the application thread when it makes a LAPI function call, or they can be executed by one of the threads that LAPI creates at initialization time.

The LAPI functions provide user-level interfaces to communicate over the SP Switch (2). The LAPI functions are broadly divided into three parts: Active message, defined set of functions, and control functions. These LAPI functions can provide a transparent communication capability, regardless of what kind of switch hardware is being used under the LAPI layer.

6.1.5.1 Active message

A basic *active message* is an infrastructure that allows programmers to install a set of handlers that are invoked and executed in the address space of a target process on behalf of the process originating the active message.

6.1.5.2 Defined set of functions

A set of defined functions are complete enough to satisfy the requirements of most programmers. These defined functions make the LAPI more usable and, at the same time, lend themselves to efficient implementation because their syntax and semantics are known.

Fundamentally, the defined set of functions for the LAPI provides a Remote Memory Copy (RMC) interface. The basic data transfer operations are memory to memory copy operations that transfer data from one virtual address space to another virtual address space. These operations are unilateral. That is, one process initiates an operation, and the completion of the operation does not require any other process to take some complementary action. The initiating process specifies the virtual address of both the source and destination of the data.

6.1.5.3 Control functions

The set of control functions is for the initialization and eventual orderly shutdown of the LAPI layer.

6.1.6 The active message infrastructure

The underlying infrastructure that was selected for LAPI is referred to as the active message. The active message includes the address of a user-specified handler. When the active message arrives at the target process, the specified handler is invoked and executes in the address space of the target process. The active message optionally may also bring with it the user header and data from the originating process. The active message operations are unilateral in the sense that the target process does not have to take explicit action for the active message to complete.

The active message brings with it data from the originating process. The architecture requires that the handler be written as two separate routines:

- A header handler function. This function is specified in the active message call. It is called when the message first arrives at the target process and provides the LAPI dispatcher (the part of LAPI that deals with the arrival of messages and invocation of handlers) with the address of where to assemble packets of arriving data, the address of the optional completion handler, and a pointer to the parameter that is to be passed to the completion handler.
- A completion handler that is called after the whole message has been received. The completion handler is used by the user to incorporate the message into the ongoing computation or message processing. Figure 98 illustrates the flow of data and control in a LAPI active message.

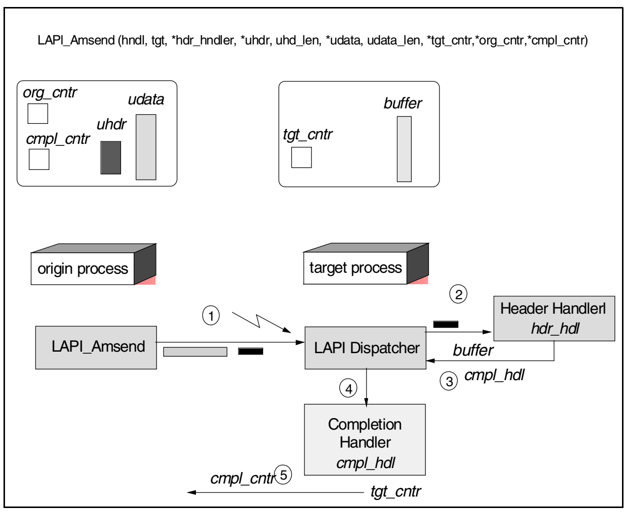

Figure 98. Active message flow

Perform the following steps:

1. A process on the origin makes a LAPI_Amsend call. The call initiates a transfer of the header *uhdr* and data *udata* at the origin process to the target process specified in the LAPI active message call. As soon as the

user is allowed to reuse *uhdr* and *udata*, an indication is provided via *org_cntr* at the origin process. When the header and data arrive at the target, an interrupt is generated that results in the invocation of the LAPI dispatcher.

2. The LAPI dispatcher identifies the incoming message as a new message and calls the *hdr_hndlr* specified by the user in the LAPI active message call.
3. The handler returns a buffer pointer where the incoming data is to be copied. The header handler also provides LAPI with an indication of the completion handler that must be executed when the entire message is copied into the target buffer specified by the header handler. The LAPI library moves the data (which may be transferred as multiple network packets) into the specified *buffer*.
4. On completion of the data transfer, the user-specified completion routine is invoked.
5. After the completion routine finishes execution, the *tgt_cntr* at the target process and *cmpl_cntr* at the origin process are updated indicating that the LAPI active message call is now complete.

6.1.7 LAPI functions

The following are some typical LAPI functions.

6.1.7.1 Active message

The active message function (LAPI_Amsend) is a non-blocking call that causes the specified active message handler to be invoked and executed in the address space of the target process.

6.1.7.2 Data transfer

LAPI data transfer operations support both pull and push. The LAPI_Get operation copies data from the address space of the target process into the address space of the origin process. The LAPI_Put operation copies data into the address space of the target process from the address space of the origin process.

6.1.7.3 Synchronizing

The LAPI_Rmw function is used to synchronize two independent operations, such as two processes sharing a common data structure. The operation is performed at the target process and is atomic. LAPI_Rmw provides four different read/modify/write operations:

- SWAP

- COMPARE_AND_SWAP
- FETCH_AND_ADD
- FETCH_AND_OR

6.1.7.4 Completion checking

The following counter functions provide the means for a process to manage the completion state of the LAPI operations:

- **LAPI_Waitcntr** - Wait on a counter to reach a specified value and return when the counter is equal to or greater than that value (blocking, polling).
- **LAPI_Nopoll_wait** - Wait on a counter to reach a specified value and return when the counter is equal or greater than that value (blocking, non-polling).
- **LAPI_Getcntr** - Get the current value of a specified counter (non-blocking).
- **LAPI_Setcntr** - Set the counter to a specified value.

6.1.7.5 Ordering

The LAPI_Fence and LAPI_Gfence operations provide a means to enforce the order of execution of LAPI functions. LAPI functions initiated prior to these fencing operations are guaranteed to complete before all LAPI functions initiated after the fencing functions. LAPI_Fence is a local operation that is used to guarantee that all LAPI operations initiated by the local process and that the same process thread are complete. LAPI_Gfence is a collective operation involving all processes in the parallel program.

6.1.8 LAPI enhancements: LAPI vector functions

PSSP 3.2 introduces the LAPI vector functions to support improved performance and ease of use of transporting non-contiguous data between tasks using LAPI. Without LAPI vector functions, non-contiguous data can be transferred either by using a series of data transfer functions, one for each contiguous part of data, resulting in pipelining overhead, or by copying the various parts of data into a contiguous buffer before using LAPI to transfer the data, resulting in copy overhead. LAPI vector avoids user overhead of either the pipeline cost or the copy cost. To simplify the description of the non-contiguous data transfer functions, the following data structure is defined:

```
typedef struct {
  lapi_vectype_t vec_type; /* operation code */
   uint num_vecs; /* number of vectors */
```

```
   void **info; /* vector of information */
   uint *len; /* vector of lengths */
} lapi_vec_t;
```

where ***vec_type***, ***num_vecs***, ***info***, and ***len*** are variables to be replaced by the respective values.

To perform the vector data transfer, three new functions are added: LAPI_Putv, LAPI_Getv, and LAPI_Amsendv.

If vec_type is set to be LAPI_GEN_IOVECTOR, it is called a general I/O vector transfer. Figure 99 shows an example of general I/O vector transfer, where four data strips with different lengths are transferred from the origin task to the target task by one vector transfer operation.

.

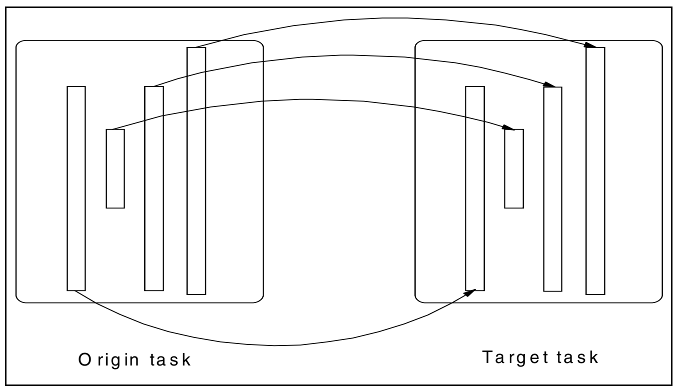

Figure 99. General I/O vector transfer

If vec_type is set to be LAPI_GEN_STRIDED_XFER, it is called a general strided transfer.

Figure 100 shows an example of general strided vector transfer, where three strided blocks of data are transferred from origin task to the target task by one vector transfer operation.

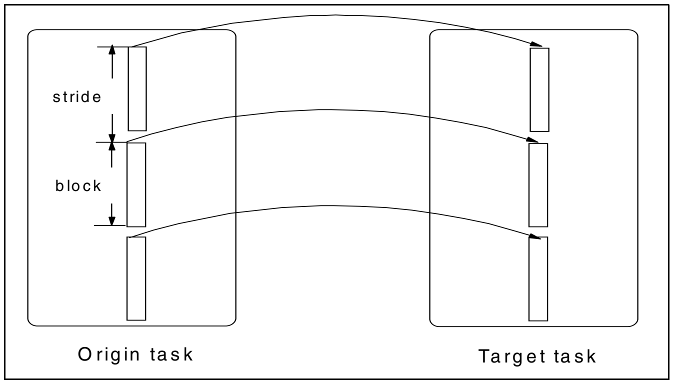

Figure 100. Strided vector transfer

6.1.9 Executing LAPI Programs

In order to compile and run a LAPI program, PE should be installed in the SP system. In fact, the POE component of PE is used to support LAPI programs.

6.1.9.1 Compiling LAPI programs

Use the commands, `mpcc_r`, `mpCC_r`, or `mpxlf_r`, which support programs using the threaded LAPI library, to compile LAPI programs that are written in C, C++, or Fortran, respectively. The `mpcc_r`, `mpCC_r`, and `mpxlf_r` commands not only compile the program, but also link in the POE Partition Manager and PSSP Communication Subsystem (CSS) interfaces. The CSS libraries will be dynamically linked with the executable program at run time.

6.1.9.2 Running LAPI programs

Before running a LAPI program, a user needs to set up his or her execution environment. There are a number of POE environment variables discussed throughout *IBM Parallel Environment for AIX: Operation and Use, Volume 1*, SA22-7425, and summarized in the “POE Environment Variables and Command-Line Flags” appendix of that book.

LAPI programs must set the environment variable MP_EUILIB (or the command-line flag -euilib) to us (us is the User Space communication subsystem). LAPI supports only User Space communication used with the SP Switch. LAPI does not support the IP communication subsystem.

6.2 MPI enhancements

This section discusses some important changes in PE, especially MPI enhancements. It covers the shared memory MPI message passing for intra-node communication, MPI one sided communication and MPI-IO.

6.2.1 Shared memory MPI

The shared memory MPI message passing provides superior performance on Symmetric Multiprocessor (SMP) nodes; that is, MPI programs may benefit from using shared memory to send messages between tasks of a parallel job, running on the same physical node, without any modifications of original programs. MPI programs can be run in shared memory mode over either User Space (US) or IP network protocols. Figure 101 shows the difference between MPI tasks running in shared memory mode and those running over an SP Switch.

.

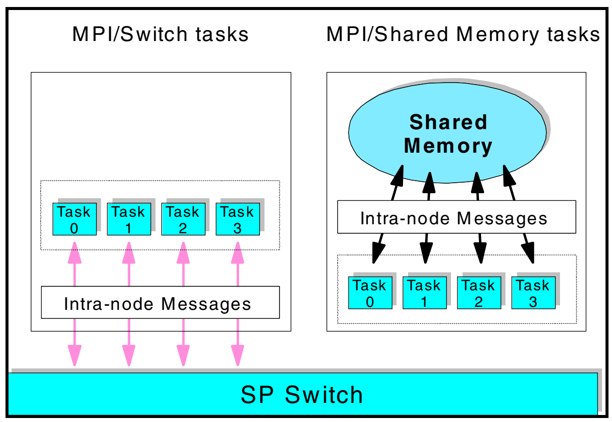

Figure 101. MPI/Switch vs. MPI/Shared memory

PE 3.1 also supports a mixed-mode MPI; that is, it provides MPI with shared memory access between tasks of a parallel job on the same node while still using normal message passing over the network between tasks of the parallel job on different nodes.

Figure 102 shows an example of mixed-mode MPI tasks running on two nodes.

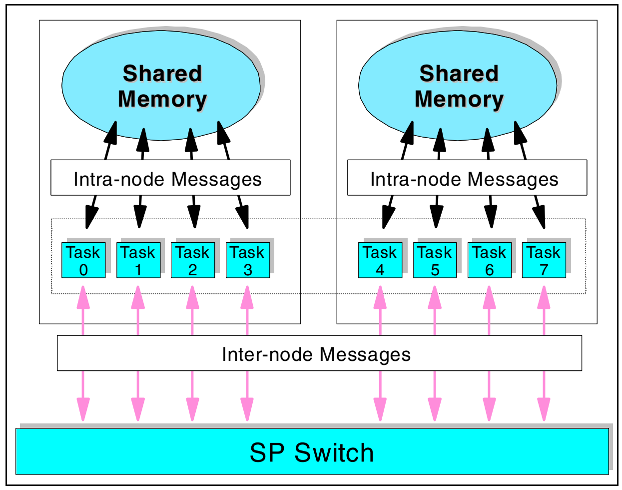

Figure 102. MPI tasks running in mixed mode

To support the shared memory MPI, PE 3.1 introduces a new POE environment variable, MP_SHARED_MEMORY, which specifies the shared memory option. The default setting is no. Setting this variable to yes directs MPI to use a shared-memory protocol between tasks of a job executing on the same node. MPI is aware of which tasks are on the same node and uses this in algorithms for collective communication and topology construction.

6.2.2 MPI-2 support

The MPI is the result of standardization effort for an expressive and flexible message passing API for parallel computing that can be implemented on almost any platform so that an MPI program could be compiled to run with a compliant MPI implementation.

The first MPI standard, MPI 1.0 (MPI-1), which focused mainly on point-to-point and collective communications, was announced by the Message Passing Interface Forum (MPIF) in May, 1994. With minor error corrections and clarifications of the MPI 1.0, the MPI 1.1 standard was released in June, 1995. IBM Parallel Environment for AIX (PE) provided a full implementation of MPI 1.1 standard since August, 1995.

After much discussion, the MPI 2.0 standard (MPI-2) was approved in July, 1997. It adds completely new types of functionality to MPI, including dynamic processes, one-sided communication, parallel I/O, among others. The MPI-2 standard also includes an MPI 1.2 chapter that provides clarifications of MPI 1.1. In the previous release, PE 2.4 contained some MPI-2 functions:

- MPI-IO file read/write with explicit offsets, both collective and non-collective versions
- MPI_Datatype decoding functions

Though complete implementation of MPI-2 standard is still going on, PE 3.1 now provides full support for two of the most important features of the MPI-2 standard:

- **MPI One-Sided Communication**
- **MPI- IO**

The following two sections describe the details on these functionalities.

6.2.3 MPI one-sided communication

In traditional MPI (MPI-1), communication between tasks of a job is realized by explicitly specifying the required parameters both in sender and receiver sides. But, the Remote Memory Access (RMA) functionality defined in MPI-2 extends the communication mechanisms of MPI by allowing one process to specify all communication parameters, both for the sending side and for the receiving side. The MPI one-sided communication approximates a shared memory model with weak coherency and explicit synchronization calls.

MPI one-sided communication provides three data transfer calls, MPI_PUT (remote write), MPI_GET (remote_read), and MPI_ACCUMULATE (remote update). It also has several synchronization calls to support different synchronization styles, as well as initialization calls, to create and release a distinct communication window.

Multiple MPI_PUT, MPI_GET, MPI_ACCUMULATE operations that occur on a task within an access epoch and have a common target will be coalesced by default. The data transfer will occur when the space available for gathering the operations fills or at the next synchronization. The hint IBM_win_cache allows the user to control the amount of space available for this. Setting IBM_win_cache to zero will shut off this feature.

In MPI one-sided communication, the process that initiates the data transfer is denoted by origin, and the process in which the memory is accessed by the origin is denoted by target. Thus, in a put operation, source=origin and destination=target; in a get operation, source=target and destination=origin (see Figure 104 on page 195 and Figure 105 on page 196).

6.2.3.1 Initialization

To process the supposed communications between participating processes, RMA requires a communication window that is created in each process' memory and is made accessible by remote processes in the group. Processes in the group can perform RMA operations only through the windows associated with that group and only after the window has been created successfully in each node. MPI provides two initialization calls to create and release the window, MPI_WIN_CREATE and MPI_WIN_FREE.

MPI_WIN_CREATE

This is a collective call executed by all tasks in a distinct communication group. It returns a window object that can be used by these tasks to perform RMA operations. Each task specifies a window of existing memory that it exposes to RMA accesses by the tasks in the group; so, any task can do RMA on the window at any task in the same group, but MPI prevents alteration of memory outside the window.

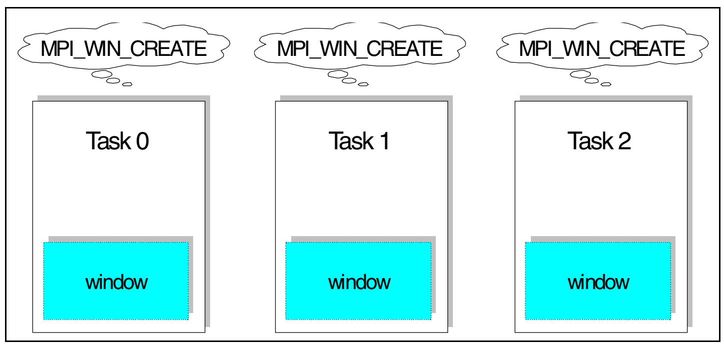

Figure 103. MPI window

The windows may have different sizes and locations at each task, and a task can define more than one window, doing more than one MPI_WIN_CREATE, while the operations on each window are independent of others.

MPI_WIN_FREE

This call frees the window object specified by the MPI_WIN_CREATE call. This is a collective call executed by all processes in the group associated with that window. When the call returns, the window memory can be freed, and processes can not do RMA operations through that window any more.

6.2.3.2 Remote Data Transfer

MPI one-sided communication supports three RMA communication calls, MPI_PUT, MPI_GET and MPI_ACCUMULATE. These operations are non-blocking, that is, the call initiates the transfer, but the transfer may continue after the call returns. The transfer is completed, both at the origin and at the target, when a subsequent synchronization call is issued by the caller on the involved window object.

MPI_PUT

MPI_PUT transfers data from any data buffer at the origin task to a window at the target task as shown in Figure 104. It is like an MPI_SEND to a matching MPI_RECV, but all arguments are supplied by the origin task. The source and destination buffers are defined using MPI datatypes, while the datatype supplied by the origin task is interpreted at the target task as if it were declared there.

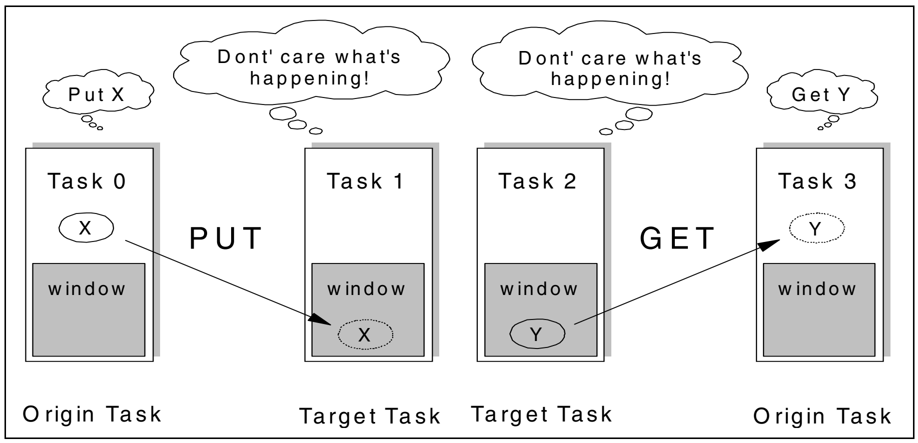

Figure 104. MPI_PUT and MPI_GET

MPI_GET

MPI_GET transfers data from a window at the target task to any buffer at the origin task. It is similar to MPI_PUT, except that the direction of data transfer is reversed as shown in Figure 104.

MPI_ACCUMULATE

MPI_ACCUMULATE accumulates, according to the specified MPI reduction operation, the contents of the origin buffer to the specified buffer at the target window as shown in Figure 105 on page 196. Any of the predefined operations for MPI_REDUCE can be used, but user-defined functions cannot be used. It is like MPI_PUT, except that data is combined into the target area instead of overwriting it. It is often useful in a put operation to combine the data moved to the target task with the data that resides at that task rather than replacing the data there. This will allow, for example, the accumulation of a sum by having all involved processes add their contribution to the sum variable in the memory of one task.

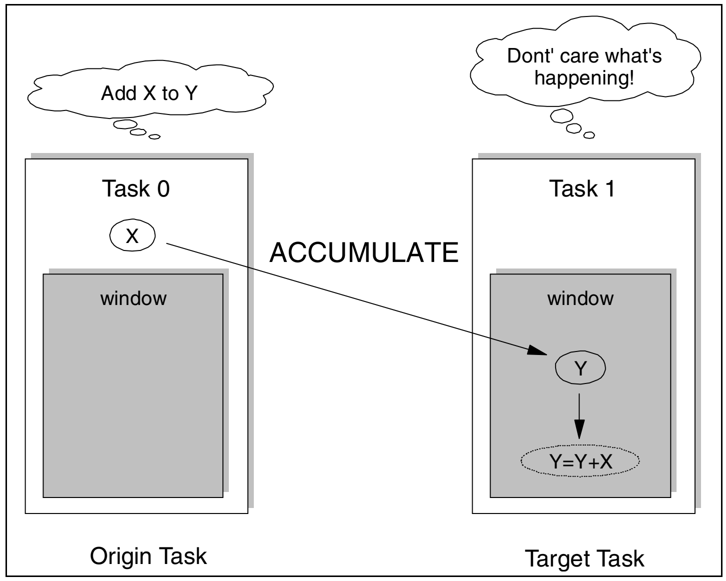

Figure 105. MPI_ACCUMULATE

6.2.3.3 Synchronization

MPI one-sided communication provides three synchronization mechanisms:

MPI_WIN_FENCE

This is a collective synchronization call that synchronizes RMA calls on the specified window. All RMA operations on the window originating at a given task and started before the fence call will complete at that task before the fence call returns. They will be completed at their target before the fence call returns at the target.

RMA operations on the window started by a task after the fence call returns will access their target window only after MPI_WIN_FENCE has been called by the target process.

A fence call usually entails a barrier synchronization: A task completes a call to MPI_WIN_FENCE only after all other tasks in the group entered their matching call.

MPI_WIN_{START; COMPLETE/POST; WAIT}

The four functions MPI_WIN_START, MPI_WIN_COMPLETE, MPI_WIN_POST and MPI_WIN_WAIT can be used to synchronize subgroups of communicating tasks:

- MPI_WIN_START declares a subgroup the task will do RMA upon and starts an RMA access epoch for the specified window.
- MPI_WIN_COMPLETE ends the RMA access epoch on the specified window started by a call to MPI_WIN_START. It enforces completion of preceding RMA calls at the origin but not at the target. A put or accumulate call may not have completed at the target when it has completed at the origin.
- MPI_WIN_POST opens an exposure epoch of the window for RMA from a specific subgroup.
- MPI_WIN_WAIT ends the RMA exposure epoch on the specified window started by a call to MPI_WIN_POST.

MPI_WIN_LOCK / MPI_WIN_UNLOCK

Shared and exclusive locks are provided by these two functions. A task may lock or unlock any window in the group:

- MPI_WIN_LOCK starts an RMA access epoch. Only the window at the specified task can be accessed by RMA operations on the window during that epoch.
- MPI_WIN_UNLOCK completes an RMA access epoch started by a call to MPI_WIN_LOCK. RMA operations issued during this period will have completed both at the origin and at the target when the call returns.

6.2.4 MPI-IO

POSIX provides a model of a widely portable file system, but the portability and optimization needed for parallel I/O cannot be achieved with the POSIX interface. The significant optimizations required for efficiency can only be implemented if the parallel I/O system provides a high-level interface supporting partitioning of file data among processes and a collective interface supporting complete transfers of global data structures between process memories and files.

MPI-IO is the I/O component of MPI-2 and provides a set of interfaces that are aimed at performing portable and efficient parallel I/O. Many would describe MPI-IO as the largest added value in MPI-2.

6.2.4.1 Definitions in MPI-IO

The following are the definitions in MPI-IO.

MPI File

An MPI file is an ordered collection of typed data items. MPI supports random or sequential access to any integral set of these items. A file is opened collectively by a group of processes. All collective I/O calls on a file are collective over this group.

Displacement

A file displacement is an absolute byte position relative to the beginning of a file. The displacement defines the location where a *view* begins.

Etype

An etype (elementary datatype) is the unit of data access and positioning. It can be any MPI predefined or derived datatype. Data access is performed in etype units, reading or writing whole data items of type etype. Offsets are expressed as a count of etypes; file pointers point to the beginning of etypes.

Filetype

A filetype is the basis for partitioning a file among processes and defines a template for accessing the file. A filetype is either a single etype or a derived MPI datatype constructed from multiple instances of the same etype. In addition, the extent of any hole in the filetype must be a multiple of the etype's extent. Figure 106 shows the Etype and filetypes.

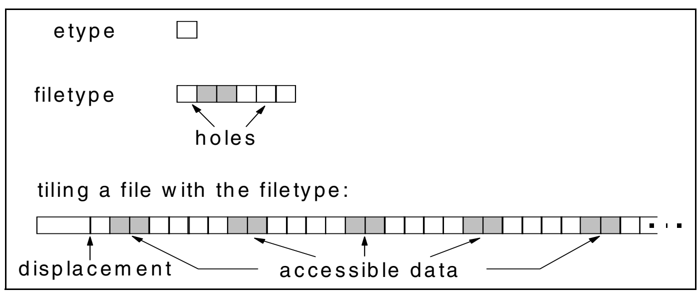

Figure 106. Etype and filetypes

View

A view defines the current set of data visible and accessible from an open file as an ordered set of etypes. Each process has its own view of the file defined by three quantities: A displacement, an etype, and a filetype. The pattern described by a filetype is repeated, beginning at the displacement, to define the view. Views can be changed by the user during program execution. The default view is a linear byte stream (displacement is zero, etype, and filetype equal to MPI_BYTE).

A group of processes can use complementary views to achieve a global data distribution, such as a scatter/gather pattern as shown in Figure 107.

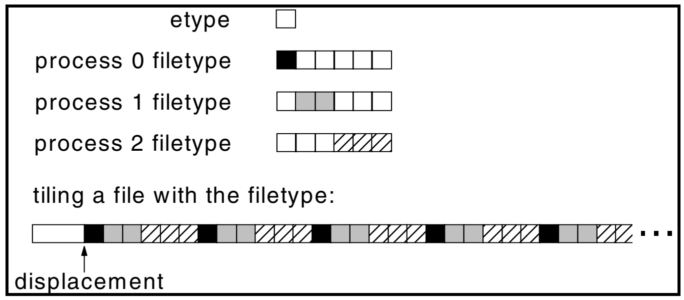

Figure 107. Partitioning a file among parallel processes

Offset

An offset is a position in the file relative to the current view expressed as a count of etypes. Holes in the view's filetype are skipped when calculating this position. Offset 0 is the location of the first etype visible in the view (after skipping the displacement and any initial holes in the view). For example, an offset of two for process one in Figure 107 is the position of the 8th etype in the file after the displacement. An explicit offset is an offset that is used as a formal parameter in explicit data access routines.

File size and end of file

The size of an MPI file is measured in bytes from the beginning of the file. A newly created file has a size of zero bytes. Using the size as an absolute displacement gives the position of the byte immediately following the last byte in the file. For any given view, the end of file is the offset of the first etype accessible in the current view starting after the last byte in the file.

File pointer

A file pointer is an implicit offset maintained by MPI. Individual file pointers are file pointers that are local to each process that opened the file. A shared file pointer is a file pointer that is shared by the group of processes that opened the file.

File handle

A file handle is an opaque object created by MPI_FILE_OPEN and freed by MPI_FILE_CLOSE. All operations on an open file reference the file through the file handle.

6.2.4.2 File manipulation

MPI-IO provides functions for MPI file manipulation, such as open, close, delete, resize, and so on.

MPI_FILE_OPEN

This opens the file identified by a file name with specified access mode on all processes in the communicator group. This is a collective call.

MPI_FILE_CLOSE

This first synchronizes file state, then closes the file. This is a collective call.

MPI_FILE_SET_INFO

This sets new values for the hints specified via info and allows a user to provide information, such as file access patterns and file system specifics to direct optimization. Providing hints may increase I/O performance or minimize the use of system resources.

MPI_FILE_GET_INFO

This returns a new info object containing the hints of the file. The current setting of all hints actually used by the system related to this open file is returned.

6.2.4.3 File view

File view functions set and get information on how to manipulate the data distribution in the MPI file.

MPI_FILE_SET_VIEW

This routine changes the process's view of the data in the file with specified displacement, etype, filetype, and data representation. It also resets the individual file pointers and the shared file pointer to zero.

MPI_FILE_GET_VIEW

This routine returns the process's view of the data in the file.

6.2.4.4 Data access

Data is moved between files and processes by issuing read and write calls. MPI-IO data access calls are characterized by three independent aspects:

- Positioning: Explicit offset or implicit file pointer
- Synchronism: Blocking or non-blocking
- Coordination: Non-collective or collective.

MPI-IO supports three types of positioning methods for data access in the file: Explicit offset, individual file pointer, and shared file pointer.

Data access by explicit offset

Explicit offset operations perform data access at the file position given directly as an argument; no file pointer is used nor updated.

Data access by individual file pointers

Individual file pointers are file pointers that are local to each process that opened the file. MPI maintains one individual file pointer per process per file handle. The current values of this pointer implicitly specifies the offset in the data access routines. After an individual file pointer operation is initiated, the individual file pointer is updated to point to the next etype after the last one that will be accessed. The file pointer is updated relative to the current view of the file.

Data access by shared file pointers

A shared file pointer is a file pointer that is shared by the group of processes that opened the file. MPI maintains exactly one shared file pointer per collective MPI_FILE_OPEN (shared among processes in the file handle group). The current value of this pointer implicitly specifies the offset in the data access routines. For the non-collective shared file pointer routines, the serialization ordering is not deterministic. This non-deterministic serialization is a principal value of shared file pointers. It allows unsynchronized tasks to either get the “next piece” from a file all tasks share or append to the end of a shared file without needing to find the “current end”. If a specific order is required, the user may explicitly enforce it but, perhaps, should reconsider the decision to use a shared file pointer. After a shared file pointer operation is initiated, the shared file pointer is updated to point to the next etype after the last one that will be accessed. The file pointer is updated relative to the current view of the file.

The different positioning methods may be mixed within the same program and do not affect each other. And, these three access modes are each offered in blocking: Non-blocking, collective and non-collective forms.

6.2.4.5 File interoperability

At the most basic level, file interoperability is the ability to read the information previously written to a file, not just the bits of data, but the actual information the bits represent. MPI-IO guarantees full interoperability within a single MPI environment and supports increased interoperability outside that environment through the external data representation as well as the data conversion functions.

MPI-IO predefines three file data formats for interoperability: Native, internal, and external32 as follows:

native
Data in this representation is stored in a file exactly as it is in memory. The advantage of this data representation is that data precision and I/O performance are not lost in type conversions with a purely homogeneous environment. The disadvantage is the loss of transparent interoperability within a heterogeneous MPI environment.

internal
This data representation can be used for I/O operations in a homogeneous or heterogeneous environment; the implementation will perform type conversions, if necessary. This is portable to any system the particular MPI implementation runs on.

external32
This data representation states that read and write operations convert all data from, and to, the external32 representation in a homogeneous or heterogeneous environment. This data representation is portable to every MPI implementation that supports the format.

MPI-IO also supports user defined file data formats.

6.2.4.6 MPI-2 hints and MPI-IO

MPI-2 provides a hints facility. Hints provide the implementation with information about things, such as the structure of the application and the type of expected file accesses for running a set of tasks.

MPI-IO and MPI one-sided communication define APIs that can recognize and use the hints. PE 3.1 supports the following hints for MPI-IO:

- **IBM_io_buffer_size**: This specifies stripe size and buffer space for I/O.
- **IBM_largeblock_io**: This allows an user to state that the application's file access patterns involve either large blocks per read/write or that each task accesses disjoint large blocks of the file even if individual read/writes are small. In other words, there is no fine-grain interleaving of access among tasks. When this hint is used, MPI-IO will skip data shipping and read/write directly to GPFS.
- **filename**: This hint can only be acquired. Its associated value contains the filename that was specified when the file was opened.
- **file_perm**: This hint allows to specify the file permissions to be used when creating a new file (if set) or to inquire the file permissions associated with the file (if acquired).

Valid hints can be used by only those functions that know the meaning of the hints. Hints may not alter program semantic, and incorrect hints do not cause errors. The hints that are specific to the PE implementation of MPI begin with the IBM_prefix. This is done to ensure that codes using these hints can be ported without alteration to another MPI implementation. Each MPI implementation is required to silently ignore hints it does not recognize. The assumption is that no other implementation will use IBM_ as part of a hint name, and hints defined by another MPI will use a distinctive prefix that will let MPI ignore its hints.

6.2.4.7 MPI-IO and GPFS

PE implementation of MPI-IO is targeted to GPFS for production use. Though JFS, NFS, AFS, or DFS can be used for development purposes on a single node or workstation. MPI-IO requires installation and configuration of GPFS for parallel I/O on files that may span multiple disk drives on multiple SP nodes.

The enhancements of inter operation between MPI-IO and GPFS are summarized as follows:

- MPI-IO data shipping reduces load on GPFS token management.
- GPFS Multiple Access Range hint exploited.
- GPFS Data shipping exploited:
  - Eliminates cost of GPFS token management because GPFS is prepared to ship data if it needs to.
  - Data shipping at the MPI-IO level means GPFS is never asked to ship data.

6.3 Dynamic Probe Class Library (DPCL)

DPCL (Dynamic Probe Class Library) is a C++ based class library whose application programming interface (API) enables a program to dynamically insert instrumentation code patches, or “probes”, into an executing program. DPCL is language or programming model independent and scalable, ultimately providing a platform to enable tools developers to build tools with less time and effort. Figure 108 shows the difference between traditional tools structure and DPCL tools structure.

The DPCL is depicted in Figure 108.

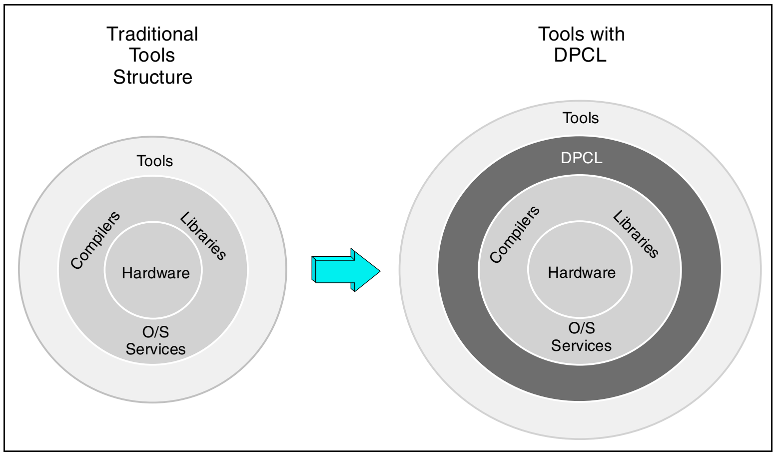

Figure 108. DPCL - Tools platform

The following sections introduce important concepts and benefits of DPCL. For more specific information, refer to *IBM Parallel Environment for AIX: DPCL Programming Guide*, SA22-7420 and *IBM Parallel Environment for AIX: DPCL Class Reference*, SA22-7421.

6.3.1 Dynamic instrumentation

Dynamic instrumentation refers to a specific type of software instrumentation. *Software instrumentation* refers to code that is inserted into a program to gather information regarding the program's run. As the instrumented application executes, the instrumented code then generates the desired information, which could include performance, trace, test coverage, diagnostic, or other data.

Dynamic instrumentation is distinct from the traditional methods of software instrumentation because it can be added and removed while the application is running. The application does not need to be terminated and restarted from the beginning in order to add and remove instrumentation.

6.3.2 DPCL architecture

DPCL has a client-server architecture for communications between control system and distributed instrumentation modules. On the client machine, a program that uses DPCL calls to insert probes is called the "analysis tool". On the server machine, a program that accepts and runs the probes is called the “target application”. On the server machine, there are also the DPCL daemons that are in charge of communications between analysis tools and target applications. In general, the DPCL system consists of an analysis tool, a target application, and DPCL daemons as shown in Figure 109.

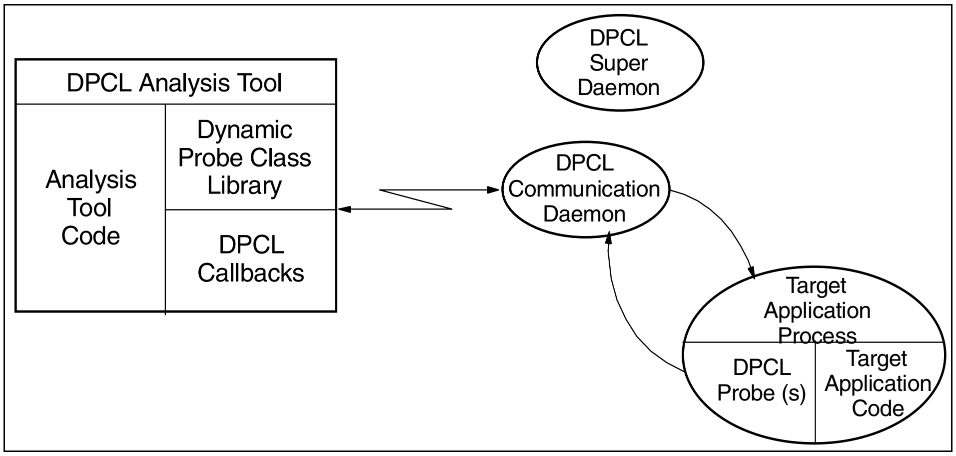

Figure 109. DPCL architecture

6.3.2.1 DPCL target application

A DPCL target application is an executable program into which the analysis tool inserts probes.

6.3.2.2 DPCL analysis tool

A DPCL analysis tool is a C++ application that links in the DPCL library and uses the DPCL API calls to instrument (create probes and insert them into) one or more target application processes.

DPCL API

DPCL’s Application Programming Interface (API) is the key means by which the analysis tool interacts with the DPCL system to effectively instrument a target application.

DPCL callbacks

DPCL callbacks are routines called by the DPCL system when certain messages arrive from a DPCL daemon. When an analysis tool initializes itself to use the DPCL system, one of the things it does is enter the DPCL main event loop so that it can interface asynchronously with the DPCL system. The DPCL main event loop listens to file descriptors and sockets for input; there will be one socket for each remote node to which the analysis tool is connected.

When the DPCL main event loop detects input on a file descriptor that is connected to a DPCL daemon, it calls a dispatch routine for the file descriptor or socket. If the input is on a file descriptor representing a socket connection to a DPCL daemon, the message is examined and the appropriate callback for the message type is executed.

6.3.2.3 DPCL daemons

There are two types of DPCL daemons: DPCL superdaemons and DPCL communication daemons.

- DPCL superdaemons are created the first time an analysis tool calls an API routine to connect to one or more target application processes on a given node. These superdaemons create the DPCL communication daemons and are responsible for ensuring that only one such daemon exists on each remote host. They also perform user authentication on the remote hosts.
- DPCL communication daemons handle the communication between the analysis tool and target application processes. This is the daemon attached to the target application process that will perform much of the actual work requested, via DPCL function calls, by the analysis tool. This daemon also relays the data collected by instrumentation probes within the target application back to the analysis tool.

6.3.2.4 Probe

The term probe refers to the software instrumentation code patch that the analysis tool can insert into the target application. Probes are created by the analysis tool and, therefore, are able to perform any work required by the tool.

Probe expression

A probe expression is a simple instruction or sequence of instructions that represents the executeable code to be inserted into the target application.

Probe module

A probe module is a compiled object file containing one or more functions written in C. Once an analysis tool loads a particular probe module into a target application, a probe is able to call any of the functions contained in the module.

Three types of probes

There are three types of probes; they are differentiated by the manner in which their execution is triggered. The three types of probes are: Point probes, phase probes, and one-shot probes.

- Point probes are installed at particular locations in the target application code and, when in an activated state, are triggered whenever execution reaches that location in the code. The fact that point probes are associated with particular locations within the target application code makes them markedly different from the other two types of probes.
- Phase probes are triggered by expiration of a timer and executed regardless of what code the target application is executing.
- One-shot probes are executed once and immediately, regardless of what code the target application is executing.

6.3.3 Advantages of DPCL

Building analysis tools on the DPCL system has the following advantages:

- Reduces the cost of developing new tools
- Reduces the intrusion cost of instrumentation
- Enables the creation of common tools across an organization or industry
- Enables greater flexibility and inter-operability among tools
- Increases industry innovation in tool development, and, in doing so, increases the number and variety of programming tools

6.3.4 Where is DPCL?

The DPCL is packaged with IBM Parallel Environment for AIX. It can be installed from the ppe.dpcl fileset. It is installed under the /usr/lpp/ppe.dpcl directory.

6.4 POE enhancements

LightWeight Core File (LWCF) and Parallel Task Identification API are enhancements of POE itself in this release.

6.4.1 LightWeight Core File

AIX supports a standard core file format. But, the core files include detailed and low-level information and are usually large in size; so, sometimes they are not very useful for program debugging. More specifically, in parallel programming environment, large parallel jobs need a way of collecting and displaying the state of all threads and processes when a job is abnormally terminated. In such cases, it will take too much time and efforts to examine a traditional core file and find out what caused the abnormal termination. PE

3.1 introduces a new choice of core file format to provide users more simple, helpful, and application-specific information.

6.4.1.1 LigthWeight core file

A LightWeight Core File (LWCF) does not have the often unnecessary low-level detail found in the traditional AIX corefile. The LWCF contains simple process stack traces and is smaller in size than a traditional core file; so, it can be more useful in debugging threaded programs.

6.4.1.2 POE implementation of LightWeight Core File

POE has new option to generate the LWCF **-**corefile_format flag or MP_COREFILE_FORMAT environment variable. Table 9 shows three types of corefile output.

Table 9. Three types of Corefile output

MP_COREFILE_FORMAT or -corefile_format value	Core file Type
Not set or NULL	Standard AIX core file
STDERR	LWCF redirected to user's standard error
Core file name	LWCF with the specified name

The actual core file directory can be specified in conjunction with the existing option -coredir flag or MP_COREDIR environment variable.

6.4.2 Parallel task identification API

PE 3.1 includes a new POE API that allows an application to retrieve the information of parallel jobs originating from a specific node. This information can be used for accounting or to get more detailed information about the tasks spawned by the POE processes.

6.4.2.1 poe_master_tasks()

This call retrieves the list of process IDs of all POE master processes currently running on the specific node. This information can be used for accounting purposes or can be passed to the poe_task_info function to obtain more detailed information about tasks spawned by these POE master processes.

6.4.2.2 poe_task_info()

Given the process ID of a POE master process as input, this call obtains information for each POE task spawned on a local or remote node. For each

POE child task, host name, IP address, task ID, AIX session ID, child process name, and child process ID are obtained.

6.5 Engineering and Scientific Subroutine Library (ESSL)

The Engineering and Scientific Subroutine Library (ESSL) family of products is a state-of-the-are collection of mathematical subroutines that provides a wide range of high-performance mathematical functions for many different scientific and engineering applications.

The ESSL family consist of:

- Engineering and Scientific Subroutine Library (ESSL) for AIX, which contains over 400 high-performance mathematical subroutines tuned to RS/6000 hardware. ESSL runs on RS/6000 workstations, servers, and SP systems.
- Parallel Engineering and Scientific Subroutine Library (Parallel ESSL) for AIX, which is specifically tuned to exploit the full power of the SP hardware with scalability across the range of system configurations. In addition to SP systems, parallel ESSL runs on clusters of RS/6000 servers and/or workstations.

ESSL and Parallel ESSL can be used to develop and enable many different types of scientific and engineering applications. New applications can be designed to take advantage of all the capabilities of ESSL family. Existing applications can be easily enabled by replacing comparable routines and in-line code with calls to ESSL subroutines.

6.5.1 What is new for ESSL Version 3 Release 2?

The new features for ESSL Version 3 Release 2 are:

- The ESSL Libraries are tuned for the RS/6000 POWER3-II.
- The Dense Linear Algebraic Subroutines now include these new subroutines:
  - Symmetric Indefinite Matrix Factorization and Multiple Right-Hand Side Solve
  - Symmetric Indefinite Matrix Factorization
  - Symmetric Indefinite Matrix Multiple Right-Hand Side Solve
- The Linear Least Squares Subroutines now include this new subroutine:
  - General Matrix QR Factorization

- The ESSL POWER and Thread-Safe libraries have been replaced by a thread-safe library referred to as the ESSL Serial Library.
- The ESSL POWER2 and Thread-Safe POWER2 libraries are no longer provided; the ESSL Serial or the ESSL SMP Library should be used instead.

6.5.2 What is new for Parallel ESSL Version 2 Release 2?

The new features for Parallel ESSL Version 2 Release 2 are:

- The Parallel ESSL Libraries are tuned for the RS/6000 POWER3-II Thin, Wide, and High nodes and the SP Switch2.
- The Dense Linear Algebraic Subroutines now include these new subroutines:
  - Inverse or a real or complex general matrix
  - Reciprocal of the condition number of a real or complex general matrix
  - General Matrix QR factorization
  - Least Squares solutions to linear systems of equations for real general matrices
- The Eigensystems Analysis Subroutines now include these new subroutines:
  - Selected Eigenvalues and optionally the eigenvectors of a real symmetric positive definite generalized eigenproblem
  - Reduce a complex Hermitian matrix to tridiagonal form
  - Reduce a real symmetric positive definite generalized eigenproblem to standard form
- The Utilities Subroutine now include these new subroutines:
  - Compute the norm of a real or complex general matrix
- The Parallel ESSL POWER2 and Thread-Tolerant POWER2 libraries are no longer provided; the Parallel ESSL Serial and the Parallel ESSL SMP libraries should be used instead.
- Support is withdrawn for calling Parallel ESSL from HPF; as a result, the Parallel ESSL HPF libraries, HPF module, HPF IVP, and sample HPF programs are no longer provided.

For more detailed information on the Engineering and Scientific Subroutine Library (ESSL) and Parallel Engineering and Scientific Subroutine Library (PESSL), refer to the ESSL and PESSL manuals located at:

`http://www.rs6000.ibm.com/resource/aix_resource/sp_books/essl/index.html`

Appendix A. Software coexistence

Table 10 shows support/coexistence between different levels of AIX and PSSP.

Table 10. AIX and PSSP coexistence

		AIX			PSSP				
		4.1.5	4.2.1	4.3.3	2.2	2.3	2.4	3.1.1	3.2
PSSP	2.2	Y	Y	Y	CI	CI	CI	CI	CI
	2.3	Y	Y	Y	CI	CI	CI	CI	CI
	2.4	N	Y	Y	CI	CI	CI	CI	CI
	3.1.1	N	N	Y	CI	CI	CI	CI	CI
	3.2	N	N	Y	CI	CI	CI	CI	CI

The following notation applies to all tables in this appendix:

Y: Supported

N: Not supported

C: Coexist in the same partition but do not interoperate

CI: Coexist in the same system partition and interoperate

Note

Always check the necessary PTFs required for coexistence and interoperability.

Table 11 shows the support/coexistence between different levels of PSSP, GPFS, and R/VSD.

Table 11. PSSP and applications coexistence

	PSSP					GPFS		
	2.2	2.3	2.4	3.1.1	3.2	1.1	1.2	1.3

		PSSP					GPFS		
PSSP	**2.2**	CI	CI	CI	CI	CI	N	N	N
	2.3	CI	CI	CI	CI	CI	N	N	N
	2.4	CI	CI	CI	CI	CI	Y	N	N
	3.1.1	CI	CI	CI	CI	CI	Y	Y	N
	3.2	CI	CI	CI	CI	CI	Y	Y	Y
GPFS	**1.1**	N	N	Y	Y	Y	CI	C	C
	1.2	N	N	N	Y	Y	C	CI	C
	1.3	N	N	N	N	Y	C	C	CI
R/VSD	**1.2**	Y	Y	Y	Y	Y	Y	N	N
	2.1	N	Y	Y	Y	Y	Y	N	N
	2.1.1	N	N	Y	Y	Y	Y	N	N
	3.1	N	N	N	Y	Y	Y	Y	N
	3.2	N	N	N	N	Y	Y	Y	Y

Table 12 shows the levels of HACMP supported by different levels of AIX and PSSP.

Table 12. AIX, PSSP, and HACMP support/coexistence

		AIX				PSSP				
		4.1.5	**4.2.1**	**4.3.2**	**4.3.3**	**2.2**	**2.3**	**2.4**	**3.1.1**	**3.2**
HACMP	**4.2.1**	Y	Y	Y	Y	N	Y	Y	Y	Y
	4.2.2	N	Y	Y	Y	N	N	Y	Y	Y
	4.3.0	N	N	Y	Y	N	N	N	Y	Y
	4.3.1	N	N	Y	Y	N	N	N	Y	Y
	4.4.0	N	N	N	Y	N	N	N	N	Y

Table 13 shows the levels of LoadLeveler and Parallel Environment supported by different levels of PSSP.

Table 13. LoadLeveler and Parallel Environment support matrix

		PSSP					LL		
		2.2	**2.3**	**2.4**	**3.1.1**	**3.2**	**1.3**	**2.1**	**2.2**
LL	**1.3**	Y	Y	Y	N	N	Y	N	N
	2.1	N	N	Y	Y	Y*	N	Y	N
	2.2	N	N	N	Y	Y	N	C**	Y
PE	**2.2**	Y	N	N	N	N	C	N	N
	2.3	N	Y	Y	N	N	C	N	N
	2.4	N	N	N	Y	N	N	C	C
	3.1	N	N	N	N	Y	N	N	C

* without PE
** PTF in 2.1 is required

Appendix B. Special notices

This publication is intended to help IBM Customers, Business Partners, IBM System Engineers, and other RS/6000 SP specialists who are involved in Parallel System Support Programs (PSSP) Version 3, Release 2 projects, including the education of RS/6000 SP professionals responsible for installing, configuring, and administering PSSP Version 3, Release 2. The information in this publication is not intended as the specification of any programming interfaces that are provided by Parallel System Support Programs. See the PUBLICATIONS section of the IBM Programming Announcement for PSSP Version 3, Release 2 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, North Castle Drive, Armonk, NY 10504-1785.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. 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 pointers in this publication to external Web sites are provided for convenience only and do not in any manner serve as an endorsement of these Web sites.

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

ADSTAR	AIX
AS/400	BookManager
ESCON	HACMP/6000
IBM	LoadLeveler
Netfinity	OS/390
POWERparallel	PowerPC 604
Redbooks	Redbooks Logo
RS/6000	S/390
SP	System/390

The following terms are trademarks of other companies:

Tivoli, Manage. Anything. Anywhere.,The Power To Manage., Anything. Anywhere.,TME, NetView, Cross-Site, Tivoli Ready, Tivoli Certified, Planet Tivoli, and Tivoli Enterprise are trademarks or registered trademarks of Tivoli Systems Inc., an IBM company, in the United States, other countries, or both. In Denmark, Tivoli is a trademark licensed from Kjøbenhavns Sommer - Tivoli A/S.

C-bus is a trademark of Corollary, Inc. in the United States and/or other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and/or other countries.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States and/or other countries.

PC Direct is a trademark of Ziff Communications Company in the United States and/or other countries and is used by IBM Corporation under license.

ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel

Corporation in the United States and/or other countries.

UNIX is a registered trademark in the United States and other countries licensed exclusively through The Open Group.

SET, SET Secure Electronic Transaction, and the SET Logo are trademarks owned by SET Secure Electronic Transaction LLC.

Lotus Notes is a registered trademark of Lotus Development Corporation.

Other company, product, and service names may be trademarks or service marks of others.

Appendix C. Related publications

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

C.1 IBM Redbooks

For information on ordering these publications see "How to get IBM Redbooks" on page 221.

- *Exploiting RS/6000 SP Security: Keeping It Safe*, SG24-5521
- *IBM RS/6000 Clustered Enterprise Servers Handbook*, SG24-5978
- *IBM 9077 SP Switch Router: Get Connected to the SP Switch*, SG24-5157
- *PSSP 3.1 Announcement*, SG24-5332
- *Sizing and Tuning GPFS*, SG24-5610
- *The RS/6000 SP Inside Out*, SG24-5374

C.2 IBM Redbooks collections

Redbooks are also available on the following CD-ROMs. Click the CD-ROMs button at ibm.com/redbooks for information about all the CD-ROMs offered, updates and formats.

CD-ROM Title	Collection Kit Number
IBM System/390 Redbooks Collection	SK2T-2177
IBM Networking Redbooks Collection	SK2T-6022
IBM Transaction Processing and Data Management Redbooks Collection	SK2T-8038
IBM Lotus Redbooks Collection	SK2T-8039
Tivoli Redbooks Collection	SK2T-8044
IBM AS/400 Redbooks Collection	SK2T-2849
IBM Netfinity Hardware and Software Redbooks Collection	SK2T-8046
IBM RS/6000 Redbooks Collection	SK2T-8043
IBM Application Development Redbooks Collection	SK2T-8037
IBM Enterprise Storage and Systems Management Solutions	SK3T-3694

C.3 Other resources

These publications are also relevant as further information sources:

- *General Parallel File System for AIX: Data Management API Guide*, GA22-7435

- *General Parallel File System for AIX: Guide and Reference*, SA22-7452
- *General Parallel File System for AIX: Installation and Tuning Guide*, GA22-7453
- *Parallel Environment for AIX: DPCL Class Reference*, SA22-7421
- *Parallel Environment for AIX: DPCL Programming Guide*, SA22-7420
- *PSSP Administration Guide*, SA22-7348
- *PSSP: Command and Technical Reference*, SA22-7351
- *PSSP: Diagnosis Guide*, GA22-7350
- *PSSP Installation and Migration Guide*, GA22-7347
- *PSSP: Managing Shared Disks*, SA22-7349
- *PSSP Message Reference*, GA22-7352
- *RSCT: First Failure Data Capture Programming Guide and Reference*, SA22-7454
- *RS/6000 SP: Planning, Volume 2, Control Workstation and Software Environment*, GA22-7281

C.4 Referenced Web sites

The following Web sites are also relevant as further sources of information:

- Main site for RS/6000 SP information and support:
  `http://www.rs6000.ibm.com/support/sp`
- `http://www.rs6000.ibm.com/resource/aix_resource/sp_books/essl/index.html`
- `http://w3.itso.ibm.com`
- `http://w3.ibm.com`
- `http://www.elink.ibmlink.ibm.com/pbl/pbl`

How to get IBM Redbooks

This section explains how both customers and IBM employees can find out about IBM Redbooks, redpieces, and CD-ROMs. A form for ordering books and CD-ROMs by fax or e-mail is also provided.

- **Redbooks Web Site** **ibm.com**/redbooks

  Search for, view, download, or order hardcopy/CD-ROM Redbooks from the Redbooks Web site. Also read redpieces and download additional materials (code samples or diskette/CD-ROM images) from this Redbooks site.

  Redpieces are Redbooks in progress; not all Redbooks become redpieces and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

- **E-mail Orders**

  Send orders by e-mail including information from the IBM Redbooks fax order form to:

	e-mail address
In United States or Canada	pubscan@us.ibm.com
Outside North America	Contact information is in the “How to Order” section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl

- **Telephone Orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	Country coordinator phone number is in the “How to Order” section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl

- **Fax Orders**

United States (toll free)	1-800-445-9269
Canada	1-403-267-4455
Outside North America	Fax phone number is in the “How to Order” section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl

This information was current at the time of publication, but is continually subject to change. The latest information may be found at the Redbooks Web site.

IBM Intranet for Employees

IBM employees may register for information on workshops, residencies, and Redbooks by accessing the IBM Intranet Web site at http://w3.itso.ibm.com/ and clicking the ITSO Mailing List button. Look in the Materials repository for workshops, presentations, papers, and Web pages developed and written by the ITSO technical professionals; click the Additional Materials button. Employees may access MyNews at http://w3.ibm.com/ for redbook, residency, and workshop announcements.

IBM Redbooks fax order form

Please send me the following:

Title	Order Number	Quantity

First name Last name

Company

Address

City Postal code Country

Telephone number Telefax number VAT number

☐ Invoice to customer number

☐ Credit card number

Credit card expiration date Card issued to Signature

We accept American Express, Diners, Eurocard, Master Card, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.

Glossary

Adapter. An adapter is a mechanism for attaching parts. For example, an adapter could be a part that electrically or physically connects a device to a computer or to another device. In the SP system, network connectivity is supplied by various adapters, some optional, that can provide connection to I/O devices, networks of workstations, and mainframe networks. Ethernet, FDDI, token ring, HiPPI, SCSI, SSA, FCS, and ATM are examples of adapters that can be used as part of an SP system.

Address. A character or group of characters that identifies a register, a device, a particular part of storage, or some other data source or destination.

AFS. A distributed file system that provides authentication services as part of its file system creation.

AIX. Abbreviation for Advanced Interactive Executive, IBMs licensed version of the UNIX operating system. AIX is particularly suited to support technical computing applications including high function graphics and floating point computations.

API. Application Programming Interface. A set of programming functions and routines that provide access between the application layer of the OSI seven-layer model and applications that want to use the network. It is a software interface.

Application. The use to which a data processing system is put, for example, a payroll application, an airline reservation application, and so on.

Application Data. The data that is produced using an application program.

Authentication. The process of validating the identity of a user or server.

Authorization. The process of obtaining permission to perform specific actions.

Batch Processing. (1) The processing of data or the accomplishment of jobs accumulated in advance in such a manner that each accumulation, thus formed, is processed or accomplished in the same run. (2) The processing of data accumulating over a period of time. (3) Loosely, the execution of computer programs serially. (4) Computer programs executed in the background.

Client. (1) A function that requests services from a server and makes them available to the user. (2) A term used in an environment to identify a machine that uses the resources of the network.

CMI. Centralized Management Interface. provides a series of SMIT menus and dialogues used for defining and querying the SP system configuration.

Connectionless Network. A network in which the sending logical node must have the address of the receiving logical node before information interchange can begin. The packet is routed through nodes in the network based on the destination address in the packet. The originating source does not receive an acknowledgment that the packet was received at the destination.

Consumable resources. The resources whose availability should be considered by LoadLeveler job scheduler to determine how to allocate tasks of a job on nodes. There are configuration-default consumable resources and there can also be administrator-defined resources.

Control Workstation. A single point of control allowing the administrator or operator to monitor and manage the SP system using the IBM AIX Parallel System Support Programs.

CSS. Communication subsystem. Software that provides both user applications and kernel access to the switch adapter for communication and switch management purposes.

Daemon. A process, not associated with a particular user, that performs system-wide functions, such as administration and control of networks, execution of time-dependent activities, line printer spooling, and so forth.

DASD. Direct Access Storage Device. Storage for input/output data.

DFS. Distributed File System. A subset of the IBM Distributed Computing Environment.

DPCL. Dynamic Probe Class Library. A C++ class library whose API enables a program to dynamically insert probes into an executing program. DPCL can be used to create client/server-type analysis tools and also provides greater flexibility and interoperability among such tools.

Ethernet. (1) Ethernet is the standard hardware for TCP/IP local area networks in the UNIX marketplace. It is a 10 Megabit-per-second baseband type LAN that allows multiple stations to access the transmission medium at will without prior coordination, avoids contention by using carrier sense and deference, and resolves contention by collision detection (CSMA/CD). (2) A passive coaxial cable whose interconnections contain devices or components, or both, that are all active. It uses CSMA/CD technology to provide a best-effort delivery system.

Failover. The assumption of server responsibilities by the node designated as backup server when the primary server fails.

Failure Group. A collection of disks that share common access paths or adapter connections and could all become unavailable through a single hardware failure.

Fall Back. Also called fallback, the sequence of events when a primary or server machine takes back control of its workload from a secondary or backup machine.

FFDC. First Failure Data Capture. Facility aimed at enhancing SP and cluster software serviceability by allowing applications to maintain a hierarchy of error messages.

Fiber Distributed Data Interface (FDDI). An American National Standards Institute (ANSI) standard for 100 Megabit-per-second LANs using optical fiber cables. An FDDI local area network (LAN) can be up to 100 km (62 miles) and can include up to 500 system units. There can be up to 2 km (1.24 miles) between system units and/or concentrators.

File Transfer Protocol (FTP). The Internet protocol (and program) used to transfer files between hosts. It is an application layer protocol in TCP/IP that uses TELNET and TCP protocols to transfer bulk-data files between machines or hosts.

File. A set of related records treated as a unit. For example, in stock control, a file could consist of a set of invoices.

File Name. A CMS file identifier in the form of 'filename filetype filemode, such as TEXT DATA A.

File Server. A centrally located computer that acts as a storehouse of data and applications for numerous users of a local area network.

Fragment. The space allocated an amount of data (usually the end of a file) too small to require a full block consisting of one or more subblocks (one thirty-second of block size).

Gateway. An intelligent electronic device interconnecting dissimilar networks and providing protocol conversion for network compatibility. A gateway provides transparent access to dissimilar networks for nodes on either network. It operates at the session presentation and application layers.

HACWS. High Availability Control Workstation function, based on HACMP, provides for a backup control workstation for the SP system.

Hashed Shared Disk (HSD). The data striping device for the IBM Virtual Shared Disk. The device driver lets application programs stripe data across physical disks in multiple IBM Virtual Shared Disks, thus, reducing I/O bottlenecks.

Hello Protocol. The routing protocol used to check the status between the National Foundation Backbone Network (NSFnet) nodes.

High Availability Cluster Multi-Processing. An IBM facility to cluster nodes or components to provide high availability by eliminating single points of failure.

Host. A computer connected to a network, providing an access method to that network. A host provides end-user services.

IBM Virtual Shared Disk. A subsystem that allows application programs executing on different nodes access to a raw logical volume as if it were local at each node.

i-node. The internal structure that describes an individual file to AIX. An i-node contains file size and update information as well as the addresses of data blocks or, in the case of large files, indirect blocks that, in turn, point to data blocks. One i-node is required for each file.

Internet. A specific internetwork consisting of large national backbone networks, such as APARANET, MILNET, and NSFnet, and a myriad of regional and campus networks all over the world. The network uses the TCP/IP protocol suite.

Internet Protocol (IP). (1) A protocol that routes data through a network or interconnected networks. IP acts as an interface between the higher logical layers and the physical network. This protocol, however, does not provide error recovery or flow control or guarantee the reliability of the physical network. IP is a connectionless protocol. (2) A protocol used to route data from its source to its destination in an Internet environment.

IP Address. A 32-bit address assigned to devices or hosts in an IP Internet that maps to a physical address. The IP address is composed of a network and host portion.

Journaled File System. The local file system within a single instance of AIX.

Kerberos. A service for authenticating users in a network environment.

Kernel. The core portion of the UNIX operating system that controls the resources of the CPU and allocates them to the users. The kernel is memory-resident, is said to run in *kernel mode*, and is protected by the hardware from user tampering.

LAN. (1) Acronym for Local Area Network, a data network located on the user's premises in which serial transmission is used for direct data communication among data stations. (2) Physical network technology that transfers data at high speed over short distances. (3) A network in which a set of devices is connected to another for communication and that can be connected to a larger network.

LAPI. Low Level Application Programming Interface. A non-standard IBM communication protocol designed to provide optimal communication performance on the SP switch network. The LAPI library provides PUT and GET functions and a general Active Message function to allow programmers to supply extensions by means of additions to the notification handlers.

Local Host. The computer to which a user's terminal is directly connected.

Logical Volume Manager. Manages disk space at a logical level. It controls fixed-disk resources by mapping data between logical and physical storage allowing data to be discontiguous, span multiple disks, and be replicated and dynamically expanded.

LWCF. Light Weight Core File. A non-standard AIX core file that only contains simple process stack traces. It doesn't have the low-level detail as in the traditional AIX core file. It is, generally, much smaller in size than traditional standard AIX core file.

Metadata. Data structures that contain access information about file data. These might include i-nodes, indirect blocks, and directories. These data structures are used by GPFS but are not accessible to user applications.

Mirroring. The creation of a mirror image of data to be preserved in the event of disk failure.

MPI. Message Passing Interface. An industry standard parallel programming interface that provides message passing libraries to parallelize a job by exchanging messages between more than two tasks. The latest release of the MPI standard is MPI 2.0, which is also referred to as MPI-2.

MPI Hints. An MPI facility that provides information about things, such as the structure of the application, the type of expected file accesses, and preferences for node selection for running a set of tasks. The MPI standard defines the reserved hints and also allows for vendors' own implementation.

MPI-IO. The I/O component of MPI. It provides a set of interfaces to perform portable and efficient parallel I/O.

MPI One-sided Communication. MPI functionality that extends the communication mechanisms of MPI by allowing one process to specify all communication parameters, both for the sending and receiving sides of message passing. It is also referred to as MPI 1-sided.

MSS. Master Switch Sequencing node. The node which periodically resequences the TOD signals on the SP Switch2.

Network. An interconnected group of nodes, lines, and terminals. A network provides the ability to transmit data to and receive data from other systems and users.

NFS. Network File System. NFS allows different systems (UNIX or non-UNIX), different architectures, or vendors connected to the same network to access remote files in a LAN environment as though they were local files.

ODM. Object Data Manager. In AIX, a hierarchical object-oriented database for configuration data.

Parallel Environment. A system environment where message passing or SP resource manager services are used by the application.

Parallel Environment. A licensed IBM program used for message passing applications on the SP or RS/6000 platforms.

Parallel Processing. A multiprocessor architecture that allows processes to be allocated to tightly-coupled multiple processors in a cooperative processing environment allowing concurrent execution of tasks.

Parameter. (1) A variable that is given a constant value for a specified application and that may denote the application. (2) An item in a menu for which the operator specifies a value or for which the system provides a value when the menu is interpreted. (3) A name in a procedure that is used to refer to an argument that is passed to the procedure. (4) A particular piece of information that a system or application program needs to process a request.

Primary node or machine. (1) A device that runs a workload and has a standby device ready to assume the primary workload if that primary node fails or is taken out of service. (2) A node on the SP Switch that initializes, provides diagnosis and recovery services, and performs other operations to the switch network. (3) In IBM Virtual Shared Disk function, when physical disks are connected to two nodes (twin-tailed), one node is designated as the primary node for each disk, and the other is designated the secondary, or backup, node. The primary node is the server node for IBM Virtual Shared Disks defined on the physical disks under normal conditions. The secondary node can become the server node for the disks if the primary node is unavailable (off-line or down).

Primary Server. When physical disks are connected to two nodes (twin-tailed), this is the node that normally maintains and controls local access to the disk.

Process. (1) A unique, finite course of events defined by its purpose or by its effect, achieved under defined conditions. (2) Any operation or combination of operations on data. (3) A function being performed or waiting to be performed. (4) A program in operation. For example, a daemon is a system process that is always running on the system.

Protocol. A set of semantic and syntactic rules that defines the behavior of functional units in achieving communication.

Quorum. The minimum number of nodes that must be running in order for the GPFS daemon to start. This is one plus half of the number of nodes in the GPFS configuration.

Quota. The amount of disk space and number of i-nodes assigned as upper limits for a specified user or group of users.

RAID. Redundant Array of Independent Disks. A set of physical disks that act as a single physical volume and use parity checking to protect against disk failure.

Recovery. The process of restoring access to file system data when a failure has occurred. This may involve reconstructing data or providing alternative routing through a different server.

Replication. The practice of creating and maintaining multiple file copies to ensure availability in the event of hardware failure.

RISC. Reduced Instruction Set Computing (RISC), the technology for today's high performance personal computers and workstations, was invented in 1975. Uses a small simplified set of frequently used instructions for rapid execution.

rlogin (remote LOGIN). A service offered by Berkeley UNIX systems that allows authorized users of one machine to connect to other UNIX systems across a network and interact as if their terminals were connected directly. The rlogin software passes information about the user's environment (for example, terminal type) to the remote machine.

RPC. Acronym for Remote Procedure Call, a facility that a client uses to have a server execute a procedure call. This facility is composed of a library of procedures plus an XDR.

RSH. A variant of the `RLOGIN` command that invokes a command interpreter on a remote UNIX machine and passes the command line arguments to the command interpreter, thus, skipping the LOGIN step completely. See also rlogin.

SCSI. Small Computer Systems Interface. An adapter supporting attachment of various direct-access storage devices.

SDR. System Data Repository. Database for SP system configuration information. Resides on the control workstation only. Information entered through the SMIT panels or commands during installation goes into the SDR. Much of the information in the SDR can be viewed, and some can be entered or changed using SP Perspectives or PSSP commands.

Secondary Node. In IBM Virtual Shared Disk function, when physical disks are connected to two nodes (twin-tailed), one node is designated as the primary node for each disk and the other is designated as the secondary, or backup, node. The secondary node acts as the server node for the IBM Virtual Shared disks defined on the physical disks if the primary node is unavailable (off-line or down).

Secondary Server. The second node connected to a twin-tailed disk. This node assumes control of local access if the primary server fails.

Server. (1) A function that provides services for users. A machine may run client and server processes at the same time. (2) A machine that provides resources to the network. It provides a network service, such as disk storage and file transfer, or a program that uses such a service. (3) A device, program, or code module on a network dedicated to providing a specific service to a network. (4) On a LAN, a data station that provides facilities to other data stations. Examples are file server, print server, and mail server.

Shared Memory MPI. An implementation of MPI that allows tasks of a parallel job on the same node to send or receive their messages through the system's shared memory instead of physical network. PE 3.2 supports the shared memory MPI.

Shell. The shell is the primary user interface for the UNIX operating system. It serves as command language interpreter and programming language and allows foreground and background processing. There are three different implementations of the shell concept: Bourne, C, and Korn.

SMIT. The System Management Interface Toolkit is a set of menu-driven utilities for AIX that provides functions, such as transaction login, shell script creation, automatic updates of object database, and so forth.

SNMP. Simple Network Management Protocol. (1) An IP network management protocol that is used to monitor attached networks and routers. (2) A TCP/IP-based protocol for exchanging network management information and outlining the structure for communications among network devices.

Socket. (1) An abstraction used by Berkeley UNIX that allows an application to access TCP/IP protocol functions. (2) An IP address and port number pairing. (3) In TCP/IP, the Internet address of the host computer on which the application runs and the port number it uses. A TCP/IP application is identified by its socket.

SSA. Serial Storage Architecture. An expanded storage adapter for multi-processor data sharing in UNIX-based computing allowing disk connection in a high-speed loop.

Standby Node or Machine. A device that waits for a failure of a primary node in order to assume the identity of the primary node. The standby machine then runs the primary's workload until the primary is back in service.

SP-Attached servers. Standalone servers that attach to the SP and might include a switch attachment. From a switch administration point of view, these nodes are fully-functional with an SP Switch. Not supported in SP Switch2 environments.

SP Switch. The medium that allows high speed communication between SP system nodes. The software called the CSS supports the communication over the SP Switch.

SP Switch2. The SP Switch2 is available as a 16-port configuration. This switch can be used only in SP systems populated with POWER3 High Nodes.

SP Switch router. A licensed version of the Ascend GRF switched IP router that is enhanced for direct connection to the SP Switch. Network connections through SP Switch Routers are typically faster and have better availability than network connections through SP system nodes.

Stripe Group. A file system written across many disks that are connected to multiple nodes.

Striping. A method of writing a file system, in parallel, to multiple disks instead of to single disks in a serial operation.

Sub-block. The smallest unit of data accessible in an I/O operation equal to one thirty-second of a data block.

Subnet. Shortened form of subnetwork.

Subnet Mask. A bit template that identifies to the TCP/IP protocol code the bits of the host address that are to be used for routing for specific subnetworks.

Subnetwork. Any group of nodes that have a set of common characteristics, such as the same network ID.

Subsystem. A software component that is not usually associated with a user command. It is usually a daemon process. A subsystem will perform work or provide services on behalf of a user request or operating system request.

Switch. A message-passing network that connects all processor nodes with a minimum of four paths between every pair of nodes.

Sysctl. Secure System Command Execution Tool. An authenticated client/server system for running commands remotely and in parallel.

System Partition. A group of non-overlapping nodes on a switch chip boundary that act as a logical SP system.

tar. Tape ARchive is a standard UNIX data archive utility for storing data on tape media.

TCP. Acronym for Transmission Control Protocol, a stream communication protocol that includes error recovery and flow control.

TCP/IP. Acronym for Transmission Control Protocol/Internet Protocol, a suite of protocols designed to allow communication between networks regardless of the technologies implemented in each network. TCP provides a reliable host-to-host protocol between hosts in packet-switched communications networks and in interconnected systems of such networks. It assumes that the underlying protocol is the Internet Protocol.

Telnet. Terminal Emulation Protocol, a TCP/IP application protocol that allows interactive access to foreign hosts.

Token Management. A system for controlling file access in which each application performing a read or write operation is granted exclusive access to a specific block of file data. This ensures data consistency and controls conflicts.

Token Ring. (1) Network technology that controls media access by passing a token (special packet or frame) between media-attached machines. (2) A network with a ring topology that passes tokens from one attaching device (node) to another. (3) The IBM Token Ring LAN connection allows the RS/6000 system unit to participate in a LAN adhering to the IEEE 802.5 Token Passing Ring standard or the ECMA standard 89 for Token Ring, baseband LANs.

Transaction. An exchange between the user and the system. Each activity the system performs for the user is considered a transaction.

UNIX Operating System. An operating system developed by Bell Laboratories that features multiprogramming in a multiuser environment. The UNIX operating system was originally developed for use on minicomputers but has been adapted for mainframes and microcomputers. Note: The AIX operating system is IBMs implementation of the UNIX operating system.

User. Anyone who requires the services of a computing system.

User Datagram Protocol (UDP). (1) In TCP/IP, a packet-level protocol built directly on the Internet Protocol layer. UDP is used for application-to-application programs between TCP/IP host systems. (2) A transport protocol in the Internet suite of protocols that provides unreliable, connectionless datagram service. (3) The Internet Protocol that enables an application programmer on one machine or process to send a datagram to an application program on another machine or process.

User ID. A non-negative integer, contained in an object of type uid_t that is used to uniquely identify a system user.

Virtual Shared Disk, IBM. The function that allows application programs executing at different nodes of a system partition to access a raw logical volume as if it were local at each of the nodes. In actuality, the logical volume is local at only one of the nodes (the server node).

Workstation. (1) A configuration of input/output equipment at which an operator works. (2) A terminal or microcomputer, usually one that is connected to a mainframe or to a network, at which a user can perform applications.

X Window System. A graphical user interface product.

Index

IBM Redbooks review

Your feedback is valued by the Redbook authors. In particular we are interested in situations where a Redbook "made the difference" in a task or problem you encountered. Using one of the following methods, **please review the Redbook, addressing value, subject matter, structure, depth and quality as appropriate.**

- Use the online **Contact us** review redbook form found at **ibm.com**/redbooks
- Fax this form to: USA International Access Code + 1 914 432 8264
- Send your comments in an Internet note to redbook@us.ibm.com

Document Number **Redbook Title**	SG24-5673-00 PSSP 3.2: RS/6000 SP Software Enhancements
Review	
What other subjects would you like to see IBM Redbooks address?	
Please rate your overall satisfaction:	O Very Good O Good O Average O Poor
Please identify yourself as belonging to one of the following groups:	O Customer O Business Partner O Solution Developer O IBM, Lotus or Tivoli Employee O None of the above
Your email address: The data you provide here may be used to provide you with information from IBM or our business partners about our products, services or activities.	O Please do not use the information collected here for future marketing or promotional contacts or other communications beyond the scope of this transaction.
Questions about IBM's privacy policy?	The following link explains how we protect your personal information. **ibm.com**/privacy/yourprivacy/

IBM Redbooks

PSSP 3.2: RS/6000 SP Software Enhancements

PSSP 3.2: RS/6000 SP Software Enhancements

Information on the latest hardware and software for the RS/6000 SP system

Install, tailor, and configure the new hardware and software

Planning information to help you design solutions for migration to new hardware

In July 2000, IBM announced enhancements to the RS/6000 SP that included a new version of the Parallel System Support Program, Version 3 Release 2 (PSSP 3.2), and support for clusters of S-series servers.

After an overview of the announcement, this redbook goes into detail about various hardware and software enhancements including support for a cluster of S-series Enterprise servers without an SP, support for the Distributed Computing Environment (DCE) Security Services, RAS enhancements including First Failure Data Capture (FFDC), new diagnostics for customer engineers, and new diagnostics for managing the SP Switch or the SP Switch2.

Other SP-related products have also announced new releases. This redbook discusses General Parallel File System 1.3, LoadLeveler 2.2, and Parallel Environment for AIX 3.1. This redbook is for IBM customers, Business Partners, IBM Technical and marketing professionals, and anyone seeking to understand the new hardware and software components and improvements included in this RS/6000 SP announcement.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks

SG24-5673-00 ISBN 0738417165