A Minute with Nsort on a 32P NEC Windows Itanium2 Server

Chris Nyberg <chris@ordinal.com>, Ordinal Technology Corp
Jim Gray <gray@microsoft.com>, Microsoft Corporation
Charles Koester <charles@koester.com>, Ordinal Technology Corp

Abstract

In March 2004, the Nsort™ program was able to sort 34 GB of data (340,000,000 100-byte records) in 58 seconds on a 32 processor Itanium® 2 NEC® Express5800/1320Xd running Microsoft® Windows® Server 2003 Datacenter Edition. This set new records for the MinuteSort benchmark. The data was read from one input file at 1.4 GB/sec. The output file was created and written at 1.2 GB/sec. Both files resided on an NTFS file system on a striped logical volume consisting of 8 Eurologic 14-disc SAN blocs attached by 8 Qlogic 2350 SAN HBAs. The 34 GB data set was read into main memory, sorted, and written to the output file. No temporary (or scratch) files were used. In April 2004 on the same NEC server, Nsort was able to sort a 1 TB file in 33 minutes using 21 Qlogic HBAs and 41 Eurologic SAN blocs. This set a new record in the TeraByte Sort benchmark.

Introduction

Mainframes have traditionally been thought of delivering the best sort performance. In the 1990's several hardware and software trends allowed large Unix systems to deliver the best sort performance for SMP servers. Now those same trends allow large Windows systems to have excellent sort performance.

To demonstrate the realization of these trends, we borrowed some time on a large NEC Windows system that had been configured for running the TPC-C benchmark. The system contained 32 1.5 Ghz Itanium 2 processors and 128 GB of main memory, and 904 fiber channel attached 36GB 15krpm SCSI disks . A subset of the available host bus adaptors and disks was used. The Nsort program, which previously held the MinuteSort record in the Daytona (commercial) category, was able to sort 34 GB in 58 seconds. This set new MinuteSort records in both the Indy (customized program allowed) and Daytona categories.

In a subsequent use of the same NEC server, Nsort was able to sort a 1 TB file in 33 minutes. This sort used temporary files and a larger I/O configuration. It set a new record in the Daytona TeraByte Sort benchmark. This was done after the 1 April 2004 annual deadline for the benchmark, so it will not be "official" until 1 April 2005.

This paper presents some background on sorting and the MinuteSort and TeraByte Sort benchmarks, then describes the server hardware, system software and record-breaking sort runs.

Background

Sorting was one of the first commercial applications for computers and is a classic problem in computer science [5]. Sorting is commonly used to bring together records with equal keys, or to facilitate later searching of data. Modern-day sort programs can include or omit records based on key value, reorder record fields, and aggregate field values by key value. For example, many utility and telephone companies use sorting to bring together service usage records by account number, subtotaling the service charges in the process.

Sorting is used in data mining applications to find patterns and trends. For example, many web sites facilitate the processing of billion-record web access logs by sorting the log records, either by requested page, requestor or time. In retailing, sorts are often used to aggregate data by various categories and produce rollup and cube reports.

Sorting is also a core utility for database systems in organizing, indexing, and reorganizing data. Presorting data usually reduces database load times dramatically:

- Database tables are usually organized in a B-tree with records in key order. Pre-sorted data can be directly loaded into a table without the database performing its own internal sort at a slow speed.
- In data warehouses, fact-table data is summarized by all combinations of its multiple dimensions. Building these aggregates with a sorting program is usually much more efficient than using the database [6].

The following hardware trends allowed UNIX systems to eclipse IBM mainframes in processing power and sort performance. Recently, these same trends, along with some Windows-specific ones, have allowed the performance of large Windows servers to skyrocket.

- Microprocessors with faster and faster clocks were developed. These processors have become both very fast and relatively inexpensive. They also rely on multiple levels of cache memory to run at full speed.
- Symmetric multiprocessor machines (SMPs) were built to allow multiple processors to work on the same task.
- Non-Uniform Memory Access (NUMA) machines have been developed to get around the bandwidth limitations of a single memory bus SMP machines.
- Main memory prices decreased to the point that multi-gigabyte memories are now standard.
- 64-bit addressing was incorporated to allow a single process to exploit large memories.
- Commodity disks became cheaper and faster -- indeed, mainframes now use the same disks as other systems (albeit with a different interface).
- Volume managers and RAID storage processors have been developed that allow the bandwidth of many disks to be combined as virtual disk devices whose speed is limited only by the raw hardware.
- SMP scalability of Windows systems has increased dramatically.
- The NTFS file system has been developed and improved to the point that it now offers exceptional performance.

Nsort is designed to take advantage of today's large, multiprocessor systems. It can access files stored on multi-disk devices at high speeds. Nsort has sophisticated buffer management to overlap computation and I/O. Its multi-threaded code allows multiple processors to be used for one sort. Nsort's algorithms pay particular attention to processor-cache locality to make the best use of fast microprocessors. By using 64-bit addressing, Nsort can perform a one-pass sort for very large data sets. The size limitation for one-pass sorts is the user's budget for main memory, not a 2GB or 4GB barrier imposed by 32-bit addressing.

MinuteSort

MinuteSort is a standard sorting benchmark [7]. The benchmark measures the number of 100-byte records that can be sorted in one minute of elapsed time. The input records have 10-byte random keys. The minute limit includes the time to:

- Launch the sort program
- Read the input file
- Sort the data
- Create and write the output file

MinuteSort is a successor benchmark to the Datamation sort benchmark, which has since been retired. The Datamation benchmark used one million 100-byte records – much too easy a sort for today's computers. There are two categories for the MinuteSort benchmark: Indy (custom, "benchmark special" sort programs are allowed) and Daytona (restricted to commercial, general purpose sort programs). The previous Indy MinuteSort winners and sort sizes achieved are displayed in the following table:

Previous Indy MinuteSort Records

Year	Sort Name	Size	Passes
2000	HPVM MinuteSort [9]	21.8 GB	One-Pass
1998	NowSort [2]	8.41 GB	One-Pass
1997	NowSort [1]	6.0 GB	One-Pass

The Daytona MinuteSort contest has been dominated by Nsort. All past Nsort records were achieved on SGI Origin2000 systems.

Previous Daytona MinuteSort Records

Year	Sort Name	Size	Passes
1999	Nsort [8]	12.0 GB	One-Pass
1997	Nsort [8]	7.6 GB	Two-Pass
1997	Nsort [8]	5.3 GB	One-Pass

TeraByte Sort

TeraByte Sort is also a standard sorting benchmark. The benchmark measures the elapsed time to sort 1 terabyte of data (10,000,000,000 100-byte records. As with MinuteSort, the input records have 10-byte random keys. The minute limit includes the time to:

- Launch the sort program
- Read the input file
- Sort the data
- Write the output file

As with MinuteSort, there are Daytona and Indy categories. The previous Daytona TeraByte Sort winners and sort sizes achieved are displayed in the following table:

Previous Daytona TeraByte Sort Records

Year	Sort Name	Minutes	Passes
1999	Compaq [3]	45	Two-Pass
1997	Nsort [8]	150	Two-Pass

The first announced terabyte sort result, 150 minutes, was by Nsort on a 32 processor SGI Origin system. That record was later lowered to 45 minutes on a 144 processor Compaq system.

Hardware

To attempt to set a new MinuteSort record, we borrowed a weekend of time on an NEC Express5800/1320Xd that was being used for TPC-C benchmarking. The server contained 32 Intel 1.5 GHz Itanium 2 processors each with a 6MB L3 cache. The system had 128 GB of main memory. The Itanium processors' 64-bit capability allowed us to address the entire 34 GB input file in main memory, and avoid having to write out any temporary data to disk. We knew from prior experience that memory bandwidth would be critical. The Express5800/1320Xd employs NEC's NUMA, crossbar switch, and high-speed memory access technologies.

The I/O subsystem consisted of:

- 16 QLogic 2350 SAN Host Bus Adaptors
- 64 2Gbit Eurologic® 14-disc SAN blocs

We were only allowed to use 8 of the QLogic HBAs as the other 8 were dedicated to TPC-C terminal I/O. The system was designed to provide a high number of disk requests per second at a minimal cost. For sorting, disk bandwidth is the critical metric. It turned out one of the Eurologic SAN blocs could flood the bandwidth capacity of an HBA giving a data rate of about 195 MBps for either reading or writing. For the MinuteSort benchmark we then used:

- 8 QLogic 2350 SAN Host Bus Adaptors
- 8 2Gbit Eurologic 14-disc SAN blocs

We created a striped logical volume using the 8 SAN blocs, and created an NTFS file system on it using 64KB allocation blocks.

For our attempt on the TeraByte Sort benchmark, we used the same NEC system and memory size. We needed at least 2TB of disk space to perform the sort: 1TB for the input/output file; and 1TB for the temporary file data. We only had access to 57GB of disk space on the SAN blocs, due to resident TPC-C data. This required a larger number of SAN blocs. Since we were using temporary files, we used a larger number of HBAs to accommodate the larger bandwidth requirement. We used:

- 21 QLogic 2350 SAN Host Bus Adaptors
- 41 2Gbit Eurologic 14-disc SAN blocs

System Software

The NEC server was running Microsoft® Windows® Server 2003, Datacenter Edition (SP1, build build 1149). We were particularly impressed by NTFS. In the one-pass MinuteSort, Nsort allocated more than 34 GB of process memory, and read into it the 34GB input file. Windows and NTFS allow space for a newly allocated file to be preallocated. We used to feature to increase the write rate to the sort output file.

Running the Sorts

To run the MinuteSort benchmark, we first generated a 34GB file of 100-byte records using random 10-byte keys on our striped logical volume consisting of 8 SAN blocs.

We ran Nsort using a custom program we wrote, *timex*, to measure the elapsed time and CPU usage of the Nsort job. The output of that command and an Nsort performance statistics report are given below:

```
C:\nsort>del g:\out.dat
C:\nsort>timex nsort -sp:f100.spec g:\34g.dat,direct,trans=32m  -o
g:\out.dat,direct,trans=32m,preallocate -stat -proc:30 -mem:50g
Nsort version 3.2.31 (Windows 64-bit) using 38110M of memory out of 51200M

Pointer sort performed Sat Mar 27 16:06:37 2004
          Input Phase       Output Phase         Overall
Elapsed    23.84              34.63              58.48
I/O Busy   23.73   100%       27.76    99%       51.49
Action  User   Sys Busy    User   Sys Busy    User   Sys Busy
sort  218.53 24.17 1018% 592.99  6.30 1731% 811.52 30.47 1440%
    Majflt    Minflt  Sort Procs
     0/0           0          30
File Name          I/O Mode Busy   Wait MB/sec  Xfers        Bytes     Records
Input Reads
  g:\34g.dat       direct   100%  23.69   1399   1014  34000000000   340000000
Output Writes
  g:\out.dat       direct    99%   0.53   1186   1014  34000000000   340000000
ExitCode: 0
Elapsed Time:     58.510
Kernel Time :     33.340
User Time   :    811.520
```

The parameters on the Nsort command line were as follows:

`-sp:f100.spec` Directs Nsort to read the *f100.spec* sort specification file. This file describes the 100-byte records. The contents of *f100.spec* are as follows:

```
# Datamation, MinuteSort records
/format=(record_size:100)
/field=(NAME=first, binary, unsigned, off=0,  size=10)
/field=(NAME=last,  binary, unsigned, off=10, size=90)
/key=first
```

`g:\34g.dat,direct,trans=32m` Specifies the name of the sort input file, that it should be read direct (bypassing the Windows buffer cache), and that the read request sizes should be 32MB in size.

`-o g:\out.dat,direct,trans=32m,preallocate` Specifies the name of the sort output file, that it should be read direct (bypassing the Windows buffer cache), the read request sizes should be 32MB in size, and that space for the file should be preallocated.

`-stat` Directs Nsort to print a statistics report.

`-proc:30` Specifies that Nsort should create 30 threads to perform sorting. Nsort will not create more than 8 sort threads by default.

`-mem:50g` Directs Nsort to use up to 50GB of memory. According to the statistics output, Nsort only used 38GB of memory.

The statistics indicate Nsort took 58 seconds to sort 34GB of data. This is a new MinuteSort record in both the Indy and Daytona categories. We ran 35GB sorts and they sometimes completed within a minute. The ½ dozen 34 GB sorts, all completed within a minute.

During the input phase, Nsort used an average of 10 cpu seconds per second. The input file was read at an average rate of 1.399 GB/sec. During the output phase Nsort used an average of 17 processors. The output file was written at a rate of 1.186 GB/sec.So, it was IO bound rather than being cpu bound.

To run the TeraByte Sort benchmark, we created a 1TB file of 100-byte records with random 10-byte keys. The file system was on a striped volume consisting of 21 SAN blocks. For the temporary files we used 20 separate volumes consisting of one SAN bloc each. The input file was also used as the output file. The Nsort command line and performance statistics report are given below:

```
C:\nsort>nsort /spec:1tb.spec /key=first k:\1tb.dat /o k:\1tb.dat
Nsort version 3.2.34-proto (Windows 64-bit) using 6364M of memory out of 40960M
Pointer sort performed Sun Apr 18 20:35:30 2004
          Input Phase       Output Phase         Overall
Elapsed     1040              931.67               1971
I/O Busy  430.71    42%      929.94   100%        1360
Temp Max  375.92    36%      355.17    38%
Action  User   Sys Busy    User   Sys Busy    User   Sys Busy
sort   20013   382 1961%  17418   417 1914%  37431   799 1939%
    Majflt    Minflt  Sort Procs
     0/0           0          30
File Name                  I/O Mode Busy   Wait MB/sec  Xfers        Bytes     Records
Input Reads
  k:\1tb.dat               direct    42%  21.13 943.64  23842 1000000000000 10000000000
Temporary Writes
  c:\sort\t0\nsort_a01172  direct    36%   0.01  52.33   8869  55549034496
  c:\sort\t1\nsort_a01172  direct    36%   0.05  52.34   8869  55549034496
  c:\sort\t2\nsort_a01172  direct    36%   0.31  52.35   8869  55549034496
  c:\sort\t3\nsort_a01172  direct    36%   0.09  52.35   8869  55543119872
  c:\sort\t4\nsort_a01172  direct    36%   1.03  52.37   8868  55542743040
  c:\sort\t5\nsort_a01172  direct    36%   0.01  52.38   8868  55542743040
  c:\sort\t6\nsort_a01172  direct    36%   0.67  52.38   8868  55542743040
  c:\sort\t8\nsort_a01172  direct    36%   0.28  52.40   8868  55546871808
  c:\sort\t9\nsort_a01172  direct    36%   0.03  52.41   8869  55553163264
  c:\sort\t10\nsort_a01172 direct    36%   0.04  52.42   8870  55559454720
  c:\sort\t11\nsort_a01172 direct    36%   0.10  52.44   8871  55565746176
  c:\sort\t12\nsort_a01172 direct    36%   0.67  52.45   8872  55572037632
  c:\sort\t14\nsort_a01172 direct    36%   0.19  52.46   8872  55578320896
  c:\sort\t15\nsort_a01172 direct    36%   0.08  52.46   8872  55578320896
  c:\sort\t16\nsort_a01172 direct    36%   0.14  52.45   8872  55574192128
  c:\sort\t17\nsort_a01172 direct    36%   0.00  52.45   8871  55567900672
  c:\sort\t18\nsort_a01172 direct    36%   0.02  52.44   8870  55561609216
  c:\sort\t19\nsort_a01172 direct    36%   0.03  52.43   8869  55555317760
 All                                        3.75 942.11 159656 1000031387648
Temporary Reads
  c:\sort\t0\nsort_a01172  direct    36%   0.00  58.38   8869  55549034496
  c:\sort\t1\nsort_a01172  direct    37%   0.00  58.38   8869  55549034496
  c:\sort\t2\nsort_a01172  direct    38%   0.00  58.38   8869  55549034496
  c:\sort\t3\nsort_a01172  direct    38%   0.00  58.37   8869  55543119872
  c:\sort\t4\nsort_a01172  direct    38%   0.00  58.37   8868  55542743040
  c:\sort\t5\nsort_a01172  direct    37%   0.00  58.37   8868  55542743040
  c:\sort\t6\nsort_a01172  direct    38%   0.00  58.37   8868  55542743040
  c:\sort\t8\nsort_a01172  direct    38%   0.00  58.37   8868  55546871808
  c:\sort\t9\nsort_a01172  direct    36%   0.00  58.38   8869  55553163264
  c:\sort\t10\nsort_a01172 direct    37%   0.00  58.39   8870  55559454720
  c:\sort\t11\nsort_a01172 direct    38%   0.00  58.39   8871  55565746176
  c:\sort\t12\nsort_a01172 direct    40%   0.00  58.40   8872  55572037632
  c:\sort\t14\nsort_a01172 direct    39%   0.00  58.41   8872  55578320896
  c:\sort\t15\nsort_a01172 direct    39%   0.00  58.41   8872  55578320896
  c:\sort\t16\nsort_a01172 direct    39%   0.00  58.40   8872  55574192128
  c:\sort\t17\nsort_a01172 direct    38%   0.00  58.39   8871  55567900672
  c:\sort\t18\nsort_a01172 direct    38%   0.00  58.39   8870  55561609216
  c:\sort\t19\nsort_a01172 direct    38%   0.00  58.38   8869  55555317760
 All                                        0.00   1051 159656 1000031387648
Output Writes
  k:\1tb.dat               direct   100%    590   1049  23842 1000000000000 10000000000
```

The contents of the *1tb.spec* file specified on the Nsort command line were as follows:

```
/format:size=100
/field=first,offset=0,size=10
/field=second,offset=10,size=10
/temp=c:\sort\t0,c:\sort\t1,c:\sort\t2,c:\sort\t3,c:\sort\t4,
  c:\sort\t5,c:\sort\t6,c:\sort\t8,c:\sort\t9,c:\sort\t10,
  c:\sort\t11,c:\sort\t12,c:\sort\t14,c:\sort\t15,c:\sort\t16,
  c:\sort\t17,c:\sort\t18,c:\sort\t19,trans=6m,count=4,preallocate
/filesys=k:,trans=40m,count=4
/processes=30
/memory=40g
/statistics
```

As seen in the Nsort statistics, the elapsed time for the 1TB sort was 1971 seconds, or 32 minutes 51 seconds. This is faster than the previous Daytona TeraByte Sort record of 45 minutes. The input file was read at 943 MB/sec, and the output file written at 1049 MB/sec. During both the input phase and output phase, the cpu utilization was around 19 cpu seconds per second (19.6 and 19.1).

Conclusion

Windows can now match the IO performance of large Unix systems. Nsort was able to use this performance with minimal tuning to deliver world-class performance – reading and writing at more than 1GBps.

Acknowledgments

NEC and Intel supported this work by making the NEC® Express5800/1320Xd available to us. We especially appreciate the help of Gerrit Saylor of Intel, who configured the hardware for us and helped us understand the configuration.

References

[1] Andrea C. Dusseau, Remzi H. Arpaci, David E. Culler, Joseph M. Hellerstein, and David A. Patterson. High-performance sorting on Network of Workstations. In SIGMOD'97 Tucson, Arizona, pages 234{254, May 1997.

[2] Andrea C. Dusseau, Remzi H. Arpaci, David E. Culler, Joseph M. Hellerstein, and David A. Patterson. Searching for the sorting record: Experiences in tuning NOW-Sort. In Symposium on Parallel and Distributed Tools (SPDT'98), August 1998

[3] Fineberg, S., P. Mehra, "The Record-Breaking Terabyte Sort on a Compaq Cluster", Proceedings of the 3rd USENIX Windows NT Symposium, Seattle, WA, July 1999

[4] Gray, J., The Sort Benchmark Homepage, http://research.microsoft.com/barc/SortBenchmark/

[5] Knuth, D.E., The Art of Computer Programming, Vol. 3, Addison-Wesley, Reading, MA, 1973.

[6] Kimball, R., The Data Warehouse Toolkit, John Wiley & Sons, New York, 1996, p. 222.

[7] Nyberg, C., T. Barclay, Z. Cvetanovic, J. Gray, D. Lomet, "AlphaSort: A RISC Machine Sort", Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data, Minneapolis, MN, 1994.

[8] Nyberg, C., C. Koester, J. Gray, "Nsort: a Parallel Sorting Program for NUMA and SMP Machines", http://www.ordinal.com/NsortPara.pdf

[9] Zhang, X., L. Rivera, A. Chien, "HPVM MinuteSort", http://research.microsoft.com/barc/SortBenchmark/Y2000_MinuteSort.pdf

Eurologic is a registered trademark of Eurologic Systems.
Intel and Itanium are registered trademarks and trademark of Intel Corporation.
Microsoft, Windows Server 2003 and SQL Server 2000 are registered trademarks and trademark of Microsoft Corporation.
NEC and Express5800 are registered trademarks of NEC Corporation.
Ordinal and Nsort are trademarks of Ordinal Technology Corp.
Qlogic is the trademark of Qlogic Corporation.