A Minute of Mainframe Batch Sorting on Windows

Chris Nyberg <chris@ordinal.com>, Ordinal Technology Corp
Charles Koester <charles@koester.com>, Ordinal Technology Corp

Abstract

In February 2006, Fujitsu Computer Systems and Ordinal Technology set a new record for the Daytona MinuteSort benchmark using NeoSort, a sort program developed by Fujitsu to perform mainframe sorts on Windows and based on Ordinal's Nsort technology. The sort platform was a Fujitsu PRIMEQUEST computer running Windows 2003 with 4 Fujitsu ETERNUS storage subsystems and 128 disks. Using 59 seconds of elapsed time, NeoSort read a 40 gigabyte (400 million 100-byte records) input file at 2.6 GB/sec, sorted the records in memory, and wrote the sorted data to an output file at 1.2 GB/sec.

Introduction

Sorting is recognized as one of the most important computing tasks on mainframes, and is a well-studied research field[1]. In recent years though, Ordinal Technology's Nsort program has delivered the best commercial sort performance on Windows and Unix systems. Now Fujitsu Computer Systems has developed its NeoSort program utilizing Nsort technology. NeoSort, along with Fujitsu's NeoBatch system, allows JCL-described sorts to be quickly processed by combining the bandwidth of large numbers of commodity processors and disks.

To demonstrate the performance of NeoSort and NeoBatch, we were able to use a Fujitsu PrimeQuest system at the Fujitsu North American TRIOLE Integration Center in Sunnyvale, California. The system contained 32 1.6 Ghz Itanium 2 processors, 128 GB of main memory, 4 Eternus storage systems and 128 disks. The NeoSort program was able to sort 40 GB in 59 seconds - a new MinuteSort record in the Daytona (commercial sort program) category.

This paper presents some background on the MinuteSort benchmark, then describes the server hardware, system software and record-breaking sort run.

MinuteSort

MinuteSort is a sorting benchmark [2] that 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 (since 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 Daytona MinuteSort contest has been dominated by Nsort. The first three Nsort records were achieved on SGI Origin2000 systems. The most recent, 2004 record was achieved on an NEC Express5800/1320Xd running Windows.

Previous Daytona MinuteSort Records

Year	Sort Name	Size	Passes
1997	Nsort [3]	5.3 GB	One-Pass
1997	Nsort [3]	7.6 GB	Two-Pass
1999	Nsort [3]	12.0 GB	One-Pass
2004	Nsort [4]	36.0 GB	One-Pass

Hardware

The hardware used for the NeoSort benchmark consisted of the following, illustrated in Figure 1:

- 1 x PRIMEQUEST 480 Server
- 32 x Intel(R) Itanium2 1.6 GHz Processors
- Memory: 127 GB
- 16 x Emulex LP10000 (2Gb/s) FC Host Bus Adaptors
- 16 x Direct (FC-AL) connections; 4 for each connection between the ETERNUS3000s and the PRIMEQUEST
- 3 x ETERNUS3000 Model 700 (2Gb/s Host Interface)
- 1 x ETERNUS3000 Model 600 (2Gb/s Host Interface)
- 128 Disks in the Storage Array made up of
    - 50 x 73GB disks
    - 78 x 146GB disks
- 32 RAID0+1(4) RAID Groups
  (4 x Disks per Raid Group; 4GB LUN per RAID Group; 8 x RAID Groups per ETERNUS3000)

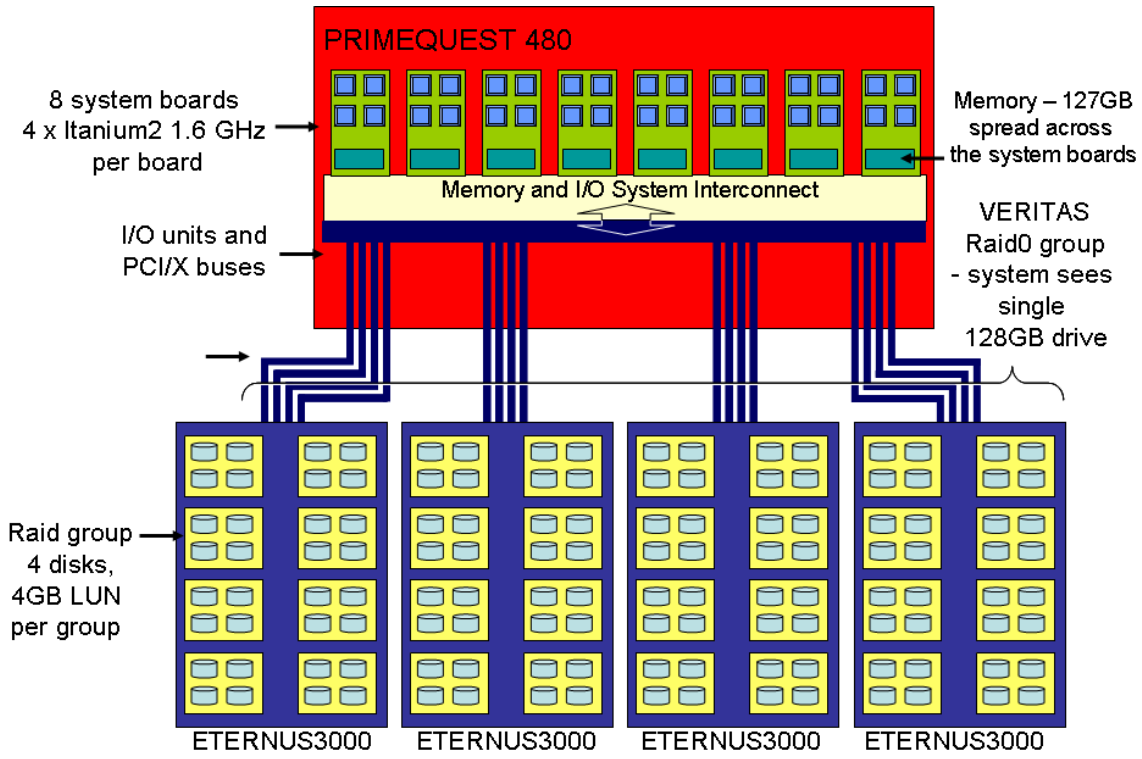

Figure 1. Diagram of Hardware Used for NeoSort Benchmark

System Software

The following system software was used on the PrimeQuest 480 server:

- Microsoft Windows Server 2003 - Data Center Edition for 64-bit Itanium Based Systems
- NTFS File System
- Veritas Volume Manager was used to stripe together the 32 RAID logical units (LUNs)
  - Capacity : 124.80 GB
  - Layout : Striped
  - StripeWidth : 512K
- NeoBatch, used to process the JCL that defined and invoked the sort operation

Running the Sorts

To run the MinuteSort benchmark, a 40GB file of 100-byte records was generated using random 10-byte keys on the striped logical volume (the F: drive). The input file F:\40g.dat was cataloged in the JCL system as BIG.40G. The output file F:\SORT\OUT.seq was cataloged as SORT.OUT. The following JCL was generated to run the sort:

```
//SORT1  JOB 62341,'K.Hollis',MSGCLASS=X,CLASS=C,
//       REGION=4M,NOTIFY=Administrator,RESTART=*
//*
//* Del files
//*
//DEL    EXEC PGM=IDCAMS
//SYSIN  DD *
  DELETE SORT.OUT
  SET MAXCC=0
//*
//SYSPRINT DD SYSOUT=*
//*
//* Sort the big file
//*
//STP1   EXEC PGM=TIMEX,PARM='SORT'
//SORTIN DD DSN=BIG.40G,DISP=SHR
//SORTOUT DD DSN=SORT.OUT,DISP=(,CATLG,DELETE),
//           DCB=(LRECL=100,RECFM=FB),VOL=SER=BIG
//SYSIN  DD *
  SORT FIELDS=(1,10,CH,A)
//SYSOUT  DD SYSOUT=*
```

Figure 2. JCL Used in NeoSort Benchmark

In our initial sort runs, we could not exceed a read rate of 1.4 GB/sec. On the other hand we found that i/o performance test programs could achieve a 2.6 GB/sec read rate with the same striped logical volume. The critical difference turned out to be that the i/o performance test programs reused their read buffers, whereas NeoSort always read its file input to newly allocated virtual memory. This required Windows, as part of its read request processing, to allocate the physical memory behind the virtual memory destinations. The standard physical memory allocation mechanism in Windows was limited to 1.4 GB/sec on the PrimeQuest system.

To get around this bottleneck, we used some high speed, multithreaded routines - AllocateUserPhysicalPages() and MapUserPhysicalPages(). With these routines we spent the first 4 seconds of the sort execution allocating the physical pages behind NeoSort's 45GB of process memory, but then were able to read the input file at 2.6 GB/sec. This reduced NeoSort's elapsed time by almost 10 seconds.

The NeoBatch output of one of the benchmark runs is show below in Figure 3. TIMEX is the test harness program used to invoke NeoSort and output the timing results. The lines marked with E, were output by the TIMEX program and provide:

- **ExitCode** – Zero indicates the operation terminated successfully
- **Elapsed Time** – The time from invoking the sort operation to receiving the return from the call
- **Kernel Time** - The amount of CPU time, across all processors, spent in the operating system (i.e. in Windows)
- **User Time** – Time spent executing the sort on all of the processors added together.

```
                        M S G L O G  --  S Y S T E M   F C S 2 - P 0
  JOB NAME: SORT1
  JOB ID:   34
  JCL: c:\NeoBatch\ProductDir\SYSOUT\Administrator\SORT1\34\JCL.jcl
  JS:  c:\NeoBatch\ProductDir\SYSOUT\Administrator\SORT1\34\JSCRIPT.js
 ------------------------------------------------------------------------------
 00034: Starting step DEL.
 00034: Executable: c:\neobatch\productdir\IDCAMS.EXE
 00034:    DELETED: (SYSIN)SYS06038.T133934.RA000034.SORT1.R0100001
 00034: Completed step DEL, RC=0
 00034: Starting step STP1.
 00034: Number of data directories is 0
 00034: Executable: c:\neobatch\productdir\TIMEX.EXE
 00034E: ExitCode: 0
 00034E: Elapsed Time:     58.790
 00034E: Kernel Time :     18.300
 00034E: User Time   :   1296.930
 00034:    DELETED: (SYSIN)SYS06038.T133937.RA000034.SORT1.R0100003
 00034:    KEPT:    (SORTIN)BIG.40G
 00034:    CATLGD:  (SORTOUT)SORT.OUT
 00034: Completed step STP1, RC=0
Job 34 Completed.  Job exit code: 0
```

Figure 3. NeoBatch message log

The elapsed time in the above message log indicates the NeoSort invocation took 58.79 seconds of elapsed time to sort the 40 GB input file. This is a new MinuteSort record.

The NeoSort output is shown below. It has been color-coded in this paper for descriptive purposes.

```
SORT: EXEC: Fujitsu NeoBatch Sort Copyright (c) 2005-2006 Fujitsu Software
Corporation
 SORT: EXEC: Parsing sort arguments
 SORT: INFO: SORT CARDS PARSED:
 SORT: INFO:   SORT FIELDS=(1,10,CH,A)

 SORT: EXEC: Beginning sort execution
 SORTIN: f:\40G.dat
 SORTOUT: F:SORT\OUT.seq
 Using Nsort for file i/o
 Nsort commandline: -format = size:100  -key = offset:0, size:10, ascending,
char -mem=90g -proc=30 -stat -touch
-in_file=f:\40G.dat,direct,tr=32m,count=20
-out=F:SORT\OUT.seq,direct,tr=32m,count=20
 Sort/Merge statistics:
 Nsort version 3.3.11 (Windows-IA64 64-bit) using 45G of memory out of 90G
 Pointer sort performed Tue Feb 07 13:39:38 2006
       Input Phase        Output Phase          Overall
 Elapsed    19.26             37.07              56.33
 I/O Busy   15.04   100%      27.91    87%       42.95
 Action  User   Sys Busy    User   Sys Busy    User   Sys Busy
 main    0.07  9.62  50%    0.15  2.21   6%    0.22 11.83  21%
   1    15.17  0.03  79%   30.63  0.03  83%   45.80  0.06  81%
   2    11.88  0.07  62%   31.52  0.03  85%   43.40  0.10  77%
   3    12.36  0.11  65%   28.88  0.04  78%   41.24  0.15  73%
   4    14.45  0.03  75%   29.10  0.04  79%   43.55  0.07  77%
   5    11.02  0.05  57%   29.62  0.06  80%   40.64  0.11  72%
   6    10.81  0.09  57%   31.88  0.01  86%   42.69  0.10  76%
   7    11.07  0.06  58%   29.24  0.01  79%   40.31  0.07  72%
   8    13.20  0.09  69%   29.04  0.06  79%   42.24  0.15  75%
   9    13.46  0.05  70%   29.42  0.03  79%   42.88  0.08  76%
  10    13.05  0.06  68%   29.54  0.07  80%   42.59  0.13  76%
  11    12.88  0.07  67%   30.68  0.04  83%   43.56  0.11  78%
  12    15.86  0.06  83%   29.88  0.04  81%   45.74  0.10  81%
  13    12.50  0.06  65%   31.91  0.06  86%   44.41  0.12  79%
  14    12.65  0.06  66%   31.14  0.04  84%   43.79  0.10  78%
  15    11.45  0.06  60%   30.32  0.00  82%   41.77  0.06  74%
  16    14.22  0.10  74%   29.78  0.02  80%   44.00  0.12  78%
  17    13.33  0.03  69%   29.32  0.04  79%   42.65  0.07  76%
  18    11.20  0.10  59%   31.83  0.07  86%   43.03  0.17  77%
  19    14.48  0.04  75%   30.62  0.03  83%   45.10  0.07  80%
  20    11.71  0.05  61%   28.98  0.01  78%   40.69  0.06  72%
  21    11.62  0.05  61%   31.17  0.07  84%   42.79  0.12  76%
  22    13.08  0.01  68%   32.25  0.04  87%   45.33  0.05  81%
  23    12.53  0.03  65%   30.33  0.01  82%   42.86  0.04  76%
  24    13.76  0.05  72%   30.57  0.06  83%   44.33  0.11  79%
  25    11.01  0.06  57%   28.95  0.03  78%   39.96  0.09  71%
  26    14.52  0.02  75%   29.78  0.02  80%   44.30  0.04  79%
  27    12.66  0.04  66%   31.67  0.05  86%   44.33  0.09  79%
  28    12.92  0.06  67%   30.85  0.05  83%   43.77  0.11  78%
  29    12.46  0.06  65%   32.49  0.02  88%   44.95  0.08  80%
  30    13.42  0.03  70%   29.16  0.07  79%   42.58  0.10  76%
 All   384.80 11.30 2057% 910.70  3.36 2466%   1295 14.66 2326%
     Majflt    Minflt  Sort Procs Aio Procs/QueueSize RegionKB
      0/0           0          30         0/0              512
 File Name          I/O Mode Busy   Wait MB/sec  Xfers        Bytes Records
 Input Reads
   f:\40G.dat       direct   100%   7.81   2587   1193  40000000000 400000000
 Output Writes
   F:SORT\OUT.seq   direct    87%   0.00   1221   1193  40000000000 400000000
```

Figure 4. NeoSort output

The NeoSort output includes the Nsort command passed to the Nsort API, and the Nsort-reported performance statistics. The statistics include the cpu use times (both in user mode and system mode) for the

main Nsort thread and its 30 sort worker threads, these are divided into the input phase (reading the input file and generating runs of records in memory) and the output phase (merging the internal runs and writing the resulting records to the output file), and the overall combination of the times for the two phases. The file statistics indicate the input file was read at 2.6 GB/sec, and the output file was written at 1.2 GB/sec.

Conclusion

Mainframe batch sorts can now be run on commodity-based, high-performance Windows systems at record-breaking speeds.

Acknowledgments

We would like to thank numerous people at Fujitsu Computer Systems for their help. Ron Langer and Andrew Mackenzie initiated and sponsored this benchmark attempt. Kelly Hollis and Basim Kadhim wrote the sort JCL, helped identify some high-speed Windows memory allocation routines that were critical to achieving NeoSort's input file read rate, and ran the final NeoSort runs. Gene Owens provided access to the PrimeQuest server. Gene's team of Rudy Thomas, Greg Rodoni, Kun Katsumata, Jim Repinski and Al Zmyslowski set up the PrimeQuest and Eternus configuration. Sandy Wilson helped us analyze some early i/o traces which led to the identification of the physical memory allocation bottleneck. John Andoh served as our interface to the rest of Gene's team.

References

[1] Knuth, D.E., The Art of Computer Programming, Vol. 3, Addison-Wesley, Reading, MA, 1973.
[2] 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.
[3] Nyberg, C., C. Koester, J. Gray, "Nsort: a Parallel Sorting Program for NUMA and SMP Machines", http://www.ordinal.com/NsortPara.pdf, 2000.
[4] Nyberg, C., J. Gray, C. Koester, "A Minute with Nsort on a 32P NEC Windows Itanium2 Server", http://www.ordinal.com/NsortMinute.pdf, 2004.