Parallel I/O Performance Characterization of Columbia and NEC SX-8 Superclusters

Subhash Saini,[1] Dale Talcott [1] Rajeev Thakur,[2]
Panagiotis Adamidis,[3] Rolf Rabenseifner [4]
and Robert Ciotti [1]

Abstract: *Many scientific applications running on today's supercomputers deal with increasingly large data sets and are correspondingly bottlenecked by the time it takes to read or write the data from/to the file system. We therefore undertook a study to characterize the parallel I/O performance of two of today's leading parallel supercomputers: the Columbia system at NASA Ames Research Center and the NEC SX-8 supercluster at the University of Stuttgart, Germany. On both systems, we ran a total of seven parallel I/O benchmarks, comprising five low-level benchmarks: (i) IO_Bench, (ii) MPI Tile IO, (iii) IOR (POSIX and MPI-IO), (iv) b_eff_io (five different patterns), and (v) SPIOBENCH, and two scalable synthetic compact application (SSCA) benchmarks: (a) HPCS (High Productivity Computing Systems) SSCA #3 and (b) FLASH IO (parallel HDF5). We present the results of these experiments characterizing the parallel I/O performance of these two systems.*

1. Introduction

Scientific and engineering applications of national interest are in general becoming more and more data intensive. These applications include simulations of scientific phenomena on large-scale parallel computing systems in disciplines such as NASA's data assimilation, astrophysics, computational biology, climate, combustion, fusion, high-energy physics, nuclear physics, and nanotechnology. Furthermore, experiments are being conducted on scientific instruments, such as particle accelerators, that generate terabytes of data [1]. For many such applications, the challenge of dealing with data, both in terms of speed of data access and management of the data, already exceeds the challenge of raw compute power. Therefore, it is critical for today's and next-generation supercomputers not only to be balanced with respect to the compute processor, memory, and interconnect, but I/O performance needs to be significantly increased. It is not just the number of teraflops/sec that matters, but how many gigabytes/sec or terabytes/sec of data can applications really move in and out of disks that will affect whether these computing systems can be used productively for new scientific discoveries [1-2].

To get a better understanding of how the I/O systems of two of the leading supercomputers of today perform, we undertook a study to benchmark the parallel I/O performance of NASA's Columbia supercomputer located at NASA Ames Research Center and the NEC SX-8 supercluster located at University of Stuttgart, Germany.

The rest of this paper is organized as follows. In Section 2, we present the architectural details of the two machines and their file systems. In Section 3, we describe in detail the various parallel I/O benchmarks used in this study. In Section 4, we present and analyze the results of the benchmarking study. We conclude in Section 5 with a discussion of future work.

2. Architectural Details

We describe the processor details, cluster configuration, memory subsystem, interconnect, and file systems of Columbia and NEC SX-8 [3].

2.1 Columbia: The Columbia system at the NASA Advanced Supercomputing (NAS) facility is a cluster of twenty SGI Altix systems, each with 512 processors and 1 TB of shared-access memory. Four of these systems are connected into a supercluster of 2048 processors. Of the 20 systems, twelve are model 3700s, and the other eight are BX2s, which are essentially double-density versions of the 3700s in terms of both the number of processors in a compute brick and the interconnect bandwidth. The results reported here are from runs on the BX2s.

The computational unit of the SGI Altix BX2 system consists of eight Intel Itanium 2 processors, with a memory capacity of 16 GB, and four application specific integrated circuits (ASICs) called SHUBs (Super Hubs). The processor is 64-bit, runs at 1.6 GHz, and has a peak performance of 6.4 Gflops/s.

I/O adapters in the BX2 reside in IX-bricks separate from the C-bricks. As configured for Columbia, an IX-brick holds up to five PCI-X adapters. The BX2 systems we tested have four IX-bricks each. Each IX-brick connects through one or two 1200 MB/s links to a SHUB in a C-brick. This connection allows any I/O card to perform DMA to any part of the global shared memory. Note that this DMA I/O shares NUMALink

[1] NASA Advanced Supercomputing Division, NASA Ames Research Center, Moffett Field, CA 94035-1000, USA, {ssaini, dtalcott, ciotti}@mail.arc.nasa.gov
[2] Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439, USA, thakur@mcs.anl.gov
[3] German Climate Computing Center, Hamburg, Germany, adamidis@dkrz.de
[4] High Performance Computing Center, University of Stuttgart, Nobelstrasse 19, D-70569 Stuttgart, Germany, rabenseifner@hlrs.de

1-4244-0910-1/07/$20.00 ©2007 IEEE

bandwidth with off-brick memory accesses by processors in the brick.

2.2 NEC SX-8: The processor used in the NEC SX-8 is a proprietary vector processor with a peak vector performance of 16 Gflops/s and an absolute peak of 22 Gflops/s if one includes the additional divide & sqrt pipeline and the scalar units. It has 64 GB/s memory bandwidth per processor and eight vector processors per node [3]. HLRS in Stuttgart, Germany, has recently installed a cluster of 72 NEC SX-8 nodes, with a total of 576 processors. The front-end to this cluster is a scalar system, called TX-7, with 16 Itanium 2 processors and a very large memory of 256 GB. The front-end and the back-end NEC SX-8 machines share fast file systems. A total of 16 1-TB file systems are used for user home directories. Another 16 1-TB file systems contain workspace, which can be used by jobs at run time. Each file system can sustain 400-600 MB/s throughputs for large block I/O.

Each of the 72 NEC SX-8 vector nodes has 128 GB of memory, about 124 GB of which is usable for applications.

2.3 File Systems: In this section, we describe the file systems on the Columbia and NEC SX-8 superclusters.

2.3.1 File System on Columbia: In the past, the 20 Altix machines of Columbia accessed a shared Network File System (NFS) containing the users' home directories. Due to the relatively poor performance of NFS file systems, each of the machines also had a local XFS-based scratch disk (/nobackup*i*, where i=1, 2, 3, …, 20), and users used these scratch disks for their performance-sensitive I/O. This configuration was not conducive to efficient use of the Columbia system. For example, if a user of host Columbia5 wanted to run an application on Columbia9, he had to ensure that files accessed by his application on /nobackup5 also existed on /nobackup9. In addition, the design of the NFS file system is to provide distributed access to files from multiple hosts, and its consistency semantics and caching behavior are accordingly designed for such access. A typical scientific-computing workload does not mesh well with the semantics of NFS, especially for concurrent writes. Therefore, in February 2006, the Columbia system was reconfigured to take advantage of SGI's Clustered XFS (CXFS) technology, which overcomes the problems associated with NFS and permits a more efficient shared file system.

With CXFS, metadata about files is still managed by shared servers, but each host has direct access via Fibre Channel to the file data disks. Currently, CXFS with file sharing is available on all 20 hosts as shown in Figure 1. In the systems under test (nodes C17-20), each host communicates with the three sets of three metadata servers via gigabit ethernet. The file-system data blocks are accessed across four, 2 Gb/s, Fibre Channel connections to dual RAID controllers, each with 2.5 GB of cache, interfacing with 30 TB of disk space striped across 8 LUNs of 8+1 RAID-3 [3].

2.3.2 File System on NEC SX-8 Cluster: The file system on the NEC SX-8 cluster, called GStorageFS, is also based on the XFS file system. It is a SAN-based (Storage Area Network) file system that takes advantage of a Fibre Channel infrastructure. The client does data transfers after negotiation with a metadata server. This metadata server is in fact an enhanced NFS3 (Network File System) server, which transfers lists of disk blocks to the client using a third-party-transfer scheme.

Using the conventional NFS3 protocol does small I/O and transactions such as creation of files. Using direct client-to-disk I/O performs large data transfer. The file system on the NEC SX-8 cluster is schematically shown in Figure 2. The file system of the NEC SX-8 cluster consists of 72 S1230 RAID-3 disks. Each RAID has 4 logical units (LUNS) consisting of 8 (+ 1 parity) disks. The NEC SX-8 nodes and the file server are connected to the disks via four Fibre Channel switches with a peak transfer rate of 2 Gb/s per port.

The tested 80 TB file system uses half of the disk resources, namely, 36 S1230 units with 72 controllers. The logical view of the file system on the SX-8 cluster is shown in Figure 2. The disks are organized in 18 stripes, each consisting of 8 LUNs. The bandwidth of one LUN is about 100-140 MB/s.

A file is created in one stripe, with the location depending on the host creating the file. The bandwidth to access a single file depends on the number of stripes it spans, which is usually one. High aggregate performance can be achieved when multiple nodes access multiple files. Figure 3 shows the assignment of the SX-8 nodes to the stripes. A consequence of this mapping is that if several nodes access the same stripe, they will not get the best performance. To achieve best performance, it is important to use nodes that are not mapped to the same stripe. Since the striping size is 512 KB, the first block size that makes optimal use of the 8-fold stripe is 4 MB. Larger block sizes increase the efficiency of striping and of access to individual LUNs.

2.4 Architectural Differences: NEC's GStorageFS has a server-centric architecture. The server on behalf

of the clients does all metadata updates. In contrast, SGI's CXFS has a token-based delegation mechanism. A client applies to the server for a token, which allows the client to update the metadata itself. For the holder of the token, CXFS is more or less a local file system. The token might remain on the client until another client asks for the token, and the server therefore revokes the token. The differences between Columbia and NEC SX-8 systems are summarized in Table 1. The Fibre Channel limits are calculated with an assumption of 200 MB/s payload on a 2 Gb/s port.

3. Parallel I/O Benchmarks: To characterize the performance of Columbia and NEC SX-8 superclusters, we used a total of seven I/O benchmarks, which are described below.

3.1 Low-level I/O Benchmarks: We used the following five low-level parallel I/O benchmarks.

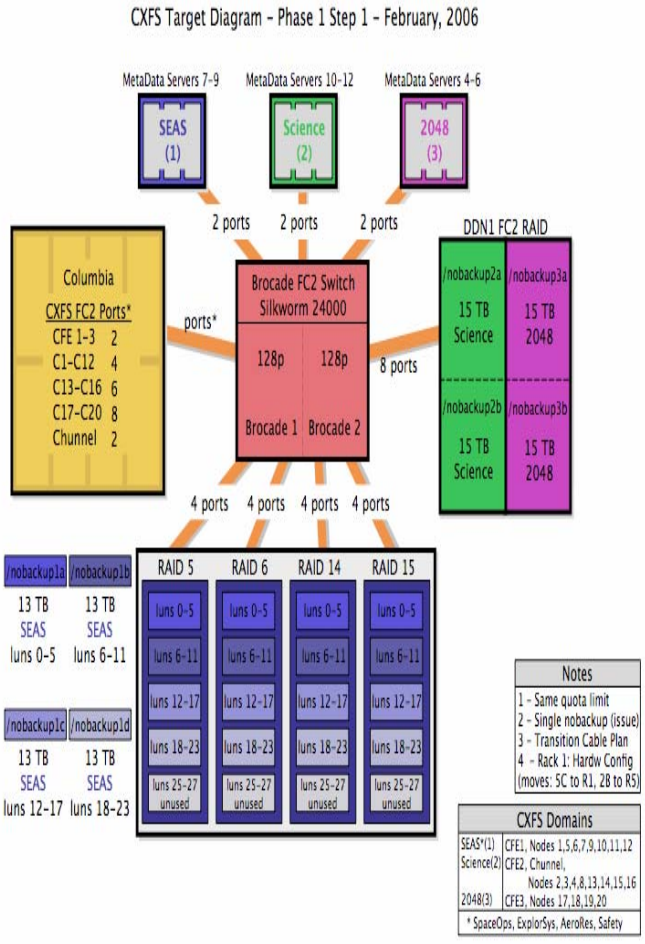
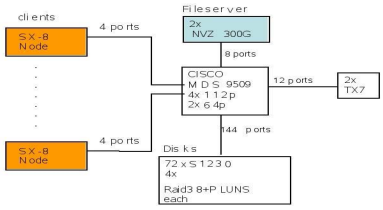


Figure 1: CXFS file-system configuration of the Columbia system at NASA Ames Research Center, USA.



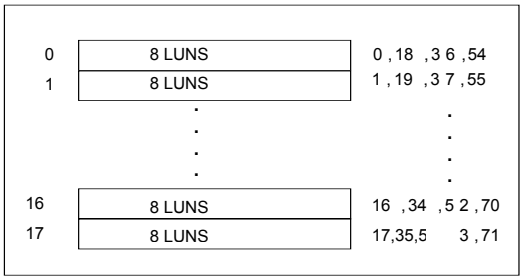


Figure 2: GStorage file-system configuration of the NEC SX-8 cluster at the University of Stuttgart, Germany.

Figure 3: Logical view of the file system on the SX-8 cluster and location of a file created by SX-8 nodes (0–71).

3.1.1 IO_Bench Benchmark: The IO_Bench benchmark is based on the High Performance Computing Modernization Office (HPCMO) Instrumental IO_Bench benchmark version 2 [4]. The benchmark measures the rate at which a computing system performs read/write to an arbitrary state of disk. The tests are designed to mimic the I/O requirements for a given shared-resource computer center's I/O workloads.

The benchmark runs a series of sequential, backward, and random tests. In particular, it measures the performance of sequential write, backward write, sequential read, random read, backward read, and random read/write. The inputs to the benchmarks are the sizes of the file and the I/O buffer. The outputs are real time, user time, and system time in seconds. The bandwidth is calculated as Bandwidth (MB/s) = File Size (MB) / Real Time (s) [4].

3.1.2 MPI Tile I/O Benchmark: This benchmark tests the performance of underlying MPI-IO library and file-system implementation under a non-contiguous access workload [5-7]. The benchmark logically divides a data file into a dense two-dimensional set of tiles. One needs to specify the number of tiles along rows and columns, along with the number of elements in the rows and columns.

3.1.3 b_eff_io Benchmark: The parallel b_eff_io benchmark measures the performance of three access methods and five pattern types [8]. The three access methods are: (a) initial write, (b) rewrite, and (c) read.

Table 1: Summary of technical differences between Columbia and NEC SX-8 systems.

Characteristic	Columbia	NEC SX-8
System size	20 nodes	72 nodes
# Processors per node	512	8
Total number of processors	10,240	576
Processor vendor	Intel	NEC
Processor model	Itanium 2	Proprietary
Processor type	Scalar	Vector
Clock speed (GHz)	1.6	2.0
Peak performance (Tflop/s)	65. 536	9.216
Node type	Shared memory	Shared memory
Memory per processor	2 GB	16 GB
CPUs used for I/O benchmarks	512 processors	512 processors
Network	NUMALink4	IXS
Network topology	Fat-tree	Multi-stage crossbar
Operating Ssystem	Linux	Super-UX
Node name tested for I/O	Columbia 20	V04 – V67
System vendor	SGI	NEC
File system type	CXFS	GStorageFS
File directory name	/nobackup3b	/nfs/nas/scr
File system size	30 TB	80 TB
FC2 ports at node	8 ports per node	4 ports per node
FC2 ports at disks	8	72
I/O FC limit per node	1.6 GB/s	0.8 GB/s
I/O FC limit total	1.6 GB/s	14.4 GB/s
Location	NASA - Ames, USA	HLRS, Germany

The five types of I/O patterns measured are: (0) strided collective access, scattering large chunks in memory to/from disk; (1) strided collective access with one read/write call per disk chunk; (2) non-collective access to one file per MPI process, that is, to separate files; (3) same as (2) except that the individual files are assembled to one segmented file; (4) same as (3) except that the access to the segmented file is done collectively.

3.1.4 IOR Benchmark: Interleaved Or Random (IOR) is a parallel file system benchmark developed by the SIOP (Scalable I/O) project at Lawrence Livermore National Laboratory (LLNL) [9]. The data are written and read using independent parallel transfers of equal-sized blocks of contiguous bytes that cover the file with no gaps and that do not overlap each other. The benchmark runs in three API modes for I/O: POSIX, MPI-IO, and HDF5.

3.1.5 SPIOBENCH Benchmark: The Scalable Parallel IO Benchmark (SPIOBENCH) tests the scalability of parallel I/O [10]. It measures the ability of the system to transfer data to/from the shared file system. The aggregate I/O done is 128 GB. By NSF rules, the benchmark must be run in its entirety [10]. All files associated with SPIOBENCH must be located on a shared file system at run time, and the benchmark itself must be executed from that shared file system.

3.2 Compact Applications: We also used one scalable synthetic compact application (SSCA) and one realistic compact application, which are described below.

3.2.1 HPCS SSCA #3 Benchmark: Scalable Synthetic Compact Application (SSCA) #3 benchmark is one of the benchmarks accepted by DARPA's HPCS productivity team to benchmark the next generation of petaflops-class computing systems for U.S. government procurement [11]. The SSCA #3 benchmark stresses computation, communication, and data I/O

3.2.2 Flash I/O Benchmark: This benchmark mimics the I/O in a block-structured adaptive mesh refinement (AMR) hydrodynamics code that solves compressible, reactive hydrodynamics equations and characterizes the physics and mathematical algorithms used in studying nuclear flashes on neutron stars and white dwarfs [12]. The benchmark uses the parallel HDF5 library. It produces three output files: (a) a checkpoint file, (b) a plot file for centered data, and (c) a plot file for corner data.

4. Results and Analysis:

In this section, we present the results of running all five low-level I/O benchmarks and the two SSCA benchmarks on Columbia and NEC SX-8 systems.

4.1 Low-level I/O Benchmarks: We present the results for the five low-level parallel I/O benchmarks.

4.1.1 IO_Bench Benchmark: Figure 4 shows the results for six I/O operations for block size of 16 MB: (i) SW-sequential write, (ii) SR-sequential read, (iii) RW-random write, (iv) RR-random read, (v) BW-backward write, and (vi) BR-backward read. Such high bandwidths for a single processor can be attributed to caching and prefetching performed by the file system. For all the six I/O operations, the bandwidth of Columbia is about 40-60% better than that of NEC SX-8. The differences in maximum performance correlate with the number of Fibre Channel ports used in this experiment: 8 on Columbia but only 4 on SX-8. On both systems, reads are about 40-50% faster than writes. Also, on both systems, backward I/O operation is slower than the normal (forward) operation.

Figure 5 shows the results for a block size of 128 MB. Although Columbia uses 8 Fibre Channel ports compared to 4 on SX-8, the write bandwidth on SX-8 in two cases (sequential and backward) is better than

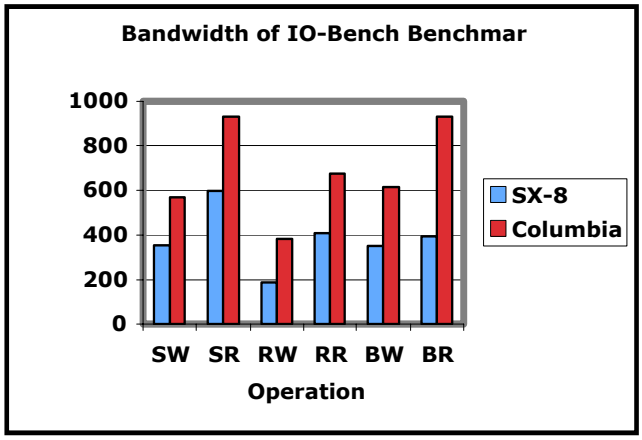


Figure 4: IO_Bench benchmark bandwidth (in MB/s) for six I/O operations on a single processor for a block size of 16 MB on Columbia and NEC SX-8.

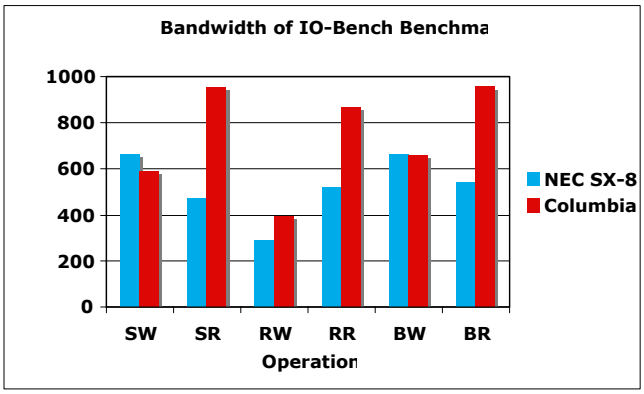


Figure 5: IO_Bench benchmark bandwidth (in MB/s) for six I/O operations on a single processor for a block size of 128 MB on Columbia and NEC SX-8.

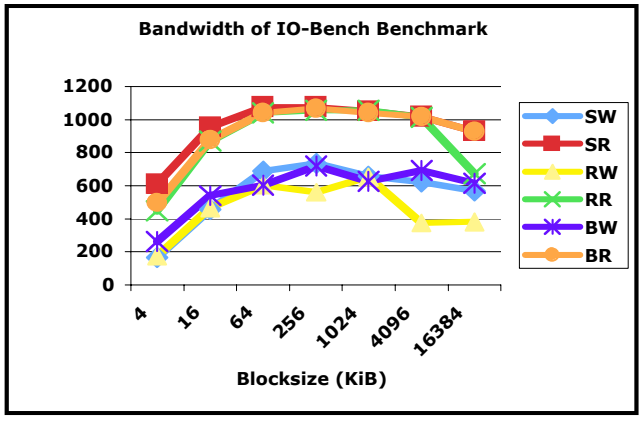


Figure 6: Bandwidth (in MB/s) for six I/O operations on a single processor of Columbia for various block sizes.

on Columbia. On Columbia, the read (write) performance is 54-60% (25-41%) of the peak Fibre Channel bandwidth (1.6 GB/s) on one node, and on the SX-8, the read (write) performance is 59-67% (36-83%) of the peak bandwidth of 0.8 GB/s.

Figure 6 shows the results for the IO_Bench benchmark on a single processor of Columbia for block sizes ranging from 4 KB to 16 MB**.** It is clear that the performance of all three read operations is almost double that of the three write operations. The performance of all six operations is good for block sizes ranging from 256 KB to 4 MB. For RR and RW, the performance drops for block sizes of 4 MB and 1 MB respectively. For all six operations, the performance falls for block sizes of 8 MB and higher. These drops can be attributed to caching and prefetching not working well in these circumstances.

4.1.2 MPI Tile I/O Benchmark: Figure 7 shows the read and write bandwidths with the MPI Tile I/O benchmark on Columbia. In this case, the size of the file was kept constant and the number of processors was varied.

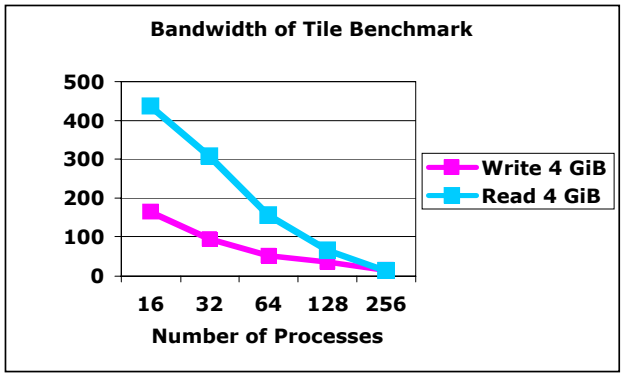


Figure 7: Read and write bandwidth (in MB/s) using MPI Tile I/O benchmark for a file of 4 GB on Columbia for a range of processors from 16 to 256.

To start with, at 16 processors, the read bandwidth is about 450 MB/s, whereas the write bandwidth is about 180 MB/s. As the number of processors increase, both read and write bandwidths decrease gradually until both reach the same value of about 20 MB/s.

4.1.3 b_eff_io Benchmark: Table 2 shows the values of b_eff_io bandwidth in MB/s on both Columbia and NEC SX-8. This number summarizes the performance of all five types of I/O patterns with different chunk sizes. The flags used for running the b_eff_io benchmark are:

```
mpirun -np 8 ./b_eff_io MB 1536
-MT 12288 -noshared -rewrite -N 8
-T 1800 -f outputile_for_8_PEs
```

where MT is the number of megabytes of memory in the total system. This value is used to compute the ratio of transferred bytes to the size of the total memory. For example, when running on 8 processors, it is 8 times the memory per processor (for SX-8, it is 8 x 1536 MB); on 16 processors, it is 16 times the memory per processor, and so forth. The option "no shared" is used to substitute the shared file pointer by individual file pointers in pattern type 1. The option "rewrite" does rewrites between write and read for all patterns. The b_eff_io benchmark has a time limit of 1800 sec, which is the minimum time b_eff_io has to be run. This time limit guarantees that real disk I/O is measured instead of caching.

b_eff_io is an effective bandwidth of the system and is calculated by giving a certain weight to a type of a pattern. For example, on Columbia for 16 processors, the weighted average bandwidth for write is

Table 2: Values of b_eff_io benchmark for various numbers of processors on Columbia and NEC SX-8.

CPUs	b_eff_io (MB/s)	
	Columbia	NEC SX-8
8	237.4	95.2
16	200.6	99.1
32	196.7	119.9
64	162.6	185.9
256	n/a	395.11

171.3 MB/s, the weighted average bandwidth for rewrite is 166.6 MB/s, and the weighted average bandwidth for read is 232.9 MB/s with pattern type 0 weighted two times. For 16 processors, the total amount of data written/read with each access method is 270104.9 MB, which is 1099.1 percent of the total memory (24576 MB). On 16 processors, with memory of 1536 MB per processor, b_eff_io is 200.6 MB/s.

The b_eff_io value, as an average over several block sizes, is highly influenced by small I/O. Therefore, in Figures 8-10, the bandwidth of three patterns with a large block size is presented. Figure 8 shows the results for pattern type 0 in which data is scattered from the processes to one common file. Figure 9 shows the results for pattern type 2 in which each process writes/reads to/from individual files. Figure 10 shows the results for pattern type 4 in which processes access a common file collectively.

On the SX-8, accessing one common file can use only 4 Fibre Channel ports and, therefore, the bandwidth is limited independently from the number of processors. With pattern type 2 (accessing individual files), up to 72 ports can be used. Columbia is restricted to 8 ports in all experiments. Figure 8 shows that less than 30% of the peak bandwidth is achieved on both platforms. In Figure 9, one can see the expected scaling on SX-8 for multiple files, and, as expected, the nearly constant performance on Columbia. Figure 10 shows on both platforms the expected independence from the number of processors. While read and rewrite performance on SX-8 is 65-90% of peak, the write performance on 64 processors is only 34% of peak. On Columbia, read performance is 9-36% and write/rewrite performance is 11-12% of the Fibre Channel bandwidth. The maximum block size was chosen according b_eff_io rules: 1/128 of available memory per processor (but at least 2 MB).

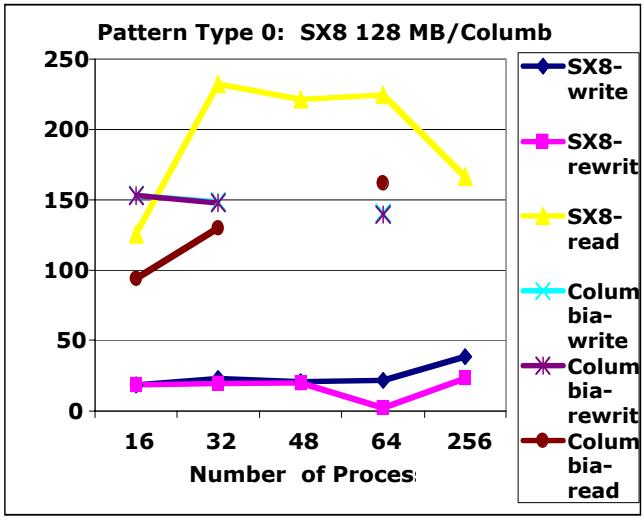


Figure 8: Bandwidth (in MB/s) for b_eff_io pattern type 0 on Columbia and SX-8 for various numbers of processors.

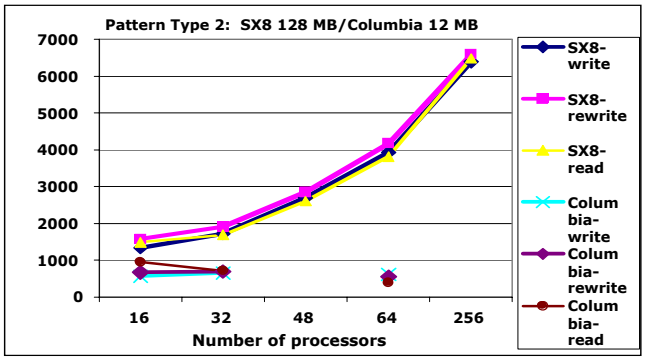


Figure 9: Bandwidth (in MB/s) for b_eff_io pattern type 2 on Columbia and SX-8 for various numbers of processors.

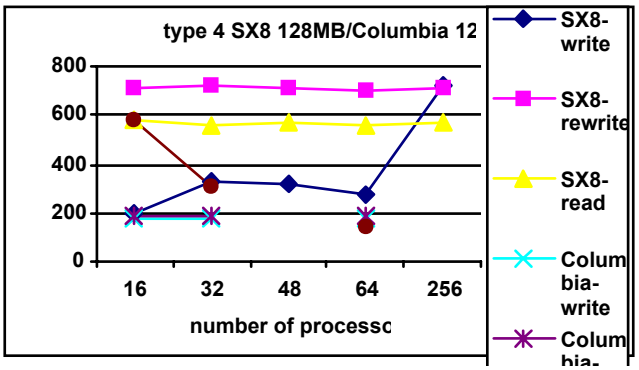


Figure 10: Bandwidth (in MB/s) for b_eff_io pattern type 4 on Columbia and SX-8 for various numbers of processors.

4.1.4 IOR Benchmark: In the present study, we used IOR version 2.8.10. Versions prior to release 2.8 provided data size and rates in powers of two. For example, 1 MB/s referred to 1,048,576 bytes per second. With the IOR release 2.8 and later versions, MB is now defined as 1,000,000 bytes and MiB is 1,048,576 bytes. The IOR benchmark runs in three API modes: MPI-IO, POSIX, and HDF5. The benchmark has two programming languages, C and Python. We used the C version.

In this section, we present results for MPI-IO and POSIX modes on Columbia for a case where a single file is opened for each process. Since the HDF5 version does not work in this mode, we ran the HDF5 version for a case in which a single file is accessed by all processors. We found that the benchmark just hung. Therefore, we do not present results with HDF5. Each test was repeated eight times, and the average is reported here. We present the results only for Columbia.

Figure 11 shows the read and write bandwidths for the MPI-IO mode for block sizes ranging from 4 KiB to 16 MiB for 256 and 476 processors of Columbia. We notice that read bandwidths are very high because of caching and read-ahead prefetching. For both 256 and 476 processors, the read performance is very high for block size of 1 MiB and 2 MiB. However, performance for write is very low on 256 and 476 processors.

In Figures 12 we present the results for the POSIX version of the benchmark. The results are similar to those for the MPI-IO version.

4.1.5 SPIO Benchmark: Figure 13 shows the performance of SPIO on Columbia with up to 508 processors. In SPIO, each process accesses a separate file, so the number of files read or written increases with the number of processes. The figures report the average of the bandwidth from all processes. The experiments are in the following sequence: initial write (Wrt0), read (Rd1), rewrite (reWr1), read (Rd2), rewrite (reWr2), read (Rd3), and rewrite (reWr3). In this benchmark, the total amount of I/O is constant, and the amount of data accessed per file decreases as

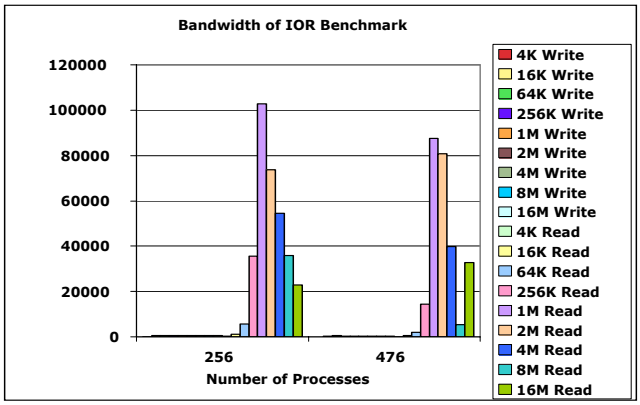


Figure 11: Read/write bandwidths (in MiB/s) with IOR MPI-IO benchmark for 256 and 476 processors on Columbia for various block sizes ranging from 4 KiB to 16 MiB.

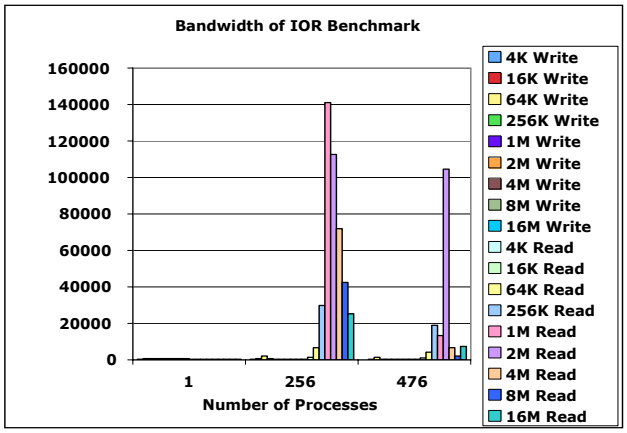


Figure 12: Read/write bandwidths (in MiB/s) with IOR POSIX benchmark for 256 and 476 processors on Columbia for various block sizes ranging from 4 KiB to 16 MiB.

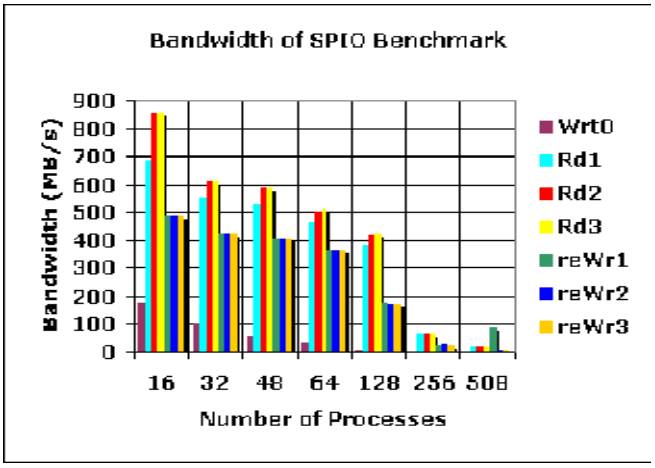


Figure 13: Bandwidth per process (in MB/s) as a function of number of processors for read and write operations using SPIOBENCH on Columbia.

the number of processors increases. As a result, we see that the performance drops as the number of processors increases. On up to 48 processors, the read performance is quite good (500-850 MB/s) because the read-ahead prefetching and caching done by the file system work favorably. The rewrite performance, however, is consistently lower than the read performance, typically by about 30-50%. Between 16 and 64 processors, the initial write bandwidth is 10-37% of the write bandwidth; with 128-512 processors it decreases to about 4%. Both read and rewrite bandwidths are substantially lower on 256 and 508 processors—well below 100 MB/s.

Figure 14 shows the results of SPIO on the NEC SX-8 cluster, on up to 512 processors, using 128 MB block size. This block size implies that direct disk I/O is performed. There is less than 4% difference between

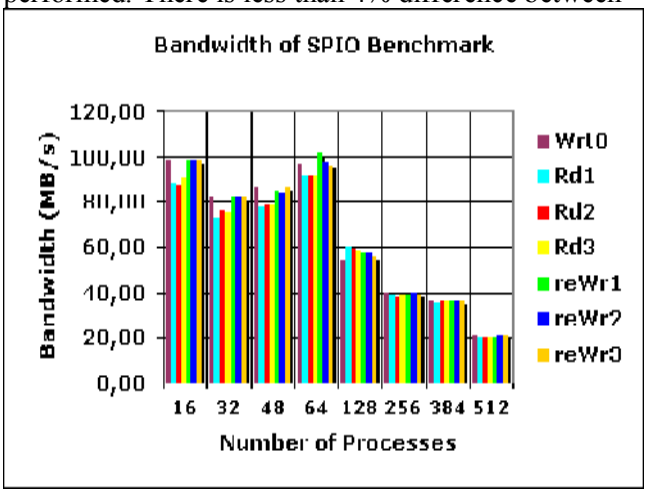


Figure 14: Bandwidth per process (in MB/s) as a function of number of processors for read and write operations using SPIOBENCH on NEC SX-8.

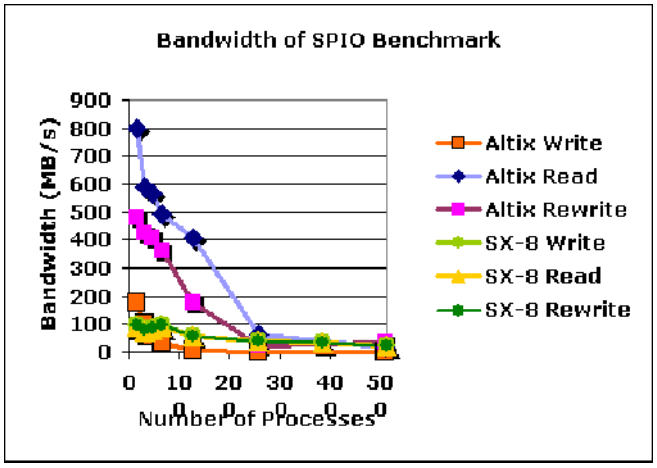


Figure 15: Read and write bandwidths (in MB/s) for various number processors on Columbia and NEC SX-8. Aggregate data accessed is always 128 GB for any number of processors.

initial write and rewrite.

Figure 15 shows a comparison of the performance of Columbia and SX-8. On Columbia, the read bandwidth begins very high (800 MB/s) on 16 processors and then drops sharply as we increase the number of processors to 256. On the other hand, on the SX-8, there is a much smaller variation, beginning at 100 MB/s on 16 processors and ending at 20 MB/s for 512 processors. The performance of initial writes on the SX-8 begins at 98 MB/s, remains nearly constant up to 64 processors, and then decreases to 21 MB/s on 512 processors. On Columbia, 16 processors can write at 178 MB/s, but on 64 processors, the bandwidth drops to 35 MB/s, and to 1 MB/s on 508 processors. The read/rewrite bandwidth on Columbia is high because data is being read/written from/to the file-system cache and not disk. On larger numbers of

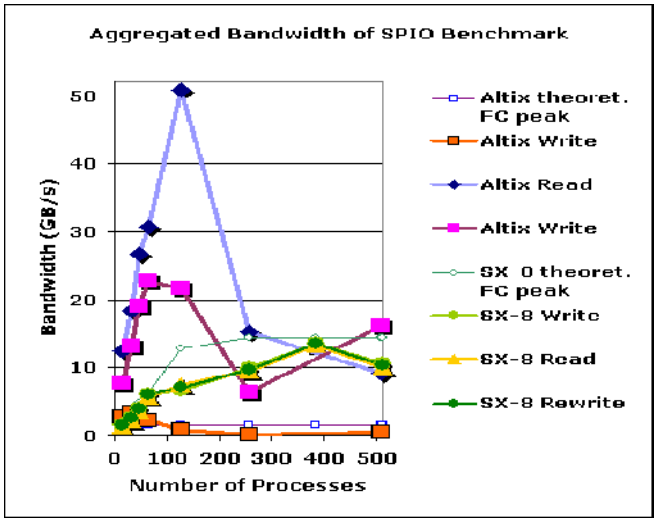


Figure 16: Aggregate bandwidth (in GB/s) of SPIO Benchmark for Columbia and NEC SX-8.

processors, the reads/writes to the cache suffer because they get serialized within the file system.

In Figure 16 the aggregate SPIO bandwidth is plotted together with the theoretical Fibre Channel peak I/O performance documented in Section 2.4. On the SX-8, for up to 18 nodes, the bandwidth is the number of nodes times 0.8 GB/s. For more than 18 nodes, it is constant at 14.4 GB/s. On Columbia, it is constant at 1.6 GB/s. On the SX-8, with 16, 64, and 384 processors, the aggregate bandwidth is 86-96% of peak; for 32, 48, 256, and 512 processors it is 67-84% of peak; and for 128 processors, it is only 53-58% of peak. On Columbia, most of the reported performance (except for writing with more than 64 processors) is because of the caching of small files by the file system in memory. The measured aggregated bandwidth is up to 32 times larger than the physical I/O speed of the 8 Fibre Channel ports.

4.2 Compact Parallel I/O Applications: We present the results for the compact applications namely SSCA #3 and Flash I/O.

4.2.1 HPCS SSCA #3 Benchmark: The SSCA #3 runs were made on a single processor of Columbia and the NEC TX-7 front-end to the SX-8 cluster. The benchmark did not run on the SX-8 system. As the TX-7 front-end (see Figure 2) uses the same GStorageFS file system as the SX-8 nodes, the performance on the TX-7 should be comparable to the SX-8. The results of the SSCA #3 benchmark for both Columbia and NEC TX-7 are presented in Table 3.

Table 3: Bandwidth of HPCS SSCA #3 benchmark on Columbia and TX7.

I/O Component of SSCA #3	Operation Read/Write	Bandwidth (MB/s)	
		Columbia	TX7
Sensor processing	R	81.2	15.8
	W	173	691
Knowledge formation	R	208	44
	W	33.6	38.4
Total	R + W	113	31.4

This benchmark was run with a scale factor of 2 and a dialed grid size of 2 x 2 x 2 = 8 images on a single processor of Columbia and TX7, using the shared file system on both. The size of the image formed is 512 x 762. The I/O components of the results fall into three categories: (a) Sensor processing, where Kernel 1 retrieves the data (read operation) and Kernel 2 stores the formed image back (write operation). Here, the read bandwidth on Columbia is 5.1 times the read bandwidth on the TX7, and the write bandwidth on TX7 is 4 times that on Columbia. (b) Knowledge formation, where Kernel 3 retrieves the image (read operation) and Kernel 4 stores the image (write operation). Here, the read bandwidth on Columbia is 4.7 times that on the TX7, and the write bandwidths on the two systems are comparable. (c) The total I/O bandwidth (K1+K2+K3+K4) on Columbia for both sensor processing (K1+K2) and knowledge formation (K3+K4) is about 3.6 times that on the TX7. In this case, the bandwidths measured are using the POSIX interface and as such they are much smaller compared to the MPI-IO bandwidths in some of the benchmarks above.

4.2.2 Flash I/O Benchmark: Figure 17 shows the results of the Flash I/O benchmark for a standard grid of 8 x 8 x 8. In this benchmark, the underlying MPI-IO routines are used to create a single file and have all processors write directly to the file. The MPI-IO implementation may do the writing collectively. From the graph, it is clear that Columbia's write bandwidth is much better than that of NEC SX-8 for all the three files. The reason for poor performance of the SX-8 is that the system's I/O performance is best for a block size of at least 16 MB, whereas in this benchmark the block size is relatively small. The size of each record from a single processor is (8 bytes / number) * (8 zones in x) * (8 zones in y) * (8 zones in z) * 100 blocks, or 400 KB, which is small for I/O. This is a

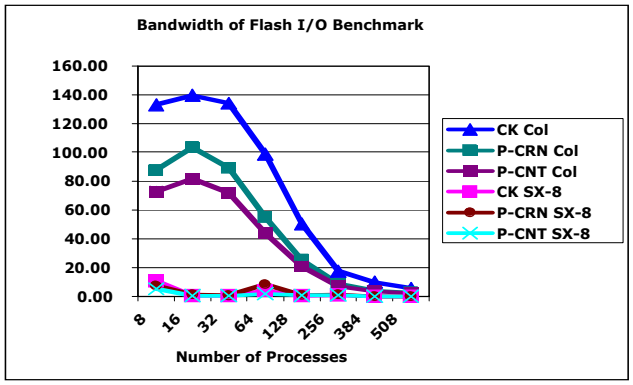


Figure 17: Bandwidth (in MB/s) of checkpointing file (CK), plot file with corners (CRN), and plot file without corners (CNT) as a function of number of processors for the FLASH I/O benchmark using parallel HDF5 on Columbia and SX-8.

major factor in the poor performance currently achieved on both Columbia and SX-8. The guard-cell overhead is a large fraction of the total memory on a processor, thus limiting the size of the record being written to disk to a small size. The checkpoint and plot file routines are identical to those used in the FLASH application [12]. Any techniques to improve the performance of the FLASH I/O benchmark should

also help the FLASH application achieve better I/O performance.

5. Conclusions and Future Work

Based on the results of all these benchmarks, we can draw the following conclusions about the I/O performance of Columbia and NEC SX-8.

On Columbia, the read bandwidth is higher than the write bandwidth, while on SX-8, the write bandwidth is often similar and sometimes better than read bandwidth. Both read and write performance depends very strongly on the block size. For SX-8, the optimal block size is 128 MB, whereas for Columbia it is 2 MB. Both read and write bandwidths can achieve a significant fraction of the physical I/O Fibre Channel speed on several benchmarks. Columbia can achieve higher bandwidth than SX-8 when multiple processors access a single file because it has twice the number of Fibre Channel ports. On the SX-8, the highest aggregated bandwidth could be achieved when accessing individual files from at least 18 nodes. I/O is not scalable with the number of processors on either Columbia or SX-8 when all processors read/write a common file. Constant bandwidth is seen on Columbia for write and rewrite of segmented files, and on SX-8, for read and rewrite.

The performance of POSIX I/O on both SX-8 and Columbia is very comparable. On Columbia, read/write bandwidth can be significantly high when the benchmark is run on a smaller number of processors. On large number of processors, contention for the file system cache causes performance to drop. With parallel MPI-IO, the performance depends highly on the access pattern, and, in the best case (on the SX-8), the performance was very close to the peak performance.

In the future, we plan to run these benchmarks on other leading supercomputers, such as Cray XT3, IBM POWER 5 cluster, and IBM Blue Gene/L.

References:

1. The Office of Science Data-Management Challenge. Report from DOE Office of Science Data-Management Workshops March-May 2004. http://www-user.slac.stanford.edu/rmount/dm-workshop-04/Final-report.pdf.
2. Henry Scott Newman, Tutorial S10: I/O Performance Analysis and Tuning: From the Application to the Storage Device, *IEEE Supercomputing 2005, Nov. 12-18,* Seattle, Washington.
3. Saini, S., Full Day Tutorial M04, Hot Chips and Hot Interconnect for High End Computing Systems, *IEEE Supercomputing 2004*, Nov. 8, 2004, Pittsburgh.
4. The High Performance Computing Modernization Program (HPCMP) Major Shared Resource Centers (MSRCs), http://www.erdc.hpc.mil
5. P. H. Carns, W. B. Ligon, R. B. Ross, and R. Thakur, PVFS: A Parallel File system for Linux Clusters, In Proc. of the Extreme Linux Track: 4th Annual Linux Showcase and Conference, 2000.
6. R. Ross, D. Nurmi, A. Cheng, and M. Zingale. A Case Study in Application I/O on Linux Clusters, Proceedings of Supercomputing 2001. ACM/IEEE, 2001.
7. Robert Ross, Bill Gropp, Rusty Lusk, Rajeev Thakur. M01: Advanced MPI: I/O and One-Sided Communication, IEEE Supercomputing 2005, Nov 12-18 Seattle, Washington.
8. R. Rabenseifner and A.E. Koniges. The Effective I/O Bandwidth Benchmark (b_eff_io), Proceedings of the Message Passing Interface Developer's Conference 2000. Ithaca, NY, 2000.
9. The IOR README File, http://www.llnl.gov/asci/purple/benchmarks/limited/ior/ior.mpiio.readme.html. Lawrence Livermore National Laboratory, Livermore, CA, 2001.
10. SPIOBENCH Benchmarking Information Referenced in NSF 05-625 “High Performance Computing System Acquisition: Towards a Petascale Computing Environment for Science and Engineering”, http://www.nsf.gov/pubs/2006/nsf0605/nsf0605.jsp
11. Meuse T., High Productivity Computer Systems (HPCS) Scalable Synthetic Compact Application (SSCA) Benchmark SSCA # 3, Sensor Processing Knowledge Formation and Data I/O, Version 1.0a Release, Principal Investigator and Productivity Team Meeting, Marina del Rey, California, January 10-11, 2006.
12. M. Zingale, Flash I/O Benchmark Routine, http://flash.uchicago.edu/~zingale/flash_benchmark_io/, University of Chicago, 2001.