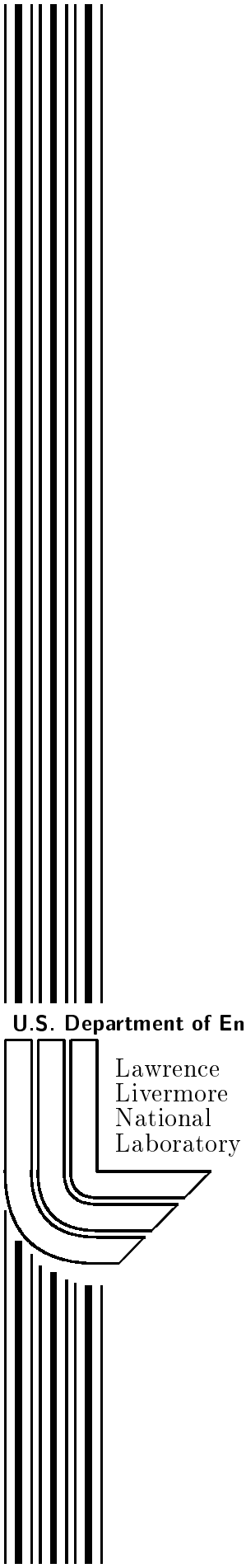
UCRL-ID144018

The Performance of PFS, the Compaq Sierra Product's Parallel File System

A. C. Uselton

June 20, 2001



DISCLAMER

This work was performed under the auspices of the U. S. Department of Energy by the University of California, Lawrence Livermore National Laboratory under Contract No. W-7405-Eng-48.

The Performance of PFS, the Compaq Sierra Product's Parallel File System*

Andrew C. Uselton

June 20, 2001

Abstract

In FY 2000 Livermore Computing took delivery of serial number one of the Compaq Sierra high performance cluster product. TheSierra product employs a derivative of the Tru64 UNIX operating system called Tru-Cluster, which provides a cluster-wide parallel file system called PFS. This report documents the observed performance of PFS along with the performance of some of the underlying file system components. Testing reveals that the underlying AdvFS file system does a good job of read-ahead and write-behind I/O performance enhancement at the expense of a high CPU utilization. On the other hand, PFS performs at only a fraction of the speed that the underlying I/O and communication hardware allow.

1 Introduction

This report presents the data transfer performance of the mass storage subsystem for a high performance computing system. All of the tests reported use sequential block data tranfers buffered through an underlying file system and measure performance with one of the benchmark programs *Bonnie* or *ior*. The benchmarks include tests of individual file system components and over-all tests of large, cluster-wide programs.

"I/O" is the generic term for input data transfer operations, or reads, and output data transfer operations, or writes. This report details values for the rate, in megabytes per second (MB/s), at which data may be read from and written to the storage subsytem, and refers to such a value as a *data rate*. There are other measures for performance, including *latency*, *reliablility* or *stability*[1], *price* and *capacity*. This report confines itself to data rate and *cpu utilization*. Each I/O operation reads or writes a *block* of data whose size is measured in

*This paper and a summary of the results it contains may also be obtained at the URL: http:/ /www.llnl.gov/sccd/lc/hpcs/benchmarks/tckk-pfs/html/.

[1]Both the "mean time between failures" and the ability to continue in the presence of errors.

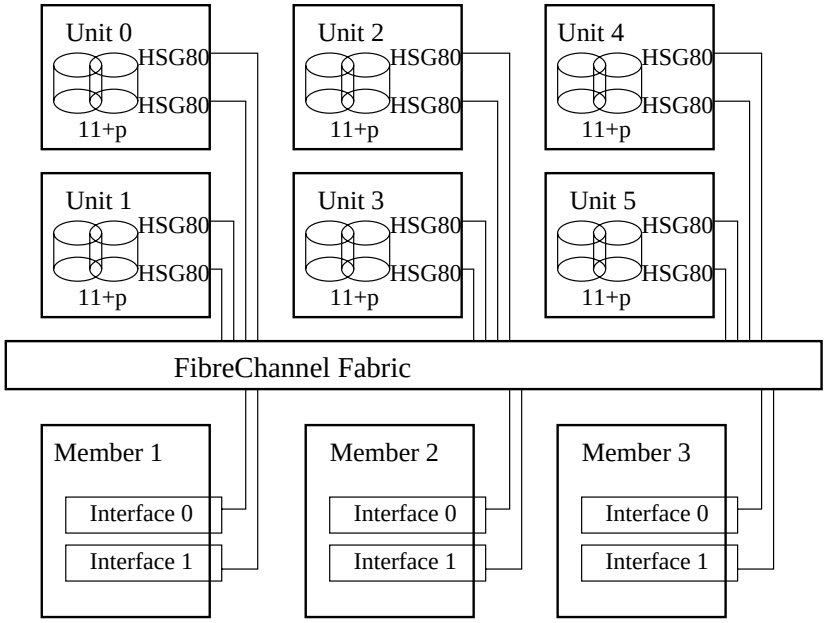


Figure 1: Test environment for clusters A and B

bytes. An I/O performance test consists of measuring the duration of one or more I/O operations of a given *block size*. The data rate varies depending on the configuration of the storage subsystem and depending on the details of the I/O. Buffered I/O tends to increase the data rate for small amounts of I/O. For larger amounts of total I/O the data rate asymptotically descends to a stable value. The tests reported here use *sequential* I/O, which will generally give higher values than for *random* I/O due to the extra seek time required by the latter. Early tests varied the total amount of the I/O in order to find the asymptotic value of data rate (the *sustained data rate*). The values reported here are all for sustained data rate.

The computer system being tested consists of 128 nodes connected via the Quadrics Elan3 high performance network interconnect. Each node is a Compaq ES40, which is an *Alpha*-based 4-way Symetric Multiprocessor (SMP) computer. The nodes are grouped into four clusters of 32 nodes each. Each of the first three in a cluster is connected to a set of Compaq StorageWorks RAID arrays and is referred to as an *I/O node*. The clusters are running Compaq's Sierra cluster software based on version 4 of the Tru64 UNIX operating system.

Figure 1 shows the layout of the I/O nodes and the RAID arrays for one cluster. The connection between I/O nodes and RAID arrays is via a *FibreChannel fabric*, which can transfer $100MB/s$. The figure shows each of the I/O nodes with two Fibre Channel interfaces connecting to a Fibre Channel switch. The switch is connected via Fibre Channel to each of two HSG80 RAID controllers on each RAID array. There are six RAID arrays in each cluster. In Figure 1 each RAID array has a set of twelve SCSI disks in a RAID-5 configuration with one disk acting as a *parity disk*. We refer to this as an 11+p RAID chain. Sec-

tion 2 examines clusters configured with RAID chains of 4+p, 5+p, 8+p, and 11+p. Those tests extend the results of [1] in which the tests measured *raw* device performance, i.e. the performance of the devices without any file system structure or buffering. The central conclusion of that report was that dividing the 12 disks of each RAID unit into two 5+p RAID chains gave 25% to 40% better performance. The best observed performance for a single I/O node in that report is as follows:

	write rate (MB/s)	read rate (MB/s)	for blocks larger than
raw I/O	77	117	1MB

The Sierra product's software constructs a cluster from a set of nodes by providing these three facilities:

- A common cluster-wide root file system called *CFS*,
- A set of cluster administration tools,
- A cluster-wide resource manager, and
- A parallel execution environment.

CFS is the mechanism whereby the Sierra product mounts the more conventional *AdvFS* file systems into a single hierarchy and then exports it across the high speed interconnect. The parallel file system, *PFS*, is a mechanism for striping files across a collection of underlying AdvFS file systems. If the component AdvFS file systems are on independent hardware then a significant speedup is possible for large I/O transactions.

The *Bonnie*[2] benchmark, as used for this report, performs a series of POSIX-compliant I/O operations, first writing a series of data blocks to a file and then reading those blocks. It times both operations and reports the data rate as the amount written or read divided by the measured time. When the I/O subsystem to be measured consists of several separate file systems this report uses multiple instances of the Bonnie benchmark, run concurrently, to measure aggregate data rate. The *ior* benchmark, in contrast, runs in the cluster's MPI-based parallel environment. It has the option of performing MPI-based I/O or POSIX-based I/O, and for this report all the I/O is POSIX-based.

The remainder of this report is organized as follows. Section 2 characterizes the performance of an AdvFS file system on a single device. PFS will gather several such file system and device combinations together to form a parallel file system. Section 3 will present the aggregate performance of several such file system and device combinations on a single I/O node, but without the PFS mechanism. Section 4 repeats the test from Section 3 with the PFS mechanism. Sections 5 and 6 repeat these tests, this time operating accross the interconnect.

[2] http://www.textuality.com/bonnie/intro.html

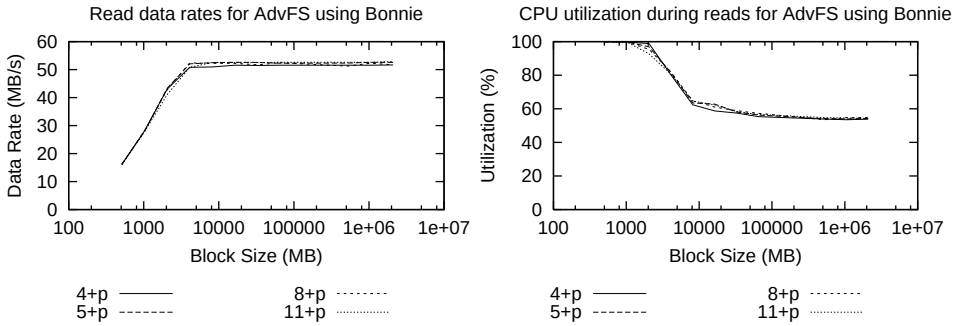


Figure 2: Read data rate and CPU utilization for various sized RAID chains

Section 7 constructs the full parallel file system from three I/O nodes and characterizes the performance of a single compute node in the parallel environment. Finally, Section 8 reports on the aggregate performance of many nodes writing to the parallel file system simultaneously via two alternative parallel I/O methods, *shared file pointers* and *individual file pointers*.

This report concludes with a brief summary of the results presented. We discover that the CFS mechanism for presenting a single file system to the cluster via the interconnect is the primary bottleneck. Of the parallel I/O methods presented in the last section the individual file pointers perform better for the size clusters observed, but did not appear to scale well. The shared file pointer did not perform as well for most of the tests but seemed to scale better and might be superior if the cluster itself were larger.

2 The Performance of AdvFS

The parallel file system's performance ultimately depends on the performance of the underlying components from which it is constructed. In the Sierra product these components are AdvFS file systems built on Compaq StorageWorks RAID arrays. As demonstrated in [1] there is a performance boost for subdividing a StorageWorks RAID array into two chains. It results from the parallelism exploited by the two controllers on the RAID array. The goal of the PFS file system is high speed bulk storage, so there is no reason to further subdivide an individual RAID chain. Similarly, PFS organizes the striping of files across storage components, so there is no reason to build an AdvFS from more than one RAID chain. Thus there is a one-to-one correspondence between the under-

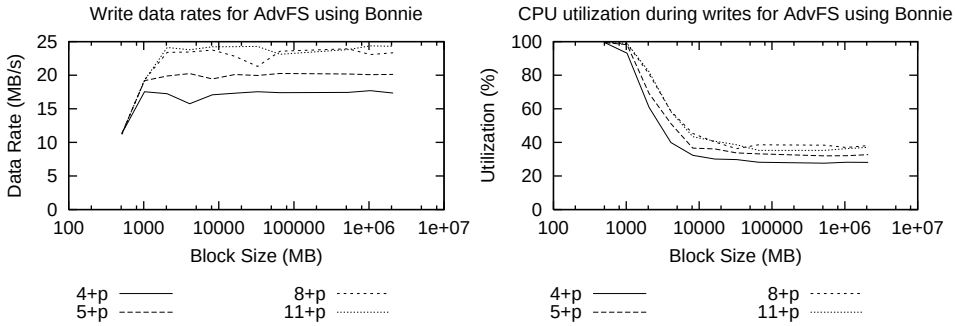


Figure 3: Write data rate and CPU utilization for various sized RAID chains

lying RAID chains and the AdvFS file systems from which a PFS file system is constructed. This section considers the performance of one such component.

Figure 2 shows the read performance of a single AdvFS file system composed of a single RAID 5 chain constructed as either 4+p, 5+p, 8+p, or 11+p. In each case the performance is the same. For block sizes above 8KB the data rate exceeds $50MB/s$ and the CPU utilization is between 50% and 60%. Smaller block sizes do not perform as well and cause more CPU activity. This is exactly what one would expect from the AdvFS file system as it performs *read-ahead* and caches the results. The performance compares favorably with the unbuffered (raw) read performance reported in [1], which was also about $50MB/s$ for each size of RAID chain. Since the raw data rates only achieved peak performance for block sizes of 1MB or larger it is clear that AdvFS employs, and benefits from, larger I/O transactions via a read-ahead strategy. This is also born out by the high CPU utilization for the smaller AdvFS reads, where the data read in ahead of a request will have to be coppied or otherwise managed while the file system awaits requests for it.

Figure 3 shows that the write performance of AdvFS resembles the read performance with one exception. As was the case for reads, the data rate achives a plateau and the CPU utilization a steady low value for all but the smallest block sizes. Similarly, the write data rates plateau at values comparable with the raw I/O performance reported in [1]. AdvFS is merging small writes into larger ones (called *write behind caching*) at the expense of higher CPU utilization. The one difference is that the peak data rate for writes improves with the size of the RAID chain up to 8 disks: $17MB/s$ for 4+p, $20MB/s$ for 5+p, and $24MB/s$ for 8+p and 11+p.

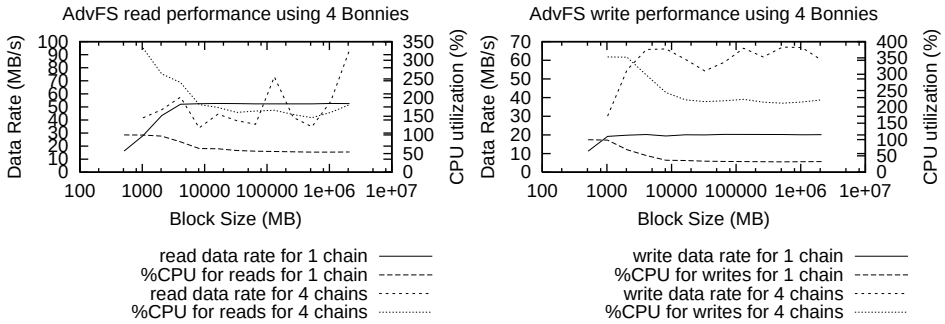


Figure 4: Data rate and CPU utilization for one and four 5+p chains

The central result reported in [1] is that organizing a RAID unit into two 5+p RAID chains gives better aggregate performance than organizing it as one 11+p RAID chain. This is because the RAID units have two controllers and can achieve some parallelism. Though not reported here, several tests showed the same was true for the RAID units organized in AdvFS file systems. The twelve disks are better organized as two 5+p chains rather than one 11+p.

The remainder of this report focuses on the performance of 5+p RAID chains, organized as two chains per StorageWorks RAID array, two such arrays per I/O node, and three I/O nodes for a total of twelve 5+p RAID chains. Each chain is formatted as a distinct AdvFS file system (domain and set), and four such file systems are mounted locally on each I/O node. The next section presents results for running all the chains on a single node at once.

3 The Aggregate Performance of a Single I/O Node

Figure 4 shows the aggregate data rate and CPU utilization of running four reads or four writes at once, one each, to the four RAID chains of a single I/O node. The cumultaive write data rate of $65MB/s$ on the plateau compares favorably with the raw I/O rate of $77MB/s$ reported in [1]. Note that the CPU utilization is around 200% at a minimum, so AdvFS is working pretty hard to achieve these results. On the other hand, the aggregate read data rate is poor. In some cases less than the rate for reading from a single chain, though for 2MB blocks it is $94MB/s$, which does approach the $117MB/s$ observed in [1] for

raw I/O. For the poorly performing reads the CPU utilization is only around 150%, so it would appear that AdvFS is having difficulty managing the four read processes efficiently.

Single I/O node best performance data rates		
	write rate (MB/s)	read rate (MB/s)
raw I/O	77	117
AdvFS	65	94

This Section has characterized the AdvFS performance of the I/O subsystem because the PFS file system is built from AdvFS components. Knowing that an otherwise unoccupied I/O node can deliver $94MB/s$ read and $65MB/s$ write throughput from the AdvFS layer gives us a basis for evaluating the performance of the PFS file system. There are two independent ways that this potential data rate, or *bandwidth*, may be lost in delivery to a parallel application running on the cluster. First, the striping activity at the PFS layer may impose a penalty, and second, the communication of the file system over the interconnect via CFS may impose a penalty. In the next section one and two instances of the Bonnie benchmark run on an I/O node, communicating with a PFS file system constructed of AdvFS components all of which are locally mounted on the I/O node. In Section 6 we will remove PFS from consideration and examine the performance of AdvFS as delivered across the interconnect.

4 The Performance of a Local PFS File System

Figure 5 shows the data rate for one and two Bonnie processes reading from and writing to a PFS file system constructed from the same four AdvFS 5+p RAID chains used in Section 3. We see a slight performance gain running two Bonnies over running one. The read data rate for one Bonnie is near $40MB/s$, and for two it is between $40MB/s$ and $50MB/s$. These values are consistent with those observed for writing to four independent AdvFS file systems, though the data never reaches $94MB/s$ as it did for very large blocks in Section 3. The write data rate for one Bonnie process hovers around $55MB/s$, and for two is quite comparable with the maximum expected performance of the I/O node at $77MB/s$.

Single I/O node to its own RAID chains		
	write rate (MB/s)	read rate (MB/s)
raw I/O	77	117
AdvFS	65	94
PFS	77	48

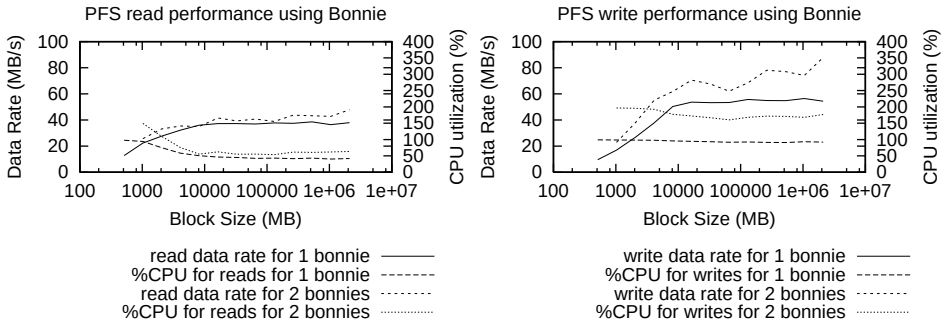


Figure 5: Data rate and CPU utilization for one and two Bonnie benchmarks on a PFS file system constructed entirely of four locally mounted 5+p chains

The foregoing tests characterize the performance of an I/O node writing to, or reading from, its own I/O subsystem. In practice, data destined for, or coming from, the storage subsystem is written, or read respecively, by the compute nodes and is communicated to the I/O nodes across the cluster interconnect. The experiments in the next section repeat those from Section 3 (i.e. no PFS) with the I/O taking place on a compute node.

5 The AdvFS Performance of an I/O Node Accessed Across the Interconnect

Figures 6 and 7 show the results of running one, two, and four Bonnie benchmarks in which each instance writes to a separate 5+p RAID chain as well as the result of running eight total Bonnies to the four RAID chains. The channel to the file system comprised of the source node, the interconnect, and the I/O node, is saturated at four instances. The interconnect is known to be capable of transmitting $200MB/s$ and we are seeing about 15% of this between two nodes. The I/O node is capable of handling about three times what one compute node can deliver across the interconnect. This would seem to indicate that the CFS layer presenting the file system over the interconnect is imposing a significant penalty.

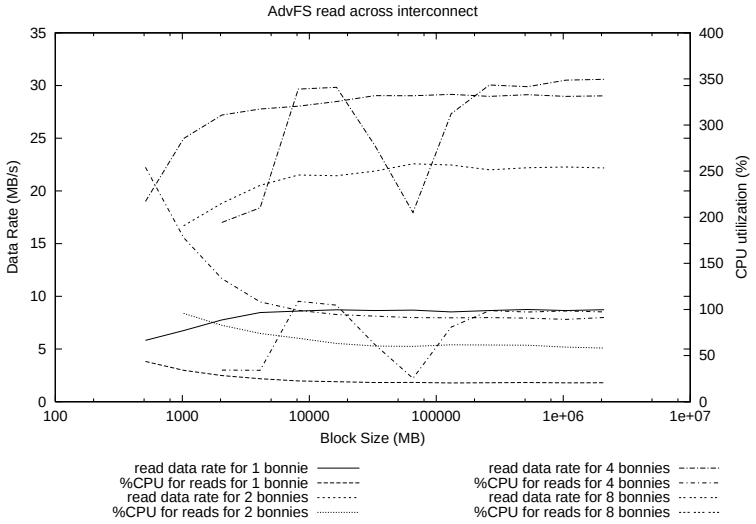


Figure 6: Data rate and CPU utilization across the cluster interconnect.

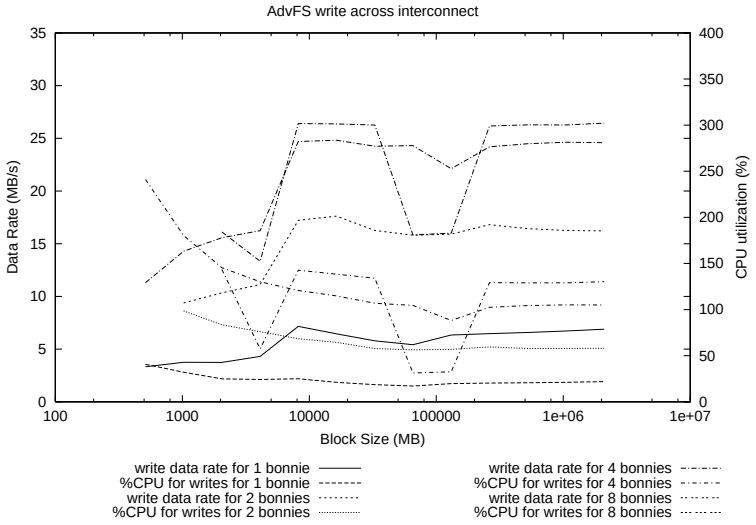


Figure 7: Data rate and CPU utilization across the cluster interconnect.

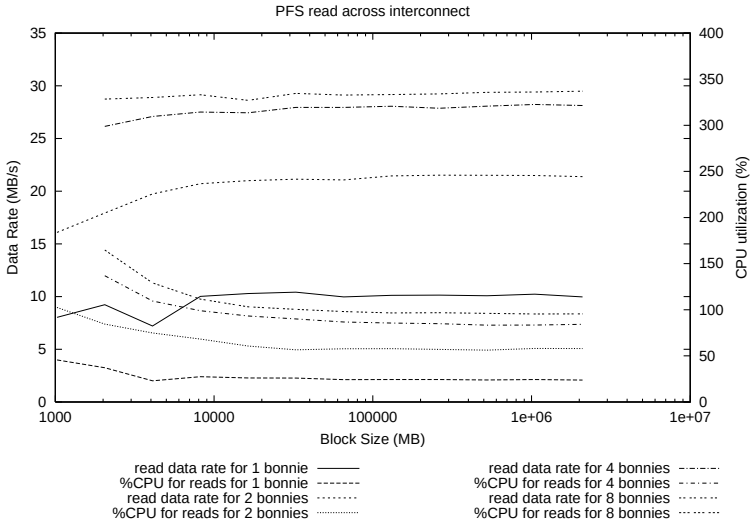


Figure 8: Performance of PFS across the cluster interconnect.

Data Rate for a single I/O node			
	data source	write rate (MB/s)	read rate (MB/s)
raw I/O	I/O node	77	117
AdvFS	I/O node	65	94
PFS	I/O node	77	48
AdvFS	compute node	26	31

6 The PFS Performance of an I/O Node Accessed Accross the Interconnect

Figures 8 and 9 show the results of running one, two, four, and eight Bonnie benchmarks in which each instance writes to a separate file on the PFS composed of the four 5+p RAID chains from the previous section. The channel to the file system comprised of the source node, the interconnect, and the I/O node, is saturated at eight instances. The data rates at saturation are comparable to those in the previous section, so it appears that the PFS layer added little extra burden to the I/O.

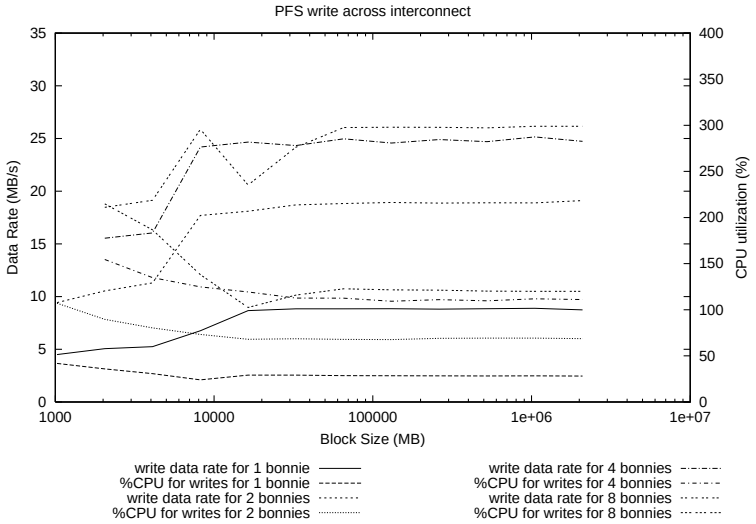


Figure 9: Performance of PFS across the cluster interconnect.

Data Rate for a single I/O node			
	data source	write rate (MB/s)	read rate (MB/s)
raw I/O	I/O node	77	117
AdvFS	I/O node	65	94
PFS	I/O node	77	48
AdvFS	compute node	26	31
PFS	compute node	26	29

The foregoing tests have characterized the performance of a single I/O node. In [1] we characterized the raw disk performance, and in this report we have presented the performance of the underlying file system components, and their performance when operated in parallel, when gathered together in a parallel file system, and when accessed across the interconnect. In practice the cluster will be operated with three I/O nodes and will be accessed by parallel programs operating on up to 32 nodes simultaneously. The next section introduces *ior*, a benchmark for measuring aggregate I/O performance across many parallel processes, and recapitulates the tests for data rate from one compute node across the interconnect to the I/O subsystem, though the I/O subsystem consists of three I/O nodes in this case. Section 8 follows with the performance of multiple nodes participating in the I/O.

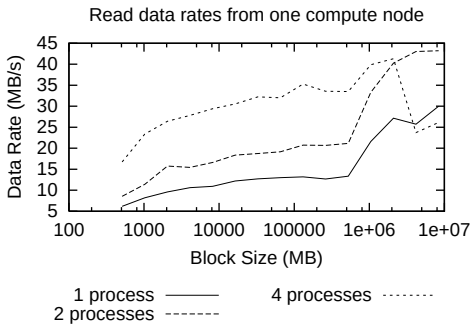


Figure 10: MPI reading in to one node

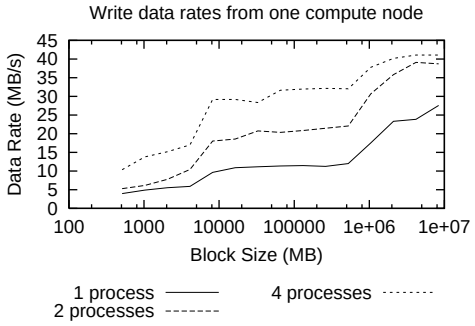


Figure 11: MPI writing from one node

7 The MPI (POSIX) I/O performance of PFS

Figures 10 and 11 show the performance of a parallel program running in the MPI environment and making POSIX compliant I/O calls to the PFS file system. Each process communicates with its own file. PFS stripes each file across the twelve AdvFS file systems on the three I/O nodes.

Running several parallel MPI tasks gives better I/O perforce on a single node than does running a single MPI task except for very large blocks being read by four MPI processes. For 2MB blocks the read and write rates reach 41MB/s aggregate over the four processes. It is not clear why the read rate falls off for four MPI processes reading very large blocks.

Data Rate for a single compute node					
	data source	parallel MPI	I/O nodes	write rate (MB/s)	read rate (MB/s)
raw I/O	I/O node	no	1	77	117
AdvFS	I/O node	no	1	65	94
PFS	I/O node	no	1	77	48
AdvFS	compute node	no	1	26	31
PFS	compute node	no	1	26	29
PFS	compute node	yes	3	41	41

Thus far we have determined that a parallel program in the MPI environment can read and write faster to three I/O nodes than it can to a single I/O node, but not yet as fast as a single I/O node can handle local I/O. We may hope that there is some "bandwidth" left to be exploited in the I/O subsystem. In the next section we look at the aggregate performance for multiple compute nodes performing I/O simultaneously.

8 The Scalability of PFS

All of the foregoing tests that have involved multiple processes have organized the I/O such that each process reads from or writes to a distinct file. In the case of parallel programs written in the MPI environment there is an alternative. MPI will allow all the processes to write to a single file using a *shared file pointer*, as well as *individual file pointers* for distinct files. Ior produced results for varying numbers of nodes with four processes per node and both individual and shared file pointers. The sierra product has a maximum cluster size of 32 nodes and the largest parallel program available in our configuration was between 27 and 29 nodes depending on the actual cluster used[3].

Figure 12 presents the results of each of these tests. For parallel programs using individual file pointers the aggregate data rate reached a peak of 112 MB/s

[3] I/O nodes and login nodes were excluded from parallel programs. A parallel program is allowed to span multiple clusters, but each cluster has its own I/O subsystem. Thus the maximum scalability test was a parallel program running entirely on a single cluster.

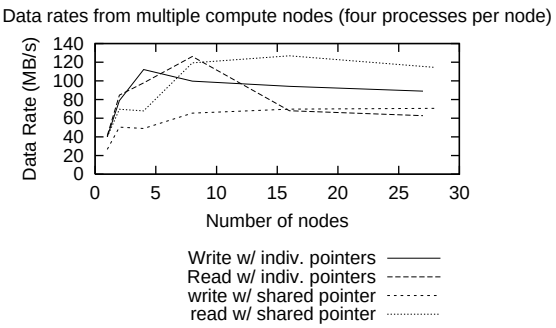


Figure 12: MPI I/O from multiple nodes

for writes and 126 MB/s for reads at four and eight nodes respectively. The write rate descends to 89 MB/s and the read rate to 63 MB/s for larger parallel programs. For parallel programs using shared file pointers the write rate reaches 71 MB/s and the read rate 127 MB/s, and neither shows a significant drop off due to a larger number of nodes.

Summary of results						
	parallel MPI	file pointer	compute nodes	I/O nodes	write rate (MB/s)	read rate (MB/s)
raw I/O	no	indiv.	0	1	77	117
AdvFS	no	indiv.	0	1	65	94
PFS	no	indiv.	0	1	77	48
AdvFS	no	indiv.	1	1	26	31
PFS	no	indiv.	1	1	26	29
PFS	yes	indiv.	1	3	41	41
peak MPI	yes	indiv.	1	4 or 8	112	126
peak MPI	yes	shared	1	28 or 16	71	127

9 Conclusion

The Compaq Sierra cluster product has an auxiliary, high performance I/O subsystem capable of sustained I/O of 77 MB/s for writes and 117 MB/s for

reads per I/O node for each of three I/O nodes giving an aggregate bandwidth of 200 to 300 MB/s. The interconnect is also capable of delivering in excess of 200MB/s. For I/O being written to the PFS parallel file system built atop this I/O subsystem there appears to be little added burden in keeping track of the striping of large files across the individual file system components. On the other hand presenting the parallel file system across the interconnect via the CFS file system seems to add a significant burden.

The two modes of parallel I/O, shared versus individual file pointers, show two different trends in aggregate performance. Individual file pointers achieve their highest performance on relatively few nodes and performance appears to drop off with additional nodes. Conversely, shared file pointers do not perform as well for fewer nodes, but also do not seem to drop off for larger numbers of nodes. For the maximum number of nodes tested the read rate for shared file pointer I/O even exceeded the best read rate for individual files. One may speculate as well that shared file pointers will scale better to larger clusters, but verifying that will have to await the next release of the TruCluster operating system.

References

[1] Andrew C. Uselton. The raw disk i/o performance of compaq storageworks raid arrays under tru64 unix. Technical Report UCRL-ID-141831, Lawrence Livermore National Lab, 2000.

University of California
Lawrence Livermore National Laboratory
Technical Information Department
Livermore, CA 94551