Direct-Attached Disk Subsystem Performance Assessment

Roger D. Chamberlain
Berkley Shands

Roger D. Chamberlain and Berkley Shands, “Direct-Attached Disk Subsystem Performance Assessment,” in *Proc. of 4th Int’l Workshop on Storage Network Architecture and Parallel I/Os*, September 2007.

Direct-Attached Disk Subsystem Performance Assessment

Roger D. Chamberlain*† and Berkley Shands†
*Exegy, Inc., St. Louis, Missouri
†Dept. of Computer Science and Engineering, Washington University in St. Louis
{roger,berkley}@cse.wustl.edu

Abstract

Direct-attached storage has historically had the reputation of being less capable than equivalently sized SAN installations. Here, we empirically demonstrate the performance achievable in multiple-terabyte, direct-attached disk subsystems. A number of parameters are explored, including file system, number of logical drives, and RAID configuration.

1. Introduction

For a number of years, our group has been constructing appliance-class systems that include significant local disk storage as an integral component of the system [1,2,3]. The appliance is one that performs text searches on large unindexed data sets. The data are streamed out of the disk subsystem into a field-programmable gate array (FPGA) which performs in initial filtering operation, eliminating most of the data from consideration. Data elements which survive the initial scan are streamed into the processor for higher-level computation that ultimately determines whether or not the data match the user's search query. As currently delivered to customers, a typical Exegy TextMiner appliance contains a 4.2 TB disk subsystem, 4 AMD Opteron processor cores, 2 Xilinx Virtex-4 FPGAs, and 16 GB of memory.

As a part of this activity, we have previously reported empirical performance results for the disk subsystem [4]. In this previous study, we investigated primarily read performance across a variety of disk subsystems (drive types, controllers, RAID configurations, etc.) and file size distributions.

Here, we describe a new set of experiments that both updates the previous investigation to currently available components and also widens the scope of the study, adding multiple file systems and usage patterns. Given the nature of the appliance, the disk subsystem can be characterized as a "write-once, read-many" system, in which data are typically loaded once and multiple queries are then performed. As a result, usage patterns that match this style are the ones we investigate here.

This research supported by AMD, Arrow, LSI Logic, Exegy, and NSF grant CCF-0427794. R.D. Chamberlain is a principal in Exegy.

2. Experimental Systems

We use three hardware platforms to carry out our experiments. The first platform occupies 3U of rack space in a single enclosure and contains 16 drives (totaling 8 TB). The second platform occupies 9U of rack space in three enclosures and contains 32 drives (totaling 13 TB). The third platform occupies 3U of rack space in 1 enclosure and contains 16 drives (totaling 8 TB). Details of each hardware platform are listed below:

Platform 1:

- 1 TSTCOM ESR316 enclosure (3U)
- 1 Tyan 3992 motherboard
- 2 AMD Opteron 2222 dual-core 3 GHz procs
- 2 LSI 8888elp SAS disk controllers
- 16 Seagate 500 GB NS series SATA-2 disks

Platform 2:

- 1 Uniwide 3546ES enclosure (3U) and motherboard
- 4 AMD Opteron 8222 dual-core 3 GHz procs
- 1 LSI 8888elp SAS disk controller
- 2 Xtore XJ1100 SAS JBOD expansion enclosures (3U each)
- 16 Seagate 500 GB NS series SATA-2 disks
- 16 Seagate 320 GB AS series SATA-2 disks

Platform 3:

- 1 TSTCOM ESR316 enclosure (3U) with expander backplane
- 1 Supermicro H8DMi motherboard
- 2 AMD Opteron 2222 dual-core 3 GHz procs
- 1 LSI 8888elp SAS disk controller
- 16 Seagate 500 GB NS series SATA-2 disks

Each platform has 16 GB of RAM, uses CentOS 5 (Linux kernel 2.6.22), and the disk controllers are each installed in 8-lane PCI-e slots.

In a few of the experiments, an additional disk controller is added to platform 3. In this case, the entire disk subsystem of platform 2 is added to platform 3. When used, this system is called platform 3+.

Also present in all systems (installed in a PCI-X slot) is a board that includes two Xilinx Virtex-4 LX100 platform FPGAs. Each system has a dedicated boot disk (using a native controller on the motherboard) that is independent of the disk subsystems used for experimentation.

3. Disk Subsystems

Since each LSI disk controller supports 8 SAS channels and there are two controllers present in platform 1, each disk is individually connected to one of the controllers via a dedicated channel. Platform 2 takes advantage of the 24 lane SAS switches present in the expansion enclosures to support 32 disks using a single LSI controller. The logical topology is shown in Figure 1. Finally, platform 3 exploits the SAS switches in its enclosure to support 16 drives with one controller.

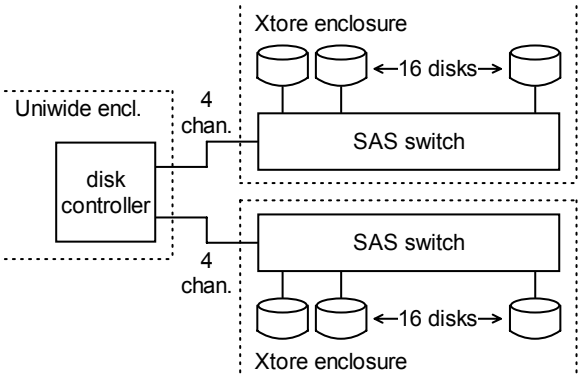

Figure 1: Disk subsystem for platform 2.

Several distinct configurations are investigated. The disks are partitioned into 1, 2, 4, 8, or 12 logical drives (RAIDs). For example, platform 1 configured for 2 logical drives would have 8 physical disks per logical drive (denoted as a 2x8 configuration), and platform 2 configured for 8 logical drives would have 4 physical disks per logical drive (denoted as an 8x4 configuration). The logical drives are also configured as either RAID0 or RAID5. Finally, two distinct file systems are utilized, ext3 or SGI's xfs.

4. Data Movement

The software organization for reading data from the disk subsystem and delivering it to the FPGA (the normal path that is exploited in our systems) is illustrated in Figure 2. A number, *k*, of independent threads are allocated to associated silos. The silos are FIFO buffers in system memory that are used to stage data from the disk subsystem prior to delivery to the FPGA. The number of silos, *k*, is normally equal to the number of independent RAIDs in the disk subsystem configuration.

It is important to point out that the data movement indicated in the figure is under the control of an execution thread on one of the general-purpose processors, but the actual data transfers are all DMA transfers commanded either by the disk controller (for data moving from disk to silo) or by the firmware socket within the FPGA (for data moving from silo to FPGA).

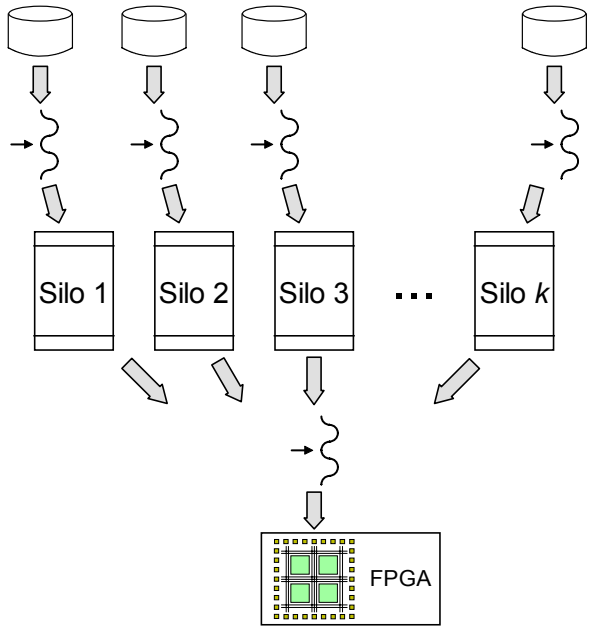

Figure 2: Read silos.

5. Experiments

The data set used in the majority of the experiments comprises a total of 1.9 million

individual files drawn from a variety of sources, including: the complete Medline PubMed abstract database, the multilingual Reuter's corpus, one full day of CNN news articles, the TREC-5 Confusion Track corpus, and 4 captured Windows system images. Figure 3 plots a histogram of the number of files for each file size, and Figure 4 plots the data volume present in each of the bins of the histogram. The total data set is 129 GB in size. The graphs clearly show a large number of small files present; however, the majority of the data is in the relatively smaller number of large files.

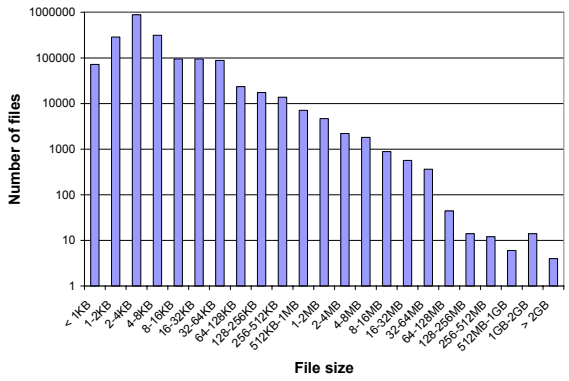

Figure 3: File size distribution.

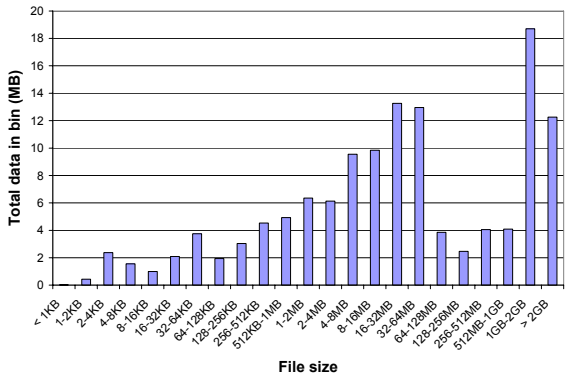

Figure 4: Allocation of data across data set.

Additional data sets are derived from the original data set of Figure 3 by concatenating files smaller than a given threshold into aggregate files (maintaining appropriate indices to retain the original file semantics) using the techniques described in [5]. This mechanism establishes a minimum file size to be managed by the underlying file system and is effective for infrequently written but frequently read file systems.

In addition to the above data sets, a collection of 8 GB synthetically generated files totaling an additional 128 GB is used to exercise the disk subsystem with sequential accesses and minimal meta-data processing required.

The following parameters are fixed for each experiment:

- platform {1, 2, or 3}
- number of logical drives {1, 2, 4, 8, or 12}
- RAID configuration {RAID0 or RAID5}
- file system {ext3 or xfs}

Prior to each experiment, all of the data sets described above are loaded onto freshly formatted file systems (one file system on each logical drive).

Once the data sets have been loaded, read tests are performed on each data set, separately measuring directory lookup times (i.e., meta-data processing) and file read times (i.e., data movement). These are followed by write tests using the same data sets.

6. Results

6.1 IOzone performance

In addition to the results presented here, [6] provides a more comprehensive reporting of the experimental data. We start with the commonly used IOzone benchmark (www.iozone.org), which tests file I/O performance for the following operations: read, write, re-read, and re-write. In keeping with the requirements of IOzone, we report the following concerning the file system construction and mounting:

```
> /sbin/mkfs.xfs -L exegy-s0 -f -d sunit=256,swidth=1024,\
unwritten=1 -l version=2,sunit=256 /dev/sdb1

> LABEL=exegy-s0 /s0 xfs noatime,nodiratime,largeio,\
ihashsize=131072,allocsize=1073741824,att2,barrier,\
logbufs=8,logbsize=262144, inode64 2 2
```

Table 1: IOzone benchmarks.

	platform 1, 4x4, RAID0	platform 2, 8x4, RAID0
read rate	1,024 MB/s	1,189 MB/s
re-read rate	1,022 MB/s	1,189 MB/s
write rate	961 MB/s	1,034 MB/s
re-write rate	804 MB/s	826 MB/s

For the xfs file system, Table 1 shows the IOzone results for platforms 1 and 2. Clearly, these are very capable systems, with performance often exceeding 1 GB/s read and write rates.

6.2 Read performance

Figure 5 shows the read throughput (including both directory lookup times and data transfer times) for the two file systems as a function of the minimum file size in the data set. These experiments were performed on platform 2.

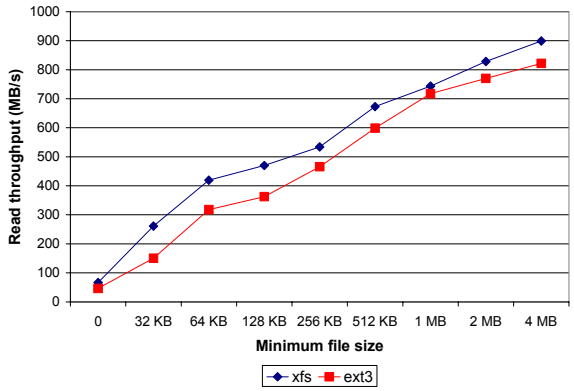

Figure 5: Read throughput, platform 2, 8x4 RAID0.

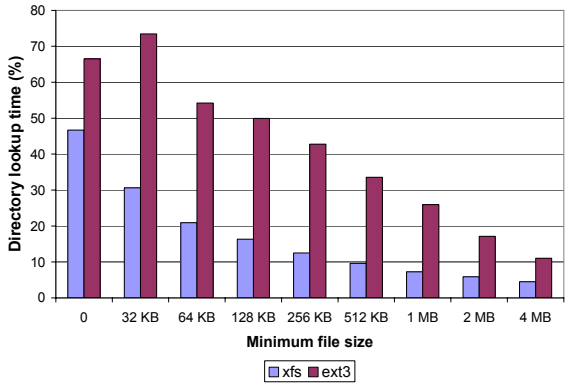

Figure 6: Fraction of time performing directory lookups, platform 2, 8x4 RAID0.

The performance is clearly impacted fairly dramatically by the large numbers of small files present in the original data set of Figure 3. The fraction of the total time that is consumed by directory lookups is plotted in Figure 6. While the fraction of time spent performing directory lookups decreases as the number of files decreases (and minimum file size increases), the directory lookup overheads in ext3 are significantly greater than those in xfs. In these experiments, the xfs file system outperforms ext3 primarily on the basis of faster directory lookup times. Both file systems achieve 1 GB/s read rate on the synthetic data set.

The experiments of Figure 5 and Figure 6 are repeated in Figure 7 and Figure 8 for platform 1 (using the pair of LSI controllers). Again, the xfs file system outperforms the ext3 file system, and for similar reasons (primarily faster directory lookup times).

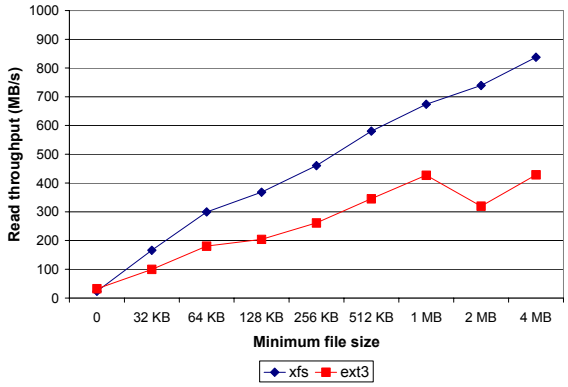

Figure 7: Read throughput, platform 1, 4x4 RAID0.

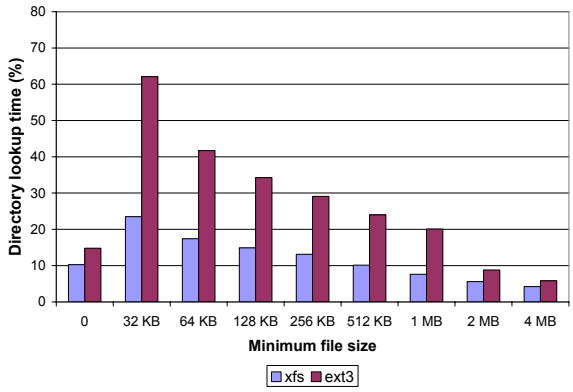

Figure 8: Fraction of time performing directory lookups, platform 1, 4x4 RAID0.

However, there are a number of additional interesting features present in these two graphs. First, in Figure 7 the throughput unexpectedly drops off for the two highest values of minimum file size. This is due to the positioning of these latter two data sets on the interior tracks of the disks. Further data demonstrating (more directly) the impact of data positioning on throughput are shown later. Second, the fraction of time spent on directory lookups is smaller for the original data set than for a minimum file size of 32 KB. Here, the seeks necessary to read the large quantity of

very small files are harder to hide (relative to the platform 2 configuration with double the number of drives).

To illustrate the performance impact of processing, Figure 9 plots the total processing throughput for a pair of search engines executing on the FPGA [7]. Here, data files are read from the disk subsystem and delivered to the FPGA for searching. The approximate match engine [8] is capable of a maximum throughput of 875 MB/s (limited by the FPGA's I/O capability and measured using the synthetically generated data set). The regular expression engine [9] is capable of a maximum throughput of 650 MB/s (limited by the internal processing capability of the FPGA and measured using the synthetically generated data set).

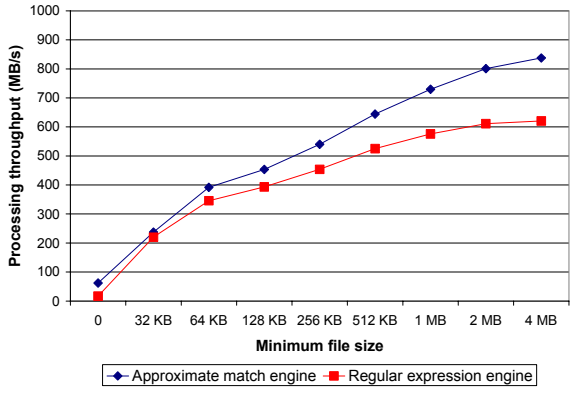

Figure 9: Processing throughput for two search engines, platform 2, 8x4 RAID0, xfs file system.

The clear implication from this plot is the fact that the disk subsystem read performance limits the overall processing throughput for low minimum file sizes and that the FPGA's requirements (whether they are I/O or internal processing) limit the overall throughput at higher minimum file sizes.

This trend can be seen to continue in Figure 10, in which platform 3+ (platform 3 augmented with an additional controller and 32 drives) is used with a 12 logical drive organization both for read-only runs and for processing by the FPGA. The results shown in the figure indicate a much higher achievable read rate for the disk subsystem alone. However the total processing throughput is limited by the FPGA's rate.

Note that for this set of experiments, the processor is not the limiting factor in determining the achievable throughput. When performing the above runs, the processor averages 72% idle time.

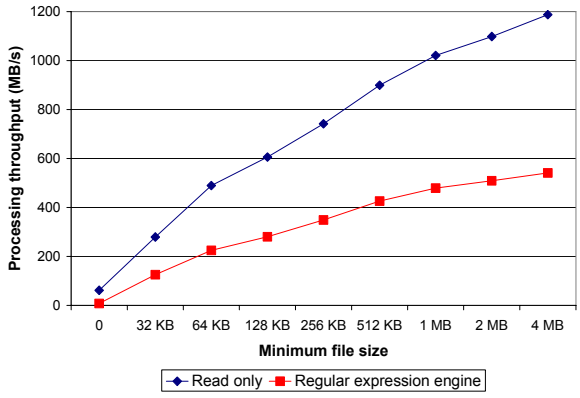

Figure 10: Processing throughput, platform 3+, 12x4 RAID0, xfs file system.

Even higher throughput can be achieved by extending platform 2 in a similar manner (adding an additional controller and 16 more drives). Here, the peak read throughput is 1.4 GB/s. An advantage to platform 2 is the fact that the two controllers each sit on independent HyperTransport links on the motherboard.

A point that should be noted here is the fact that for any configuration, the maximum throughput will always be limited by the bottleneck resource, and which resource ends up being the bottleneck is not always readily evident. In the case above (platform 2), one might expect close to 2 GB/s read throughput given that each individual controller is capable of at or near 1 GB/s on its own. When combined, however, the single HyperTransport link between the processor sockets becomes a bottleneck and the peak throughput ends up being limited to less than that predicted by a simple sum.

6.3 Write performance

Moving from read performance to write performance, the remaining experiments depart from the original data set of Figure 3 and Figure 4 and move to the synthetic data set comprised of large individual files. As a result, the meta-data processing has essentially been removed from the equation and does not impact the measured performance.

Figure 11 plots the write rate on platform 3 as a function of the position on the disk. Note that

position is only known approximately, as indicated by logical block number.

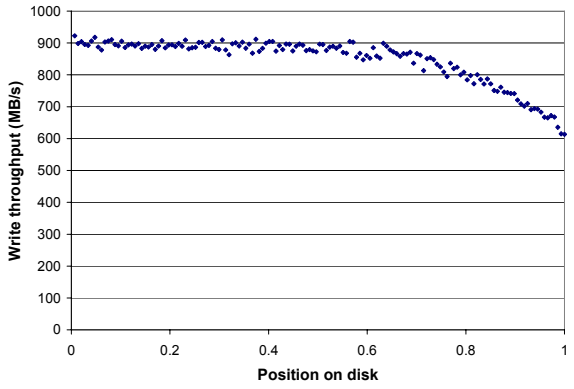

Figure 11: Write throughput, platform 3, 4x4 RAID0, xfs file system.

As we would expect, the performance of the outside tracks of the disks is better than the performance of the inside tracks. By the time one has made it completely to the interior, performance has dropped by approximately one third of that achievable on the outside tracks.

This trend is echoed with the same experiment using platform 1, shown in Figure 12. The pair of controllers present on platform 1 enables the peak write rate to be higher than that of platform 3. However, the trend of higher performance on the outside tracks and dropping performance towards the inside tracks continues.

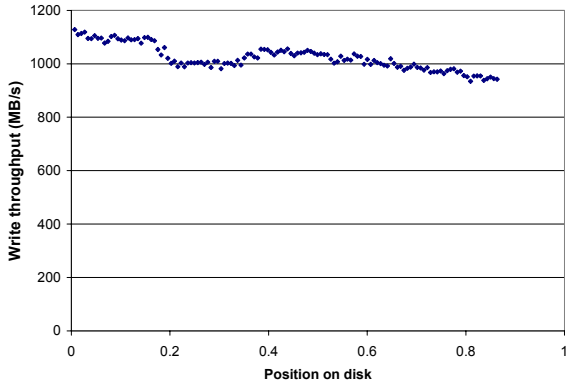

Figure 12: Write throughput, platform 1, 4x4 RAID0, xfs file system.

Table 2 shows the peak and mean read rates and write rates as a function of the number of logical drives. The experiment is run on platform 3, with a fixed number (16) of physical disks.

Table 2: Impact of number of logical drives on throughput, platform 3, RAID0, xfs file system.

no. of logical drives	peak read rate (MB/s)	mean read rate (MB/s)	peak write rate (MB/s)	mean write rate (MB/s)
1x16	779	766	753	619
2x8	917	899	861	820
4x4	1,032	972	930	848

The results show that organizing the disk subsystem as a single logical drive has lowest performance, while the greatest performance is achievable with higher logical drive counts. This is primarily due to the ability to queue greater numbers of pending requests to a set of drives rather than a single drive. Clearly, the ability to exploit this property will depend upon the control one has as to where files are placed in a file system (or set of file systems). This level of control is more likely to be reasonably achieved in an appliance than in a general-purpose system.

Table 3 examines the two RAID configurations, showing both peak and mean read rates and write rates for a 4 logical drive organization using platform 2.

Table 3: Impact of RAID configuration on throughput, platform 2, 4 logical drives, xfs file system.

disk config.	peak read rate (MB/s)	mean read rate (MB/s)	peak write rate (MB/s)	mean write rate (MB/s)
RAID0	1,306	1,285	1,309	1,235
RAID5	1,150	1,127	312	273

For read performance, the implications of moving from RAID0 to RAID5 are fairly minor, incurring approximately a 10% overall degradation in read throughput. The implications on write performance, however, are much more severe. When configured as RAID5, the write throughput is less than one fourth of that achievable as RAID0.

We finish with an investigation of the bottlenecks imposed by individual SAS channels used to connect the controllers and the disks themselves. Here, platform 2 is used in a RAID0 configuration and a 4x8 logical drive organization. Two of the logical drives (0 and 1) are physically allocated to the top Xtore enclosure (see Figure 1) and the remaining two logical

drives (2 and 3) are physically allocated to the bottom Xtore enclosure.

Table 4 shows the achievable throughputs (both read and write) when using 2 of the 4 logical drives concurrently and leaving the remaining two idle (in effect, using it as a 2x8 organization).

Table 4: Impact of sharing SAS channels, platform 2, 2x8 RAID0, xfs file system.

logical drives	read rate (MB/s)	write rate (MB/s)
0,1	712	676
0,2	1,085	801

Clearly, when both logical drives share the 4 SAS channels to the top enclosure, the achievable throughput is less than when the two logical drives are spread across 8 channels. This is true even with the fact that only one controller is in use. In this case, it is the channels between controller and physical disk that limit performance.

7. Conclusions

Many high-throughput, data-intensive applications are fundamentally limited by the performance sustainable by the disk subsystem. Here, we have shown the impact of minimum file size, meta-data processing requirements, logical drive organization, and RAID configuration on the sustainable performance of a set of high-capacity, direct-attached disk subsystems.

Generally, the xfs file system exceeds the performance of the ext3 file system. This is primarily due to better meta-data processing times. When meta-data processing is eliminated as a consideration, the top-end read throughput of both file systems are very close to one another.

Additionally, the larger, 32-disk subsystems are less susceptible to perturbations in performance due to secondary considerations (such as placement of data on exterior versus interior tracks). This is at least partially because the greater number of physical disks helps mitigate the performance implications of individual disk constraints (e.g., seek times) and partially due to the fact that the SAS switch limits the data rate from each individual drive, further diminishing the performance sensitivity due to each drive.

Across the board, improved performance is seen with increasing number of logical drives. While informative, this fact is less straightforward to exploit, since in many applications the system designer does not have control over how drives are logically organized. Appliance-class systems are one counter example to this, and our systems are of this type.

As a general rule, a RAID5 configuration incurs only a small performance penalty relative to a RAID0 configuration for read performance. However, write performance is significantly impacted, with overall write rates one quarter that of a RAID0 configuration.

While the achievable performance of direct-attached disk subsystems is quite high, that performance is highly susceptible to degradation due to a number of factors. It is only by careful attention to all the pertinent factors can maximum performance be sustained.

Direct-attached disk subsystems clearly now have the capacity and potential performance previously only seen in much larger SAN-based environments. Exploring the capabilities of this class of data storage is important, especially for dedicated appliances that need to deliver high-throughput, data-intensive results.

References

[1] R.D. Chamberlain, R.K. Cytron, M.A. Franklin, and R.S. Indeck, "The *Mercury* System: Exploiting Truly Fast Hardware for Data Search," in *Proc. of 1st Int'l Workshop on Storage Network Architecture and Parallel I/Os,* September 2003, pp. 65-72.

[2] M.A. Franklin, R.D. Chamberlain, M. Henrichs, B. Shands, and J. White, "An Architecture for Fast Processing of Large Unstructured Data Sets," In *Proc. of 22nd Int'l Conf. on Computer Design*, Oct. 2004.

[3] R.D. Chamberlain, B. Shands, and J. White, "Achieving Real Data Throughput for an FPGA Co-Processor on Commodity Server Platforms," in *Proc. of 1st Workshop on Building Block Engine Architectures for Computers and Networks*, Oct. 2004.

[4] R.D. Chamberlain and B. Shands, "Streaming Data from Disk Store to Application," in *Proc. of 3rd Int'l Workshop on Storage Network Architecture and Parallel I/Os*, September 2005, pp. 17-23.

[5] R.D. Chamberlain and R.K. Cytron, "Novel Techniques for Processing Unstructured Data Sets," in *Proc. of IEEE Aerospace Conference*, March 2005.
[6] R.D. Chamberlain and B. Shands, "Disk Subsystem Performance, Empirical Measurements." Technical Report, Dept. of Computer Science and Engineering, Washington University, 2007.
[7] B.C. Brodie, R.D. Chamberlain, B. Shands, and J. White, "Dynamic Reconfigurable Computing," in *Proc. of 9th Military and Aerospace Programmable Logic Devices Int'l Conference*, September 2006.
[8] Q. Zhang, R.D. Chamberlain, R S. Indeck, B. West, and J. White, "Massively Parallel Data Mining Using Reconfigurable Hardware: Approximate String Matching," in *Proc. of Workshop on Massively Parallel Processing*, April 2004.
[9] B.C. Brodie, R.K. Cytron, and D.E. Taylor, "An Architecture for High-Throughput Regular-Expression Pattern Matching." in *Proc. of 33rd Int'l Symp. on Computer Architecture*, June 2006.