J Supercomput (2015) 71:761–780
DOI 10.1007/s11227-014-1321-8

Understanding I/O workload characteristics of a Peta-scale storage system

Youngjae Kim · Raghul Gunasekaran

Published online: 11 November 2014

Abstract Understanding workload characteristics is critical for optimizing and improving the performance of current systems and software, and architecting new storage systems based on observed workload patterns. In this paper, we characterize the I/O workloads of scientific applications of one of the world's fastest high performance computing (HPC) storage cluster, Spider, at the Oak Ridge Leadership Computing Facility (OLCF). OLCF flagship petascale simulation platform, Titan, and other large HPC clusters, in total over 250 thousands compute cores, depend on Spider for their I/O needs. We characterize the system utilization, the demands of reads and writes, idle time, storage space utilization, and the distribution of read requests to write requests for the Peta-scale Storage Systems. From this study, we develop synthesized workloads, and we show that the read and write I/O bandwidth usage as well as the inter-arrival time of requests can be modeled as a Pareto distribution. We also study the I/O load imbalance problems using I/O performance data collected from the Spider storage system.

1 Introduction

Parallel file systems have been widely adopted on large-scale storage systems providing high I/O throughput, over 100 GB/s to 10 TB/s, and storage capacity of several

The preliminary version of the paper was published in the Proceedings of the 5th PDSW (Parallel Data Storage Workshop).

Y. Kim (✉) · R. Gunasekaran
National Center for Computational Sciences, Oak Ridge National Laboratory,
Oak Ridge, TN, USA
e-mail: kimy1@ornl.gov

R. Gunasekaran
e-mail: gunasekaranr@ornl.gov

peta-bytes. Typically, these large-scale storage systems use tens to hundreds of storage servers, each with tens to hundreds of disks, to guarantee the performance and capacity needs of peta-scale scientific applications. The parallel file systems distribute individual files over subsets of disks to improve single file performance. The Oak Ridge Leadership Computing Facility (OLCF)'s *Spider* file system serves as a center-wide storage resource for all compute systems including Titan, currently ranked second in the Top 500. The Spider II system is being deployed with a capacity of 32 PB from tens of thousands of spindles for an aggregate peak throughput of 1 TB/s. However, Spider II can achieve its peak, only when all the spindles operate at their maximum speed for an ideal, sequential I/O workload. Moreover, the storage systems are heavily shared resources, multiple users running distinct scientific applications with diverse I/O patterns use the storage resource simultaneously, resulting in throughputs much lower than the expected peak capacity. Also, hotspots (i.e., congestion, multiple users accessing the same resource pool) occur in a subset of disk volume groups, which results in an I/O load imbalance over the disk volumes.

Resolving I/O load imbalance is a challenging problem in parallel storage systems [8]. To corroborate the I/O load imbalance problem, we observed the I/O workload on 24 RAID controllers, which is a single file system partition of the Spider storage system. Figure 1 presents the I/O activity at individual controllers' in terms of the usage bandwidth observed at two different time periods. We can infer from the plot that a few controllers are overloaded while others are idle. The details on how we collected the data will be explained later in Sect. 2.2 Thus, one slow (over utilized) disk volume can significantly decrease the performance of a bulk parallel, synchronous I/O workload. Most parallel file systems balance I/O loads in terms of data volume, enforcing all disk volumes to grow at the same rate, with the constraint that individual files can only use a subset of storage servers (data striping). As a result, a few disk volumes are overloaded by a larger number of read and write requests. Caching and buffering can ease I/O load imbalance to certain degree, but cannot resolve the problem completely. System and load aware scheduling schemes can be used to alleviate the problem of overloading disk volumes; however, in HPC environments, it is very difficult to estimate I/O workloads and data stripping requirements.

For a comprehensive understanding of HPC workloads, we characterize the workload of Spider file system, a Lustre parallel file system [15], which hosts data for

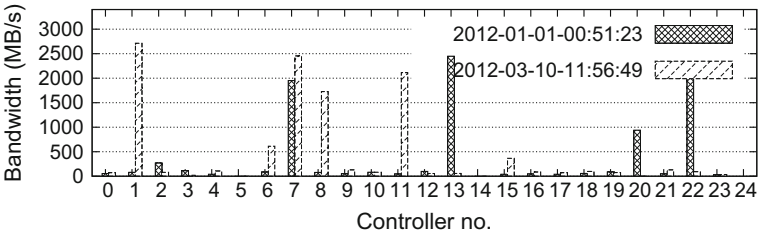


Fig. 1 Snapshot of individual controller's bandwidth for two different times. We show the statistics of performance data for each day above with (Min, Max, Avg, STD, Date). (0.0, 2449.0, 373.2, 724.6, 2012-01-01), (0.0, 2713.0, 465.5, 822.6, 2012-03-10)

scientific jobs running on Titan, and a host of other compute infrastructures at Oak Ridge National Laboratory (ORNL). In general, workload or scientific application characterization efforts have been carried out by collecting data by instrumenting client applications or enabling RPC trace at the backend servers. This data collection results in considerable overhead (at least 2 %), and the data collection process directly impacts the core process running on the system. Moreover, collecting such detailed trace data requires non-trivial development and integration effort by the application developer. This restricts the trace data collection method for development and debugging purposes only.

The data we used for our characterization effort are from the RAID controllers, queried using their custom API and the data are collected out of band on the management network. Also, the data we collect from the controllers are sufficient for workload characterization and provide us with an estimate of application I/O workloads. In Yang et al. [7], we correlated the controllers data with the schedulers log to get an estimate of I/O activity for users with jobs that have an identical and repetitive pattern, which is a parallel effort to our workload characterization study. Our study of the peta-scale storage system workloads have the following merits.

- Our study is based on real-system data that provide useful insight into the I/O activity as well as system requirements based on the workload, in particular I/O requirements. Our observation and analysis will provide invaluable design guidelines towards building large-scale storage systems for scientific workloads.
- We also synthesize workloads, finding a mathematical function that can generate similar synthetic workloads. From the results of our workload synthesis process, we found that the bandwidth distribution can be modeled as a Pareto distribution.
- We identified the I/O load imbalance problem in our peta-scale storage systems. And we investigated the problem of load imbalance over disk volume groups on our shared storage systems.

2 The storage cluster

2.1 Overview of Spider storage system

Spider (active since 2008) is a Lustre-based storage cluster of 96 DDN S2A9900 RAID controllers (henceforth referred to as controller) providing an aggregate capacity of over 10 petabytes from 13,440 1-terabyte SATA drives. The overall Spider architecture is illustrated in Fig. 2. Each controller has 10 SAS channels through which the backend disk drives are connected. The drives are RAID 6 formatted in an 8 + 2 configuration requiring disks to be connected to all the ten channels. In our current configuration, we connect 14 disks per channel, thereby each controller is provisioned with 14 tiers. Controllers are paired together forming a couplet, failover configuration, and overall each couplet has 28 RAID 6 8 + 2 tiers. Each controller has two dual-port 4× DDR IB HCAs for host side connectivity. Access to the storage is through the 192 Lustre Object Storage Servers (OSS) connected to the controllers over InfiniBand. Each OSS is a Dell dual-socket quad-core server with 16 GB of memory. Four OSSs connect to a single couplet with each OSS accessing 7 tiers. The OSSs are also con-

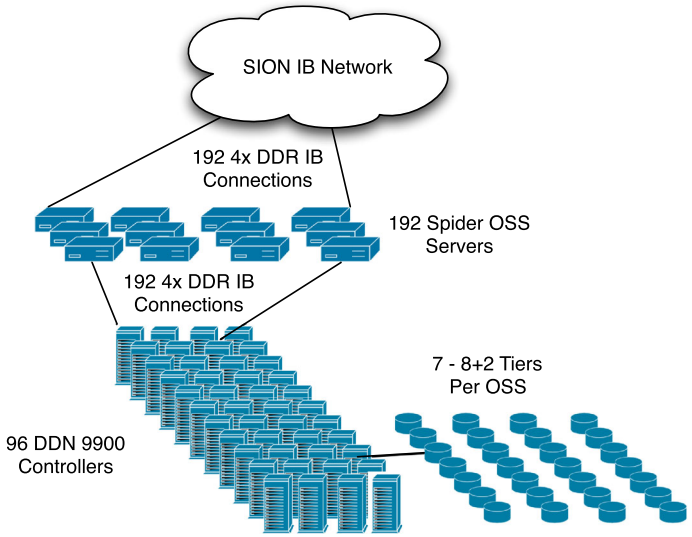


Fig. 2 Spider storage system architecture [14]

figured as failover pairs and each OSS connects to both controllers in the couplet. The compute platforms connect to the storage infrastructure over a multistage InfiniBand network, referred to as SION (Scalable I/O network), connecting all OLCF compute resources.

Currently, Spider-I has been upgraded to Spider-II, which is also a center-wide parallel file system to support a peak bandwidth of up to 1 TB/s. Architectures of both file systems are very similar, for the study in this paper, we used data collected from the Spider-I system.

2.2 Data collection

For characterizing workloads, we collect I/O statistics from the DDN S2A9000 RAID controllers. The controllers have a custom API for querying performance and status information over the network. A custom daemon utility [11] periodically polls the controllers for data and stores the results in a MySQL database. We collect bandwidth and input/output operations per second (IOPS) for both read and write operations in 2-s intervals. We measure the actual I/O workload in terms of the number of read/write request with the size of the requests. The request size information is captured in 16 KB intervals, with the smallest request less than 16 KB and the maximum being 4 MB. The request size information is sampled approximately every 60 s from the controller. The controller maintains an aggregate count of the requests serviced with respect to size from last system boot, and the difference between two consecutive sampled values will be the number of requests serviced during the time period.

3 Characterizing workloads

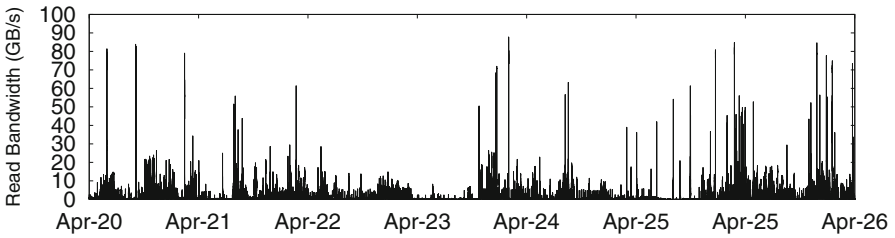
We studied the workload of our storage cluster with data collected from 48 RAID controllers over a period of 6 months (January 2010–June 2010). This partition is one half of our total capacity (max bandwidth is 120 GB/s), and is representative of our overall workload. We characterize the data in terms of the following system metrics.

- *I/O bandwidth distribution* helps understand the I/O utilization and requirements of our scientific workloads. Understanding workload patterns will help in architecting and designing storage clusters as required by scientific applications:
- *Read to write ratio* is a measure of the read to write requests observed in our storage cluster. This information can be used to determine the amount of partitioned area required for read caching or write buffering in a shared file cache design.
- *Request size distribution*, essential towards understanding and optimizing block device performance. Request sizes can range from 16 KB to 4 MB, impacting the overall filesystem throughput. The underlying device performance is highly dependent on the size of read and write requests, and correlating request size with bandwidth utilization will help understand device characteristics.
- *Inter-arrival time distribution* provides us with an estimate of time between requests, and can be used to characterize the arrival rate of read and write requests.
- *Idle time distribution* provides an estimate of idle time between bursts of requests. This information is useful for initiating background services such as disk-scrubbing [10], without interfering with the foreground service.
- *Storage capacity utilization*, representative of our ever increasing storage demand and is important for provisioning storage systems in the future.
- *Load imbalance*, a measure of I/O load distribution across storage servers. High load imbalance results in low throughput and underutilization of the file system. Understanding load imbalance helps in the design of I/O aware smart tools for uniform load distribution.

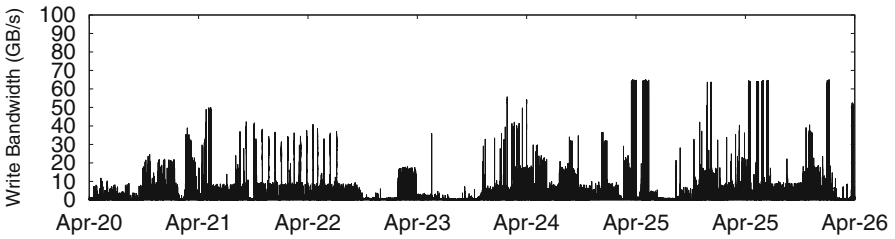
In addition to these metrics, we project future storage demand, fitting a linear regression model on current usage trends. Also, we conducted an analysis of the daily and weekly effects of HPC workloads.

4 Bandwidth distribution

An understanding of read and write I/O characteristics is critical for architecting and provisioning systems, guaranteeing users an effective throughput. Figure 3 shows the filesystem usage in terms of bandwidth for a week in the month of April 2010. This is representative of our normal usage patterns for a mix of scientific applications on our compute clients. The bandwidth numbers are obtained by summing the observed usage across 48 controllers. We have observed a maximum of around 90 GB/s for reads and around 65 GB/s for writes. Also, we can infer from the plot that the arrival patterns of I/O requests are bursty and the I/O demands can be tremendously high for short periods of time.



(a) Read



(b) Write

Fig. 3 Observed I/O bandwidth usage for a week in April, 2010

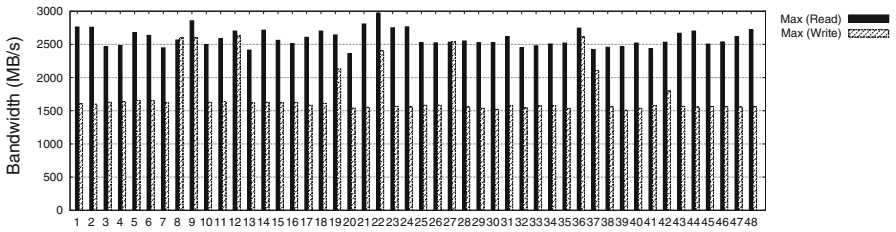
From the 6 months of performance data, we generated the cumulative distribution function (CDF) plot of the bandwidth provided by individual controllers. From the CDF plot, we extracted the 95th, 99th, and 100th (max) percentile of read/write bandwidth, refer to Fig. 4.

We observe a large difference between 95th and 99th percentiles and between 99th and 100th percentile values of the bandwidth. For example, in controller 1, we observe 115 and 153 MB/s for 95th percentiles of read and write bandwidth, respectively, and for the 99th percentile we observe 508 MB/s of read and 803 MB/s of write bandwidth. 99th percentile of read is 4.41 times higher than its 95th percentile while 99th percentile of write is 5.24 times higher than its 95th percentile. For 100th percentile, we observe that peak read bandwidth can reach 2.7 GB/s, and for the write it peaks at 1.6 GB/s. This bandwidth distribution is representative of a heavy long-tail distribution, and we see that these trends are observed across all our controllers.

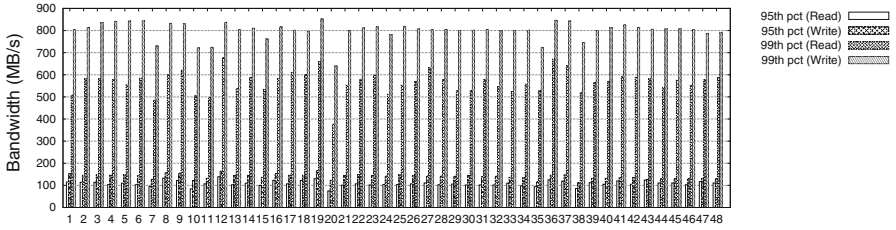
Interestingly, we observe the 95th and 99th percentile values of the write bandwidth is higher than read bandwidth; however, for the 100th percentile values, the read bandwidth is higher than write bandwidth.

4.1 Modeling I/O bandwidth distribution

We provide a mathematical model for the bandwidth usage for synthesizing workloads. The gradient of the slope indicates that the distribution is mostly likely to be a power law distribution or a long-tailed distribution. The Pareto model is one of the simplest long-tailed distribution models, and we use it as the model for our bandwidth distribution. The cumulative distribution function of a Pareto random variable is defined as:



(a) Peak bandwidth



(b) 95th and 99th percentiles bandwidth

Fig. 4 Percentile distribution of the observed read and write bandwidth usage recorded at 48 controllers. The *number* on x-axis shows the no. of controller. *pct* percentile

$$F_X(x) = \begin{cases} 1 - \frac{x_{\mathrm{m}}^{\alpha}}{x} & \text{for } x \geq x_{\mathrm{m}} \\ 0, & \text{for } x < x_{\mathrm{m}}, \end{cases} \tag{1}$$

where x_{m} is the minimum positive value for x and α is referred to as the shape parameter.

Figure 5 plots the observed and modeled data for controller 1. As we have similar observations with other controllers, we only present the results for one controller, which is representative of our overall workloads. For the current read bandwidth distribution α is 1.24. The measure of fit is evaluated using the R^2 goodness-of-fit coefficient. It was found to be 0.98 for read bandwidth. Similarly, the write bandwidth distribution was matched to a Pareto model with an α value of 2.6, giving a fitness coefficient of 0.99.

Observations Read write bandwidth across all controllers follows a long-tail distribution. Read peak bandwidth is much higher than write peak bandwidth; however, a majority of bandwidth peaks observed are for writes than reads (e.g., 95–99 percentiles of bandwidth). Also, high variation in peak bandwidth is observed across controllers.

4.2 Aggregate versus peak bandwidth

In Fig. 6, we compare the aggregate bandwidth of 48 controllers for different percentile values and the sum of max bandwidth observed at each of the 48 controllers. From the plot, we see that the aggregate bandwidth is much lower than simple summation

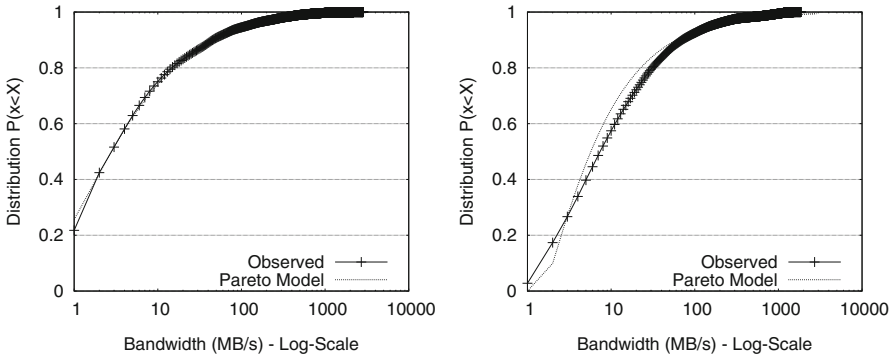


Fig. 5 Pareto model—I/O bandwidth for controller 1

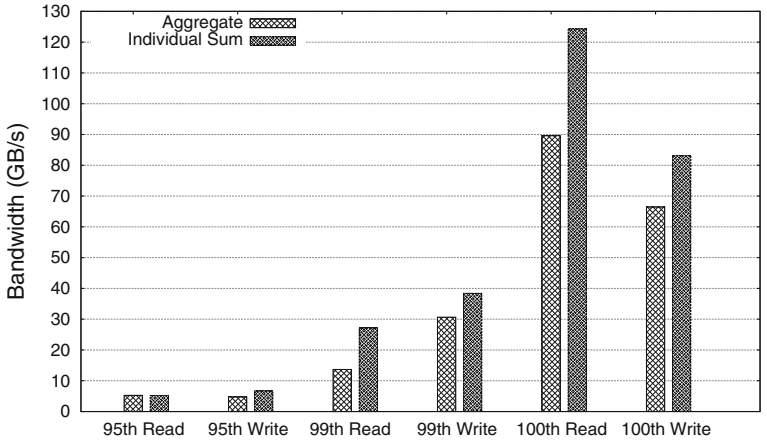


Fig. 6 Aggregate bandwidth. In the legend, *aggregate* denotes the aggregate bandwidth of 48 controllers and *individual* denotes a sum of bandwidths individually observed from 48 controllers

of individual controller's bandwidth at 99th and 100th percentiles. We infer that the peak bandwidth of every controller is unlikely to happen at the same time.

Observations Peak bandwidth utilization for every controller is unlikely to happen at the same time. Read bandwidth is more unlikely to happen at the same time than write bandwidth for 99th and 100th percentiles of bandwidth.

5 Read to write ratio

Typically scientific storage systems are thought to be write dominant; this could be attributed to the large number of checkpoints for increased fault tolerance and journaling requests. However, in our observation, we find only a marginal difference between the read and write workload.

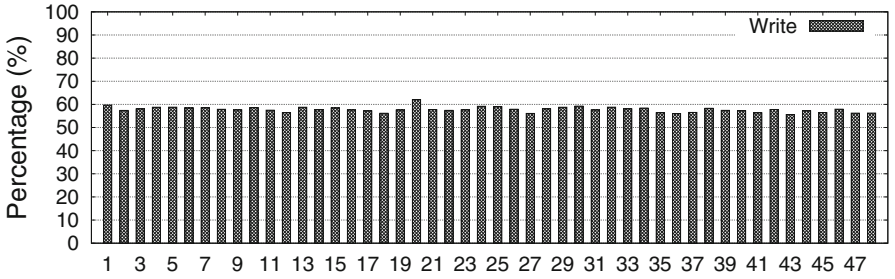


Fig. 7 Percentage of write requests observed at the controllers. The number on x-axis show the no. of controller

Figure 7 presents the percentage of write requests with respect to the total number of I/O requests in the system. The plot is derived by summing the read and write data volume observed during the 6-month period. We observe that the write requests are over 50 % across all controllers, the remainder being read requests. We see that the difference is marginal, this could be attributed to the center-wide shared file system architecture of Spider, supporting an array of computational resources, and the movement of data across file system partitions or the HPSS archival storage system.

Observations Read requests are 42.2 % on average, and write at 57.8 %, which is contrary to the general perception that HPC storage systems are write intensive.

6 Request size distribution

6.1 Request size

On our Spider storage system, request sizes vary from less than 16 KB to a max of 4 MB, and we aggregate statistics of the number of requests in 16 KB intervals for both read and write operations. Figure 8 shows the distribution of request sizes, ranging from 16 KB to 1.5 MB, request sizes beyond 1.5 MB account for only a small fraction and hence not plotted. In Fig. 8a, we observe three peaks at less then 16 KB, 512 KB and 1 MB, which account for more than 95 % of the total requests. From Fig. 8b, we can infer that 50 % of writes and about 20 % of read requests are smaller than 16 KB. However, for requests less than 512 KB, we observe that reads are almost twice the number of write requests. For request sizes at 1 MB, 25 % of the requests are reads and 30 % are writes.

The request sizes cluster near 512 KB boundaries because of an issue in the Linux block layer. *DAM-Multipath* is a virtual device driver that provides link redundancy and aggregation features. To ensure it never sends requests that are too large for the underlying devices, it uses the smallest maximum request size from those devices. If all of those devices support requests larger than 512 KB, it uses that size as its maximum request size. The lower lever devices are free to merge the 512 KB into larger requests, but that generally requires queue pressure, i.e., having more outstanding requests for the device than it can process concurrently. Lustre tries to send 1 MB requests to storage when possible, thus providing frequent merge opportunities under load.

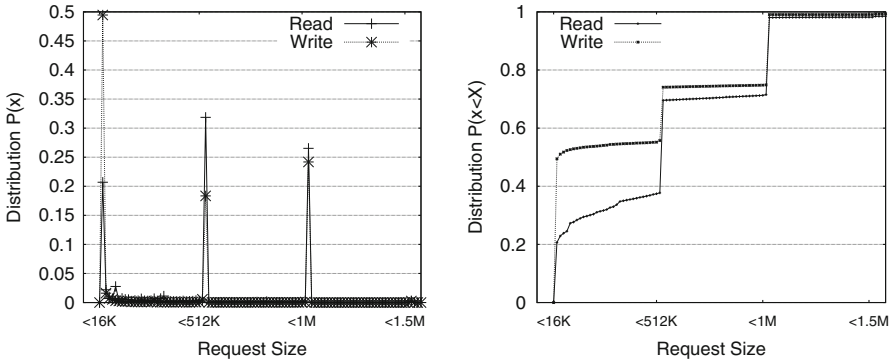


Fig. 8 Distribution of the request size. **a** Probability density function (PDF) and **b** cumulative density function (CDF)

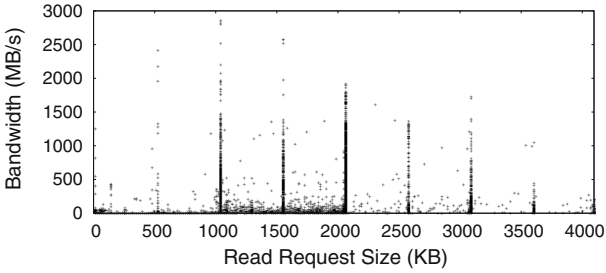
Observations Most request sizes are either less than 16 KB or between half a megabyte and 1 MB. Generally in PFS, metadata operations are often small. Also, Lustre block layer has been optimized to cluster near 512 KB boundary, and our default Lustre stripe size is 1 MB.

6.2 Correlating request size and bandwidth

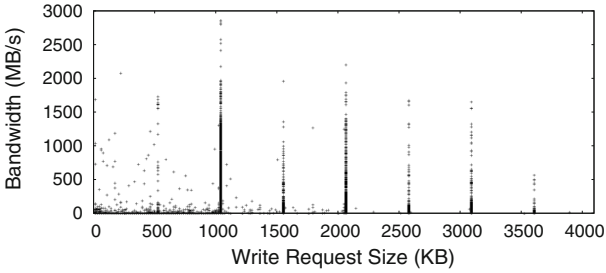
In addition to understanding I/O bandwidth demands in terms of read and write operations, it is important to investigate the correlation of request size to I/O bandwidth. Figure 9 is a scatter plot relating request size and the observed bandwidth at the time of read and write operations. Each point is represented by a combination of $(request\ size, bandwidth)$ based on data collected from 2010-04-20 to 2010-05-24 on controller 1. We used the 99th percentile of each X and Y value in the plot. Since the sampling rate of bandwidth data was 2 s while that of request size distribution was around 60 s, each data point shown in Fig. 9 is a value obtained at 99th percentile in 60-s intervals. Under the assumption that larger request sizes are more likely to lead to higher bandwidth during a time interval, we see that the peak bandwidth will be attained at 1 MB size requests.

6.3 Inter-arrival and idle time

In our analysis of inter-arrival time, every controller shows very similar arrival patterns. In Fig. 10, for read and write we observe peaks at different inter-arrival time. For reads, 2 ms period has the highest probability measure, which implies that we have a high read request rate with each request arriving every 2 ms. For writes, we observe that requests are placed at 4-ms intervals. However, from Fig. 10, we observe that about 90 % of write requests have around 10–11 ms of inter-arrival time while that of read

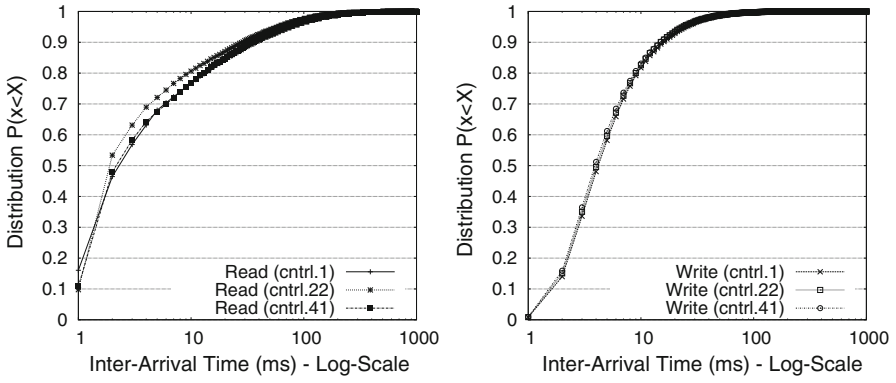


(a) Read bandwidth vs. request size



(b) Write bandwidth vs. request size

Fig. 9 Correlation between I/O bandwidth and request size



(a) Read **(b)** Write

Fig. 10 Distribution of inter-arrival time for three different controllers

requests do have around 13–14 ms of inter-arrival time. Overall the arrival rates of read and write requests are very intense.

Similar to the bandwidth distributions, we see that the inter-arrival time distribution also follows a long-tailed Pareto distribution, as shown in Fig. 11. The read inter-arrival distribution can be modeled as a Pareto distribution with an α of 1.17; similarly, the write idle time distribution can be modeled with an α of 1.72, respectively. The R^2

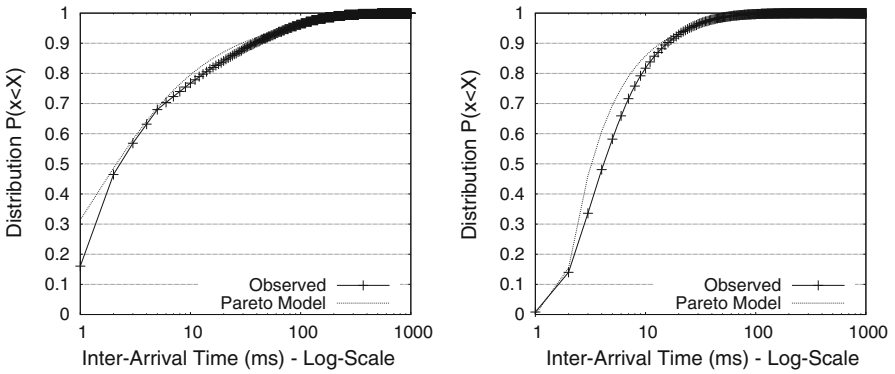


Fig. 11 Pareto model—inter-arrival for controller 1

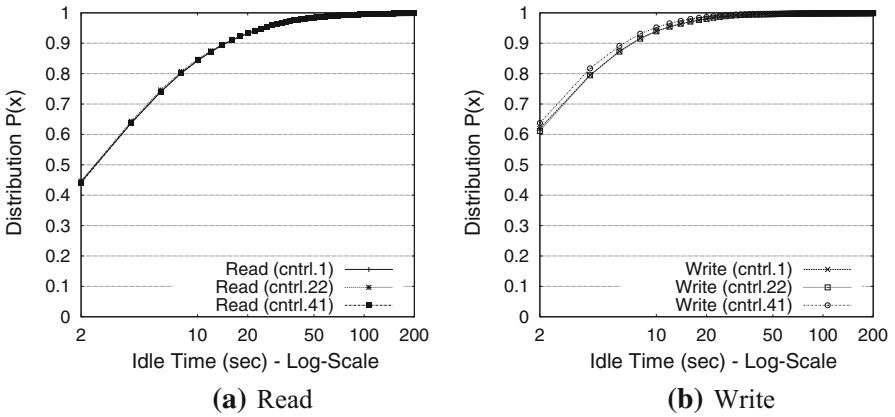


Fig. 12 Distribution of the idle time process

goodness-of-fit coefficient was 0.98 and 0.99 for read and write inter-arrival distributions, respectively.

We define idle time as the period when there is no user I/O request and the controller performs background services such as verify and rebuild operations. Our calculation is limited by the 2-s sampling interval. Figure 12 shows the distribution of idle time for reads and writes. Similarly in the inter-arrival time distribution, we present data for a few controllers representative of our entire data set. In particular, from Fig. 12, we see that approximately 10 % of read requests have more than 10 s of idle time period between requests while about 10 % of write requests have more than 16 s of idle time period. Moreover, we do observe that there are idle time periods of more than 20 s between bursts of requests.

We see that the idle time distribution also follows a long-tailed Pareto model. Figure 13 plots the observed data and modeled data with a correlation coefficient of

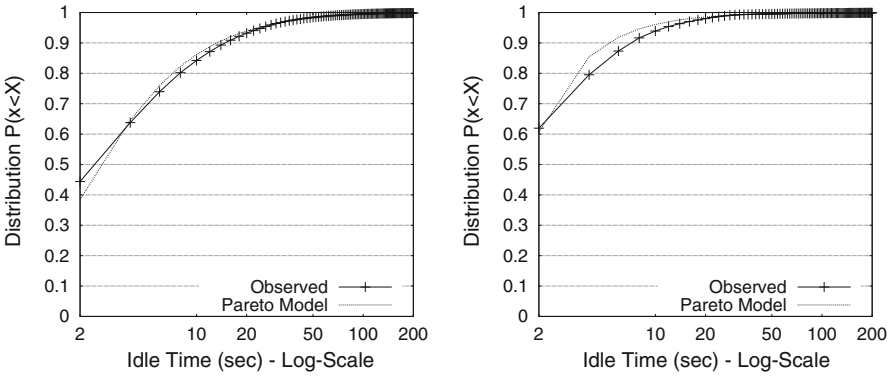


Fig. 13 Idle time—Pareto model

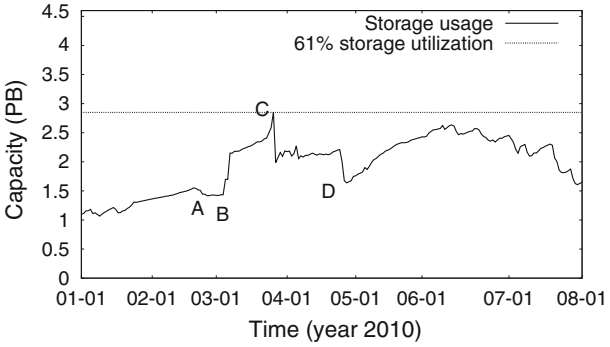


Fig. 14 Spider storage usage. Total storage capacity is 4.6 PB

0.98 for both read and write distributions for controller 1. The read idle time distribution can be modeled as a Pareto distribution with an α of 1.36; similarly, the write idle time distribution can be modeled with an α of 1.43, respectively.

Observations Inter-arrival time and idle time distributions both follow a long-tail distribution, and they can be modeled in a Pareto distribution.

7 Effects of time-of-day

7.1 Demand

The plot in Fig. 14 illustrates the usage of the spider storage cluster, for the partition in production that has a total capacity of 4.6 petabytes.

The data are collected on a daily basis using the *df* utility. The plot shown in the figure is representative of data generated by a petascale simulation environment and increasing trend shows the growing storage needs of a petascale environment.

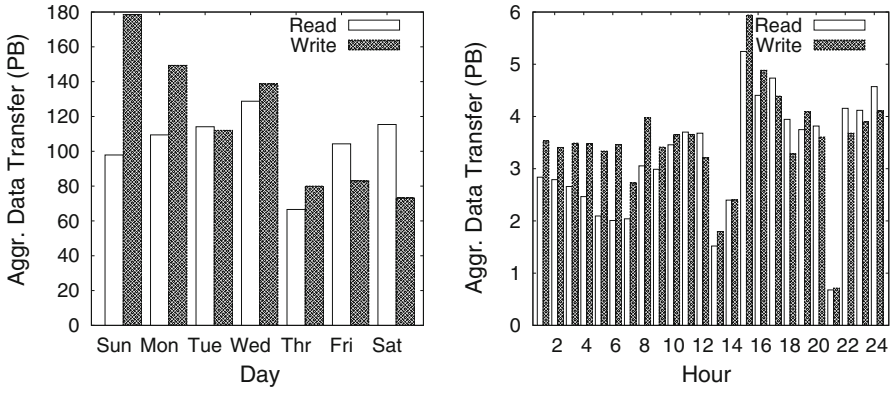
The storage cluster is not used for archiving purposes; the users are requested to use the space for computational needs only. The system administrator periodically runs a utility tool sweeping through the file system and deleting files that have not been touched in the past 2 weeks. Points *A* and *D* point to the slopes in the plot where the overall usage drops over a few day period; this is a result of the administrators sweep action. The slope *B* is an extreme case where there was a sudden increase in utilization over a period of few days, which is totally based on the application requirements. Similarly, we notice a spike in usage and drop the following day, labeled *C* in the plot, this is the requested behavior from a user. The user after a successful application run moves the data to a tape-based storage system, HPSS.

We also show that the storage usage can be projected as a first-order linear function based on 7 months of observation. Assuming that the growth rate of storage usage follows the linear function, the utilization of the Spider storage cluster can increase up to 90 % of 4.6 petabytes in less than a year.

Observations The utilization of storage space is well maintained below 60–70 % by periodically triggering sweep operations by system administrators.

7.2 Daily and weekly effects

The motivation is to find patterns in the day-to-day usage of the storage cluster or distinctive usage patterns in days of the week. A regular pattern of I/O traffic over time will help develop more efficient I/O scheduling policies and the regular maintenance schedules of the system can be done on periods with least user demand. As discussed earlier, average bandwidth usage is not a good measure of system utilization, so we try to identify the daily and weekly effect in terms of the total data transfer observed at the storage cluster during a period of time. The usage patterns for the daily and weekly effect in I/O bandwidth are from our observations for a period of 5 months (January 2010–May 2010), and are as shown in Fig. 15.



(a) Day of the week effect **(b)** Time of the day effect

Fig. 15 Each point shows aggregate bandwidth of 48 DDNs

In Fig. 15a, we observe that the usage on Thursday is the lowest for both reads and writes. This is expected as the primary compute infrastructure has a scheduled maintenance period every Thursday. However, writes shows decreasing usage from about 180 PB on Sunday to lower than 80 PB on Saturday. On the contrary, reads are more or less identical within a range of 100 PB and 130 PB all the week except for Thursday. From the plot, we can suggest Sunday is desirable for application that are not write intensive. Also we observe more available write bandwidth present between Thursday and Saturday than read bandwidth.

Figure 15b presents the time of the day effect. We observe that around noon the usage drops, and then starts increasing from 13:00 and has more or less constant usage for the next few hours. The read and write have almost identical usage pattern in the day. It is interesting to note that the lowest activity is around 9 pm in the day.

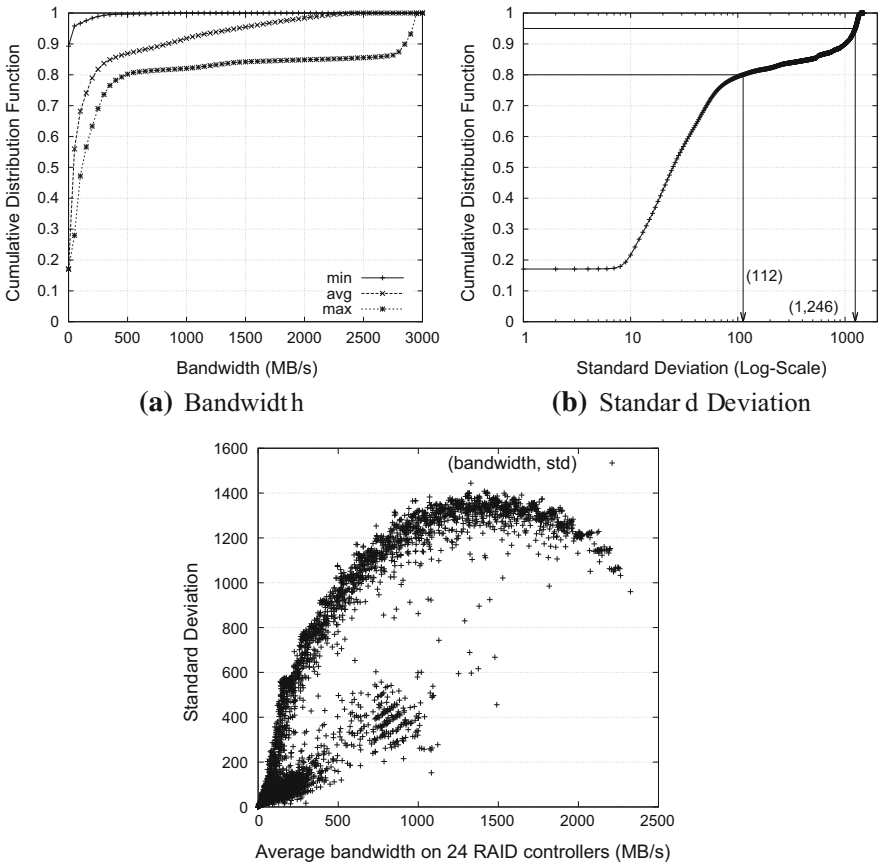
8 I/O load imbalance on the spider

In this section, we look into the I/O load imbalance problems in the storage systems for HPC workloads. We investigate the presence of intermittently congested storage servers and the duration of this congestion. We study the workload for 24 of the controllers over a period of 3 months (January–March 2012). This partition, an active lustre filesystem, is a quarter of our total storage system.

For a better understanding of the load imbalance problem we plotted the max, min and average I/O usage aggregated across all the controllers, the CDF shown in Fig. 16a. For calculating the aggregate I/O usage, we summed read and write bandwidth, and also accounted for I/O from background services, such as disk scrubbing for latent sector errors [10]. The RAID controller performs these background activities and these data are collected along with read/write bandwidth data in 2-s intervals. Previously, in Sect. 4, we modeled the I/O usage of max read and write at a specific controller, which is identical to the max plot in Fig. 16a. From the CDF plot, we extract the 95th percentile of the minimum, average, and maximum bandwidth: 21 MB/s, 1,430 MB/s, and 2,908 MB/s, respectively. This observation implies that the bandwidth distribution is highly likely to follow a heavy, long-tail distribution. The heavy-tail distribution can be found in many phenomena such as stock market crashes and earthquakes, which do not occur often but are clustered when they occur. Similarly, our observation implies that an individual controller would receive a burst of I/O demands in a very short time, overloading the controller. However, the CDF plot in Fig. 16a is not enough to explain I/O load imbalance on the storage system. Thus, we do an in-depth analysis of load imbalance on storage in the following section.

8.1 Load imbalance analysis with bandwidth

One of the metrics for evaluating I/O load imbalance is the standard deviation of the controllers' bandwidths. For a given time period i, the instant standard deviation (std), std_i is calculated by $\text{std}_i = \sqrt{\frac{\sum(x_{i,j}-\bar{x})^2}{n-1}}$ where j is an index for controller. If all the controllers' I/O load is evenly distribution at time i, the standard deviation will be



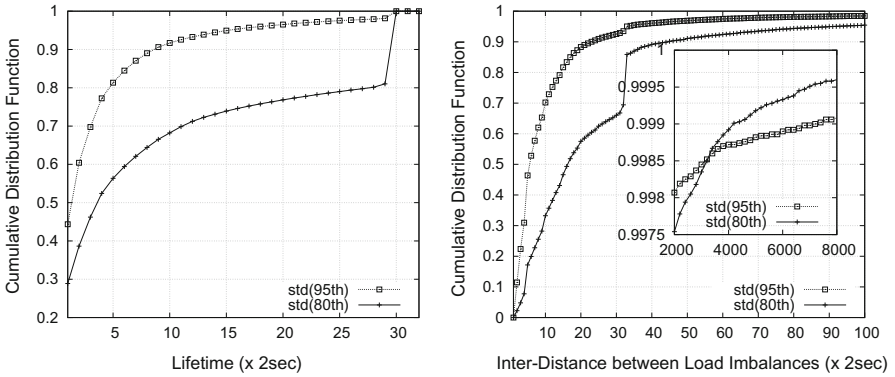
(a) Bandwidth **(b)** Standard Deviation

(c) Bandwidth and Dispersion Relationship

Fig. 16 Analysis results for I/O load imbalance on the Spider storage system

very low. Therefore, higher standard deviation leads to more I/O load imbalance. Figure 16b shows the CDF plot of the standard deviation of bandwidth of the controllers. Similar to the observation for bandwidth distribution, the standard deviation plot also shows a long-tail distribution. For example, from the CDF plot, we observe that the 80th percentile is 112 MB/s and the 95th percentile is 1,246 MB/s. This indicates that, over the 3 months, 80 % of the time the standard deviation was less than 112 MB/s, which is reasonably low.

Figure 16c presents a dispersion plot of bandwidth with standard deviation. The region with `std<200` accounts for about 82 % of the measurements during the 3 months (refer to Fig. 16b), which is fairly balanced compared to the rest. However, the rest on the region with `std > 200` shows significantly high deviation of bandwidth versus the region below 200. We observe that the standard deviation increases with respect to bandwidth. It shows either a log linear increase or a linear increase, which implies that higher bandwidth leads to more load imbalance. Interestingly, the standard



(a) Lifetime of LI **(b)** Distance betwee n LIs

Fig. 17 Lifetime and distance distribution of load imbalance

deviation decreases after the bandwidth is around 1,000 MB/s. It implies that most controllers are serving I/O requests, hence the high bandwidth, thereby reducing the I/O load imbalance.

8.2 Lifetime and inter-distance of load imbalance

The lifetime distribution of load imbalance period is essential to understanding and optimizing storage and file system performance. Above, we use standard deviation as the imbalance metric. For a given time period i, the instant standard deviation is smaller than the threshold, then the storage systems are fairly balanced; otherwise, the system is poorly balanced. We use two thresholds, which are 112 and 1,246 standard deviations corresponding to 80th and 95th percentiles in Fig. 16b. Figure 17a presents the lifetime distribution of load imbalance. Interestingly, we observe that the higher the threshold, the shorter is the lifetime. For example, for the 90th percentile, the lifetime for std 95th is around 16 s, whereas that for std (80th) is around 58 s. The load imbalance does not continue over long periods in our data analysis.

Inter-distance analysis provides an estimate of time between load imbalance periods. Interestingly, we see that shorter lifetime for std 95th shows a smaller inter-distance than for std 80th for 99 % of the time, which means that once the system is highly imbalanced, the system is likely to remain imbalanced. However, as we see from the zoom-in plot, the two lines cross around the 3,000 s. This indicates that the 95th data are more bursty than the 80th data in terms of I/O load imbalance duration. Note that the 95th shows a longer tail distribution than the 80th.

Observations We observe that (i) the higher bandwidth on average leads to more load imbalance on the storage, and (ii) the lifetime of load imbalance can extend up to 60 s, and if the loads are highly imbalanced, then the load imbalance is more likely to prolong.

9 Related work

A comprehensive understanding of system workloads will not only aid in architecting new storage systems with enhanced performance and capabilities, but also be very useful for storage controller, network, and disk subsystem designers. A number of studies have characterized and analyzed I/O workloads for server systems.

I/O profiling tools can characterize I/O patterns of applications in HPC systems. Darshan [4], Tracefs [1], TRACE [9], and ScalaHTrace [16] are examples of such tracing tools. These tracing tools in general trace I/O systems calls in time, and produce raw outputs of every call and time, or summarizes statistics with I/O calls. blktrace [3] is a block level I/O tracing tool to characterize I/O patterns on the storage without file system impact. All these tools are used to understand the impact of applications I/O on the storage system. On the other hand, our server-side tracing tool is lightweight, because while existing tools will invoke non-negligible tracing overhead because it has to be run with an actual code, ours is out of band from the applications, which becomes no tracing overhead.

Many prior works [4,6,13,17] attempted to understand I/O characteristics of applications on the storage systems. Zhang et al. [17] conducted a comprehensive study on synthesizing enterprise workloads, such as TPC-H I/O workloads. Alma et al. [13] characterized disk-level traces of enterprise system at three different levels of time granularities—millisecond, hour, and lifetime of the disk. Kavalanekar et al. [6] analyzed storage traces collected from various production servers at Microsoft. These studies characterized enterprise-scale I/O workloads. I/O workload characterization in social network environment is also attractive, so many prior works [2,5] studied the networked I/O characteristics in such an Internet environment. However, there has been very limited studies on the characterization and analysis of workloads on storage systems supporting high-end computing systems, specifically scientific workloads on peta-scale compute platforms. Recently, Carns et al. [4], studied application behavior with respect to I/O activities on a petascale storage system supporting IBM BlueGene/P system [4]. Frank et al. [12] used ScalaTrace analyzed MPI and IO events at scale.

Our work can be complementary to prior works [4]; however, it has several key contributions over them. (i) We used data collected at a storage controller for our I/O workload characterization, which none of prior works used for I/O storage characterization study. (ii) Our characterization is not an application-specific study, which has been exploited in many literatures. Instead, our study is based on aggregated I/O data of multiple applications, which allows to collectively study I/O impact on the entire storage system. We used data collected over a period of 6 months, which we believe is sufficient to present useful observations and interpretations. (iii) We also identified load-imbalance issues on the shared storage system from the observed data.

10 Conclusion

We characterized and analyzed the I/O workloads of a leadership class storage cluster, Spider, from I/O stats data collected the RAID controllers. Our findings from the

workload characterization are summarized as: (i) max bandwidth is much higher than 99th percentile bandwidth, (ii) peak bandwidth occurs at 1 MB of request size, (iii) read requests are not small, (iv) inter-arrival time and bandwidth usage follow a Pareto distribution; higher bandwidth observed tends to cause load-imbalance across storage servers. We have identified several venues for future study. Our current study does not include analysis of data locality because we have collected only performance statistics. We plan to collect block-level and RPC (remote procedure call) trace information from the OSS servers and extend our performance statistics-based analysis to be able to perform block-level data locality analysis of applications. The analysis of the block level traces and RPC logs will help infer individual applications behavior and help profile applications I/O access patterns.

The analysis in this work can be used for architecting and provisioning storage systems resources in terms of capacity planning when building a hierarchical storage system using SSDs and HDDs. The synthesized mathematical studies were used to develop test suites for evaluating next generation storage systems, which we used extensively to test new storage servers and devices when architecting the Spider storage. Finally, the observations detailed in this work motivate the development of I/O-aware scheduling algorithms to avoid storage server or disk congestion problems.

Acknowledgments This research is sponsored by the Office of Advanced Scientific Computing Research, U.S. Department of Energy and used resources of the Oak Ridge Leadership Computing Facility, located in the National Center for Computational Sciences at Oak Ridge National Laboratory, which is supported by the Office of Science of the Department of Energy under Contract DE-AC05-00OR22725.

References

1. Aranya A, Wright CP, Zadok E (2004) Tracefs: a file system to trace them all. In: Proceedings of the 3rd USENIX conference on file and storage technologies, FAST '04. USENIX Association, Berkeley, pp 129–145
2. Arlitt MF, Williamson CL (October 1997) Internet web servers: workload characterization and performance implications. IEEE/ACM Trans Netw 5(5):631–645
3. Axboe J (2007) blktrace User Guide. http://www.cse.unsw.edu.au/~aaronc/iosched/doc/blktrace.html
4. Carns P, Harms K, Allcock W, Bacon C, Lang S, Latham R, Ross R (2011) Understanding and improving computational science storage access through continuous characterization. Trans Storage 7(3):8:1–8:26
5. Jeon M, Kim Y, Hwang J, Lee J, Seo E (2012) Workload characterization and performance implications of large-scale blog servers. ACM Trans Web 6(4):16:1–16:26
6. Kavalanekar S, Worthington BL, Zhang Q, Sharda V (2008) Characterization of storage workload traces from production Windows Servers. In: 4th international symposium on workload characterization (IISWC 2008), Seattle, Washington, USA, September 14–16, 2008, pp 119–128
7. Liu Y, Gunasekaran R, Ma X, Vazhkudai SS (2014) Automatic identification of application I/O signatures from noisy server-side traces. In: Proceedings of the 12th USENIX conference on file and storage technologies (FAST 14). USENIX, Santa Clara, pp 213–228
8. Jay L, Fang Z, Qing L, Scott K, Ron O, Todd K, Karsten S, Wolf M (2010) Managing variability in the IO performance of Petascale Storage Systems. In: SC
9. Mesnier MP, Wachs M, Sambasivan RR, Lopez J, Hendricks J, Ganger GR, O'Hallaron D (2007) Trace: parallel trace replay with approximate causal events. In: Proceedings of the 5th USENIX conference on file and storage technologies, FAST '07. USENIX Association, Berkeley, pp 24–24
10. Mi N, Riska A, Zhang Q, Smirni E, Riedel E (2009) Efficient management of idleness in storage systems. ACM Trans Storage 5(2):1–25

11. Miller R, Hill J, Dillow DA, Gunasekaran R, Shipman GM, Maxwell D (2010). Monitoring tools for large scale systems. In CUG '10: proceedings of Cray User's Group (CUG) meeting
12. Noeth M, Ratn P, Mueller F, Schulz M, de Supinski BR (August 2009) ScalaTrace: scalable compression and replay of communication traces for high-performance computing. J Parallel Distrib Comput 69(8):696–710
13. Riska A, Riedel E (October 2009) Evaluation of disk-level workloads at different time scales. SIGMETRICS Perform Eval Rev 37(2):67–68
14. Shipman G, Dillow DA, Oral S, Wang F, Fuller D, Hill J, Zhang Z (2010) Lessons learned in deploying the World's Largest Scale Lustre File System. In: CUG
15. Shipman GM, Dillow DA, Oral S, Wang F (2009) The spider center wide file systems; from concept to reality. In: CUG '09: Proceedings of the 2009 Cray User Group
16. Wu X, Vijayakumar K, Mueller F, Ma X, Roth PC (2011) Probabilistic communication and I/O tracing with deterministic replay at scale. In: International conference on parallel processing, ICPP 2011, Taipei, Taiwan, September 13–16, 2011, pp 196–205
17. Zhang J, Sivasubramaniam A, Franke H, Gautam N, Zhang Y, Nagar S (2004) Synthesizing representative i/o workloads for tpc-h. In: Proceedings of the 10th international symposium on high performance computer architecture, HPCA '04. IEEE Computer Society, Washington, DC, pp 142