Heavy-tailed Distribution of Parallel I/O System Response Time

Bin Dong
Lawrence Berkeley National Laboratory
One Cyclotron Rd
Berkeley, CA 94720
DBin@lbl.gov

Surendra Byna
Lawrence Berkeley National Laboratory
One Cyclotron Rd
Berkeley, CA 94720
SByna@lbl.gov

Kesheng Wu
Lawrence Berkeley National Laboratory
One Cyclotron Rd
Berkeley, CA 94720
KWu@lbl.gov

ABSTRACT

Estimating I/O time of applications is critical for computing system research and developments, such as performance tuning and job scheduling. Parallel I/O systems on large-scale HPC systems typically use several I/O servers attached to a number of hard disk drives to read and write data concurrently. As a result, the response time of individual I/O servers affects the overall I/O performance and modeling the response time distribution holds the key to estimate I/O time. Existing studies have generally considered that the response time follows a Uniform or a Normal distribution. However, none of these studies considered supercomputing environments that are actively used by a number of users to verify the existence of Uniform or Normal distributions. In this study, we collected ≈ *2,500,000* measurements on two peta-scale class supercomputers that are actively used by ≈5000 users. These two systems, $Hopper$ and $Edison$ at the National Energy Research Scientific Computing Center (NERSC), typically support hundreds of concurrent jobs. Our performance measurements include the overheads introduced by the entire parallel I/O stack (I/O library, network, parallel file system software, cache and hardware). Our study shows that the response time of parallel I/O system follows a *heavy-tailed property*, in contrary to the widely accepted Normal or Uniform distributions. In exploring for new models, we identify that a mix of Power Law and Normal distributions is a good fit for the response time of parallel I/O systems that are actively used by hundreds of jobs concurrently.

General Terms

Peta-scale storage system, performance and benchmarking

Keywords

Parallel I/O, heavy-tailed distribution, response time

1. INTRODUCTION

Estimating the time that applications spend on reading and writing data is a common task for their I/O performance tuning [19, 20, 10, 31] and job scheduling [14] on HPC systems, and SQL query plan optimization [2, 30] in database systems. One specific example is the Scientific Data Services Framework (SDS) [8, 7, 9] we are working on. SDS is a persistent service, which organizes the same dataset in different layouts and selects a copy of the data for achieving best performance based on the I/O request pattern. To select the best copy, the service needs to predict performance of accessing data stored in different organizations [18]. The performance estimation in SDS depends on estimating the I/O time. To accurately estimate the I/O time of applications, one key requirement is to determine the probability distribution of the response time of underlying parallel I/O system [11]. In this work, we aim to identify the probability distribution of the response time of a parallel I/O system.

In parallel I/O systems, a file is partitioned into multiple chunks that are then assigned onto independent devices, i.e., hard disks [15]. From the perspective of a user, a large file request is usually partitioned into multiple small requests and these small requests are served by storage devices concurrently. Once all these small file requests are finished, the large file request is considered to be complete. In other words, the performance of the large file request is determined by the longest response time of all involved small requests.

Generally, let R be a big file request, divided into $n(n \geq 1)$ small requests $\{r_1, r_2, \ldots, r_n\}$, and $\{t_{r_1}, t_{r_2}, , \ldots, t_{r_n}\}$ be the response time of these n small requests, the response time T_R of R can be expressed as Eq. 1, where ψ is the overhead of merging these small requests. Widely accepted method to calculate T_R with Eq. 1 is order statistics [13]. The idea of order statistics is to firstly determine the probability distribution of T_R using the probability distribution of each $t_{r_i} (i \in [1, n])$ and then calculate T_R based on its determined probability distribution. As merging all these small requests into big one usually takes fixed time, ψ can be treated as a constant value and therefore it has small impact on computing T_R with the order statistic theory.

$$T_R = \max_n \{t_{r_1}, t_{r_2}, \ldots, t_{r_n}\} + \psi \qquad (1)$$

The details of applying order statistics to calculate T_R and to further tune the performance of I/O systems and applications are out of the scope for this paper. Here we focus on the discussion of *the probability distribution of the response time of each small file request, which is the first step to estimate the response time of* T_R.

In existing research [21, 25, 11, 26], a single Uniform or a single Normal probability distributions was used to model the response time of a parallel I/O system. Using Normal distribution to describe the response time implicitly assumes that the most-frequent values (long or extremely long response time in this case) are rare and occur near the tail ends. Using Uniform distribution to model the response time implicitly assume that all response times have

PDSW2015, November 15-20, 2015, Austin, TX, USA
Copyright 2015 ACM. ISBN 978-1-4503-4008-3/15/11 ...$15.00
DOI: http://dx.doi.org/10.1145/2834976.2834978.

equal chance to happen. From the Eq. 1 above and its associated discussions, we know that the distribution of the response times, especially the long response times, is essential to model the parallel I/O performance. However, none of the efforts have tried to evaluate these distributions on a parallel I/O system in production. A typical parallel I/O system consists of multiple layers, such as hard disk drives, server-side cache, network, client-side cache, parallel I/O middleware, high-level I/O libraries, etc. All these layers need to be considered carefully in the evaluation of these two probability distributions.

To verify the parallel I/O response time distribution, in this study, we develop a method to sample the response time of individual I/O servers employed by two Lustre file systems [15]. These two Lustre file systems are separately used by two peta-scale HPC systems located at NERSC. These systems, Hopper and Edison[1], are currently serving around $5,000$ active users from different disciplines, such as Lattice Gauge Theory, Fusion, Energy, and Material Science. The sampling method developed here (Section 2) emphasizes the parallel file system that essentially delivers the parallelism of underlying hardware. Our method is wrapped as batch jobs, which run at computing nodes and send file requests to sample the response time of Lustre periodically. In the path of file transfer, the network transfer overhead is measured. We also test the response time with and without cache separately. The sampled response times are analyzed to validate the Uniform or the Normal distribution. Our key observations include:

- Response times of the parallel I/O systems on Hopper and Edison show a heavy-tailed behavior rather than properties of a single Uniform or a single Normal distribution (Section 3). Based on our observations, 14.6% of the response times are long or extremely long. These long response times result in a long tail to the right of the density function plotted for the overall response times. In contrast, based on the theory of Normal or Uniform distribution, these long response times shaped as a heavy tail only happen rarely ($\approx$ 2.5% for Normal and $\approx$ 0% for Uniform). Although the generality of our validations are limited by the number of systems for sampling, Edison and Hopper are representative HPC systems with I/O intensive workload nowadays.
- Response times of the parallel I/O systems of Hopper and Edison can be fitted with a mixed distribution of Power Law [6] and Normal. As the performance variability of parallel I/O system, we find that a single distribution function failed to model the entire response time. In that sense, we employ a partition method to divide the response times into head and tail groups and fit them separately. The tail group, representing the long or extremely long response times, can be fitted with Power Law distribution. The head group, representing the short response times, can be fitted with Normal distribution. While the properties of this mixed distribution needs to be further explored on different systems, our study provides a foundation for building sophisticated performance models for parallel I/O systems (Section 4).

2. TEST ENVIRONMENT AND METHODOLOGY

In this section, we report the details of Edison and Hopper systems and our method for sampling the response times.

2.1 Parallel I/O Systems of Hopper and Edison

Table 1: Lustre file systems statistics on September 2014

	/SCRATCH2 of Edison	/SCRATCH2 of Hopper
# of OSSs	18	26
# of OSTs	72	156
# of Users	1382	4597

Hopper and Edison, two supercomputing systems installed at NERSC, are capable of delivering 1.28PF (petaflop) and 2.39PF performance, respectively. Based on our observations of the length of the job queues on these two systems in September 2014, there are on average 255 and 330 application jobs running concurrently on Hopper and Edison separately. In other words, each system is serving hundreds of applications at a time. These applications come from diverse fields, such as Lattice Gauge Theory (26%), Fusion Energy(17%), Material Science(15%), Combustion (13%), etc. [1]. These applications usually read initial data and dump check-point or intermediate/final results from or to the parallel I/O systems. The combination of the vast number of currently running jobs and the diversity of I/O operations makes Hopper and Edison suitable environments for profiling and analyzing parallel I/O system.

Lustre file system [15] is used by both systems to partition files, allocate partitioned files, and coordinate parallel data accesses. The hardware used as the back-end are 7200 RPM hard drives. These hard drives are organized as a RAID. Then, multiple RAIDs are connected to OSTs (object storage target) and OSSs (object storage servers) of Lustre through a high speed network. Usually, each OST in Lustre can be regarded as a standalone device like a single hard disk drive. Hence, the parallelism of the I/O systems of Hopper and Edison is determined by the number of OSTs in its Lustre file system. We summarize these configurations in Table 1.

2.2 Measurement Method

The response time of reading and writing is measured as the total amount of time when these two basic I/O operations are finished by the parallel I/O system. As the cache and network might have effect on the I/O response time, our sampling method is divided into two functions *Sample-NoCache* and *Sample-Cache*. *Sample-NoCache* is used to test the response time without cache effect. It invalidates the cached data through reading and writing a RAM-sized data multiple times. The data used for reading test tends to be retrieved from disk. To test the time of writing data, file synchronization (fsync) is called by *Sample-NoCache* to dump data to disk. *Sample-Cache* function tests the response time of cached data. In such a case, file data is read and written without cache-flush and file synchronization. *Sample-Cache* function also reflects the network effect on the response time to some extent. It is because that RAM access is fast and therefore network transferring consumes most of time in accessing data from remote RAM cache of Lustre Server. Both functions employ standard method of MPI-IO interface to read and write the data. We also tested the POSIX-IO and the difference from MPI-IO is small in results. Hence, we only report the results from MPI-IO in this work.

We set the file request size to be equal to or be a multiple of the striping size of parallel file system (i.e., Lustre). As the big file requests are partitioned into multiple independent small requests at first and then these small requests are sent to different OSTs of Lustre, the file requests arriving at one OST can be regarded as being independent from the file requests arriving at other OSTes. Even though, in real applications, the file request size might be less than the striping size, a consistent ratio can be applied to reflect

[1] `http://www.nersc.gov/systems/hopper-cray-xe6/`, `http://www.nersc.gov/users/computational-systems/edison/`

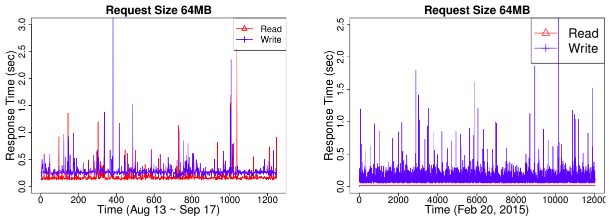
this case. Nevertheless, we tested the request size which is smaller than the striping size of Lustre. We choose these request sizes from 512KB to 1024MB and try to cover basic striping size of most existing systems. For example, 1MB striping size is employed by Lustre and 64MB by HDFS (Hadoop Distributed File System).

To ensure none or ignore-able interference to existing systems and running applications, we run our sampling program as a batch job periodically with normal user privileges. The actual interval for running the sampling programs varies because the batch systems of Hopper and Edison start a job based its resource requirement and wall-clock length. Since *Sample-NoCache* is immune to cache effect, we run it as normal job in a long period. Once a job of *Sample-NoCache* starts to run, it tests the response time of reading and writing certain request size once. As shown in next section, the sampling period lasts almost the whole year. The results from *Sample-NoCache* are expected to reflect the long-term condition of the whole system. To simulate the cache effect, we run *Sample-Cache* with a *30* minutes job. Once the job starts to run, it repeatedly calls *Sample-Cache* to test the response times. Short interval ensures that the data for test is kept in RAM cache by parallel I/O system. We create different files for different request sizes before we start to run our sampling code. Thus, the OST assigned for each test file is fixed.

3. OVERVIEW OF SAMPLED RESPONSE TIME

Table 2: Summary statistics of sampled response time

	# of Jobs	Total Observations
Edison-NoCache	14977	14977
Edison-Cache	12	927691
Hopper-NoCache	13868	13868
Hopper-Cache	12	1581364



(a) Edison-NoCache (b) Edison-Cache

Figure 1: Pictorial demonstration of the I/O responses time trace log

We started to sample the response time of parallel I/O systems on Hopper and Edison in February 2014. Several interruptions were caused by the system maintenance events. Here we report the most recent and the longest continuously results. Table 2 summarizes these results. The response times without cache effect for Edison were sampled from August 13, 2014 to September 17, 2014 and for Hopper from October 1st, 2014 to January 13, 2015. Considering that the size of sample results for cache results is too large, we only reported here the response times with cache effect that were sampled on February 20, 2015 for both Edison and Hopper. In total, we ran 28869 jobs to obtain the samples of the response time.

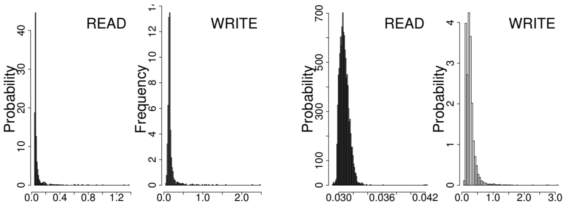
Figure 1 presents a pictorial demonstration of the response time sampled from Edison. We observed the same trend on Hopper. But as the limited of page size, we only report results from Edison here. The X-axis is the time to sample and the Y-axis is the sampled response time in seconds. For simplicity, we only present the figures for the request size of 64MB on Edison. We can see that the response time for both read and write operations has high variability. As tests were conducted on production systems, such variability might come from the interference of the applications running currently on the systems. In other words, if a user wants to read/write data from the parallel I/O systems, such variability is what the user is expected to experience. We can also see that the cache helps to smooth and reduce the response time of reading data, but it has small help to reduce variability of writing data.

4. STATISTICAL ANALYSIS OF RESPONSE TIME

In this section, we analyze the statistic properties of the sampled response time. Specifically, we demonstrate that the ill-fitting of single Uniform or single Normal distribution. We also identify that the mix distribution of log-normal and normal can provide the good fit for the response time.

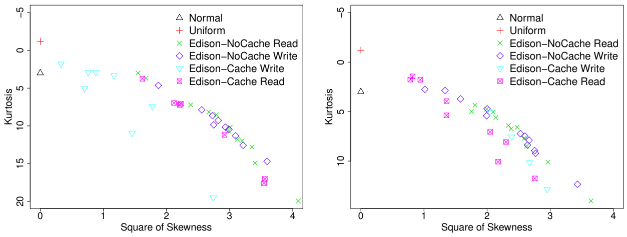
In the following analysis discussions, the *sample* refers to the response times which are tested with a fix requests size, having cache or without cache, and on Edison or Hopper. As we tested 12 requests sizes from 512KB to 1GB. In total, we have $48(2\times2\times12)$ samples. For simplicity, we will not repeatedly present the same results for all samples. We randomly choose the stripe size 64MB on Edison and the request size 16MB on Hopper as the example to show our main results. As cache is important factor for response time, we show the results for cache and non cache separately.

4.1 Ill-fitting of single Uniform and single Normal distribution



(a) Edison-NoCache (b) Edison-Cache

Figure 2: Histograms of the response time of request size 64MB on Edison.



(a) Edison (b) Hopper

Figure 3: Skewness-kurtosis graph for Uniform, Normal, and sampled response time

Fig. 2 presents the histograms (also called density) for the response times tested with the request sizes 64MB on Edison and the request size 16MB on Hopper. The density of other request sizes show similar pattern and thus are not repeated to show here.

It is easy to identify that the histograms of the response times are asymmetrical, have a peak, and decay (long tail) to the right. It is known that the histogram for Normal is symmetrical and has a peak and for Uniform is symmetrical. Hence, Uniform and Normal are not good fit for these response times. To further assure this statement, we compute the Skewness and Kurtosis values [6] for our sampled response times, Normal distribution, and Uniform distributions. Skewness and Kurtosis values are widely used by scientists to select the distributions for data. Fig. 3 presents the computed results. The vertical axis is Kurtosis value and horizontal axis is Skewness value. Except the two plots (triangle and plus) for Normal and Uniform, each of other points represents one test sample using a certain request size with or without cache. We can see that the Skewness and Kurtosis values of almost all sampled results are far from that of Uniform and Normal distributions. Hence, neither single Uniform nor single Normal distribution are good fit for the response time of parallel I/O systems on Hopper and Edison.

4.2 Mix distribution for Response Time

As the single Uniform or single Normal distribution are not good fit for the response time, we now move on to find out that which distribution fits the response time. After trying to fit the response times with most single probability distributions, including Gamma, Cauchy, Weibull, Log Normal, Power Law, and Exponential, we noticed that none of them fit the whole data very well. For example, the Cauchy distribution fits the peak (i.e., small response time) very well while it provides poor fit for the decay parts (i.e., big response time). On the other hand, the functions proposed for heavy tail distributions fit the decay parts very well but they fail to match smaller response times.

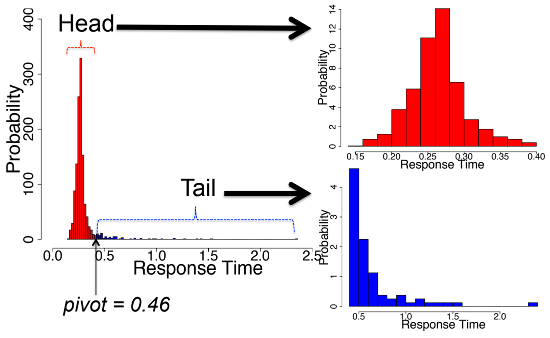


Figure 4: Example shows the idea of dividing the whole response times into two groups, head and tail, and fitting them independently.

Based on these observations, the method we employ to fit the distribution of response time is that dividing each sample data into two groups, head and tail, and fitting these two groups separately. This method is similar to the work in [24], where the authors fit the same distributions for each partition and combine the fitted results together. Our method, however, needs to fit different distributions for different groups. Another reason is that, as stated by most researchers [6], heavy tail distributions only exist in parts (tails) of the data. These researchers always find a minimum value x_{min} and fit the heavy tail distribution with the data bigger than x_{min}.

An example of our partition method is shown in Fig. 4, where the horizontal axis is the response time and the vertical axis is density. The head group with the small response times shows peak and symmetric characters. The tail group containing big response times shows decay and asymmetric properties. Through partition, more accurate distributions is possible to be found for modeling the response time. In the following paragraphs, we will describe our methods used to partition each sample into groups and find the proper probability for each group.

To describe our partition method formally and clearly, we define a value named *pivot* to partition the data, as shown in Fig. 4. The head are the response times which are less than *pivot* and the tail are the response times which are greater than *pivot*. The *pivot* is similar to the x_{min} value used in fitting heavy tailed distributions. To determine the x_{min} for heavy tailed distributions, the Kolmogorov-Smirnov distance[6] between the real data and fitted distribution is computed for each value at first. Then, the one with minimum Kolmogorov-Smirnov distance is selected as x_{min}. Employing the similar idea of estimating x_{min} , we propose a method to determine *pivot*. For each response time $t_i, i \in [2, n-1]$, we compute the Kolmogorov-Smirnov distances for both head group and tail groups, which are partitioned based on t_i.

For the tail group, it owns asymmetric and decay properties, as shown in Fig. 4. The candidate probability distributions for such data include Power Law, Exponential, Log Normal, Weibull, and Gamma. Maximum likelihood method is popular and widely accepted. Hence, we use maximum likelihood method to find the probability distributions of the response time[6]. Most researchers estimate x_{min} for the Power Law and Log Normal distributions and then only fit the data that are bigger than x_{min}. Since we have already partition the data based on *pivot*, we fitted these distributions to the whole tail group without computing x_{min} for it.

For the head group of the response times, it presents symmetric and peak properties, as shown in Fig. 4. The candidate probability distributions for such data include Normal and Cauchy. We also employed the maximum likelihood method to find the parameters of these candidates distributions. Because that x_{min} in not necessary in fitting for Normal and Cauchy, we apply Normal and Cauchy to the whole head group.

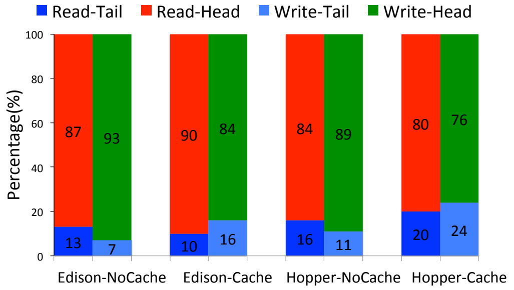
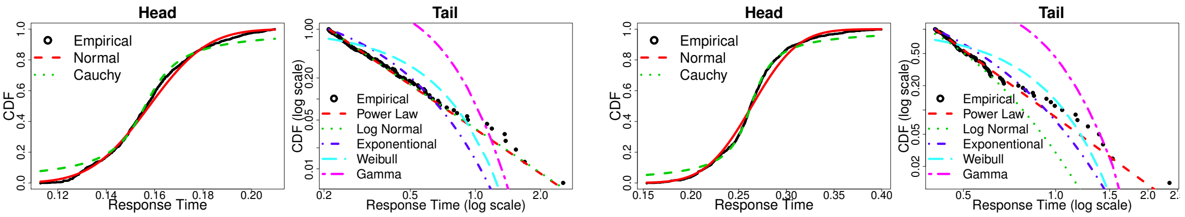


Figure 5: Percentage of Response in Tail and Head group

Results Discussion The partition method divides each response time sample into head and tail groups based on *pivot*. Fig. 5 gives the percentage of the response times partitioned into each group. For example, the bottom left blue box indicates that 13% of the read response times sampled for Edison-NoCache fall in the tail group. We can see that even most response times are in the head groups, the long response times consume a big portion in all cases, even with caches. On average, 14.6% of the response times in the whole sample belong to the tail groups and therefore 85.6% of response times falls in the head groups. As discussed earlier, the head group represents the short response times and the tail group represents long response times. In that sense, we can say that a single file request for users have around 14.6% chance to experience poor performance and 85.6% chance to get good performance. Even users have much higher chance to have good performance than poor performance in a single file request, user might have higher chance to have poor performance in parallel I/O operations. It is because



(a) Read (b) Write

Figure 6: Edison-NoCache, Request Size 64MB

that, in parallel I/O operations, users might read data from multiple sources and they can obtain long response time from multiple OSTs.

When a $pivot$ is determined, each sample can be partitioned into head and tail group. We presented the results for partitioning the response time of writing 64MB striping size on Edison without cache in Fig. 4. As we computed the Kolmogorov-Smirnov distance of Normal and Cauchy for the head group, we can see that the head group (top right) extracted from the original data (left) shows a peak and symmetric distribution. For the tail group, we use the heavy-tailed distributions (e.g., Log Normal, Power Law, etc.) to compute the Kolmogorov-Smirnov distance. We can see that the tail group show decay and symmetric property. We observed the same pattern in other request sizes. In summary, our partition method can divide the whole data into two independently groups, one with peak and symmetric distribution and another one with asymmetric and decay distribution.

To help visualizing the fitted results, we presented the cumulative distribution functions (CDF) for both empirical data and fitted distributions on Edison in Fig. 6. As the space is limited and we observed the same trends on Edison with cache and on Hopper, their results are not reported here. The results indicates that Normal distribution matches the head groups better than Cauchy distribution in all cases. It is reasonable to conclude that small response times follow Normal distribution. The minimum response time can be regarded as the optimal response time that the storage system can provide. As the response time of parallel I/O system follows the Normal distribution, most (94%) of small response times gather around a average values but only 3% of them are optimal ones. As we discussed earlier, the parallel I/O system we sampled on Hopper and Edison are extensively shared by hundreds of applications at one time. The interference of these concurrent applications might have impact on disk access, network transfer, or cache capability. Among the distributions fitted for the tail groups, we found that Power Law matches the response times in most cases. Log Normal provides competitive fits. The Weibull and Gamma are poor fits for the response times in tail group. Typically, Power Law is one example of heavy-tailed distribution, for which the probability density function goes to zero as a power. In parallel I/O system, the Power Law distribution indicates that large population of response time in tail group falls in the tail. In other words, users tend to get extreme long response time with high probability. We test the response time with the striping sizes from 512KB to 1GB. We have observed this heavy tailed distribution in all cases.

5. RELATED WORK

Several efforts [28, 5, 18, 27] estimate performance of parallel I/O systems in order to reduce the I/O latency. Many of these models, however, assume the absence of interference from other jobs. Our work considers the existence of interference in throughout the system. Our analysis can be used as a building block for accurate modeling, which are needed for designing future parallel I/O systems and its associated optimizations.

The Charisma project [22] characterized parallel I/O workloads on Intel iPSC/860 and CM-5 systems. The authors analyzed job query length, quantity and size of the files accessed by applications, I/O request size, and I/O access patterns. The trace logs of various I/O benchmarks were analyzed for both static and dynamic I/O workload property [29]. Carns et al. analyzed the I/O workloads of four applications, MADBench2, Chombo, S3D-IO, and HOMME, on IBM Blue Gene/P [3, 4]. This analysis included statistical properties of file request size, datatype distribution, and collective I/O operations. Various other research efforts also traced and analyzed statistics of I/O operations [17, 23, 12]. Among these, [23] found correlations of inter-arrival rates of I/O operations that exhibit Poisson or Markov distribution.

Recently, Kim et al. [16] characterized storage cluster, Spider, at the Oak Ridge Leadership Computing Facility (OLCF) in an observation of parallel file system logs collected for six months. The focus of the study included system utilization, distribution of read/write sizes, and inter-arrival for I/O requests. In 2013, the Darshan log gathered on Hopper from January 1 to March 13 were analyzed at application-level and system-level [4]. The metrics authors analyzed include redundant I/O traffic, metadata overhead, access patterns (i.e, small independent writes).

6. CONCLUSIONS AND FUTURE WORK

Single Uniform or single Normal distributions is generally used in modeling the response time performance of a parallel I/O system. In this study, we tried to verify them in two real parallel I/O systems separately used by two peta-scale super computers, Hopper and Edison. We sampled these two parallel I/O system around one years and therefore our overall sample size for response time is over $2,500,000$. Our finding is that the response time of these two parallel I/O system exhibits heavy-tailed property rather than the property of Uniform and Normal distribution. Through partitioning the response times of each sample into head group and tail group, we proposed a mix distribution of Power Law and Normal to fit the response time of parallel I/O systems. The evaluated results manifested that it fits the response time very well. We believe that the mix function of Normal and Power Law is pioneering work in building new and accurate performance model for parallel I/O system. Further work includes exploring the mathematical characters of this mix distribution of Power Law and Normal. Then, we want to explore the application of this mix distribution in real applications, such as collective I/O decision and striping size selection.

Acknowledgment

This work is supported by the Director, Office of Laboratory Policy and Infrastructure Management of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231, and used resources of the National Energy Research Scientific Computing Center (NERSC).

7. REFERENCES

[1] K. Antypas, T. Butler, and J. Carter. The hopper system: How the largest xe6 in the world went from requirements to reality. CUG 2011.

[2] S. Babu and H. Herodotou. Massively parallel databases and mapreduce systems. *Found. Trends databases*, 5(1):1–104, Nov. 2013.

[3] P. H. Carns, R. Latham, R. B. Ross, K. Iskra, S. Lang, and K. Riley. 24/7 characterization of petascale i/o workloads. In *Proceedings of the First Workshop on Interfaces and Abstractions for Scientific Data Storage*, New Orleans, LA, USA, 09/2009 2009.

[4] P. H. Carns, Y. Yao, K. Harms, R. Latham, R. B. Ross, and K. Antypas. Production i/o characterization on the cray xe6. In *CUG2013*, Napa Valley, CA, May 9 2013.

[5] P. M. Chen and D. A. Patterson. Maximizing performance in a striped disk array. In *Proceedings of the 17th Annual International Symposium on Computer Architecture*, ISCA '90, pages 322–331, New York, NY, USA, 1990. ACM.

[6] A. Clauset, C. R. Shalizi, and M. E. J. Newman. Power-law distributions in empirical data. *SIAM Rev.*, 51(4):661–703, Nov. 2009.

[7] B. Dong, S. Byna, and K. Wu. Expediting scientific data analysis with reorganization of data. In *Cluster Computing (CLUSTER), 2013 IEEE International Conference on*, pages 1–8, Sept. 2013.

[8] B. Dong, S. Byna, and K. Wu. SDS: A Framework for Scientific Data Services. In *Proceedings of the 8th Parallel Data Storage Workshop*, PDSW '13, pages 27–32, New York, NY, USA, 2013. ACM.

[9] B. Dong, S. Byna, and K. Wu. Parallel query evaluation as a Scientific Data Service. In *2014 IEEE International Conference on Cluster Computing, CLUSTER 2014, Madrid, Spain, September 22-26, 2014*, pages 194–202, 2014.

[10] B. Dong, X. Li, L. Xiao, and L. Ruan. Exploring storage optimizations to accelerate parallel out-of-core matrix product. In *Proceedings of the 2011 Fourth International Joint Conference on Computational Sciences and Optimization*, CSO '11, pages 1–2, 2011.

[11] B. Dong, X. Li, L. Xiao, and L. Ruan. A new file-specific stripe size selection method for highly concurrent data access. In *Proceedings of the 2012 ACM/IEEE 13th International Conference on Grid Computing*, GRID '12, pages 22–30, 2012.

[12] D. Feng, Q. Zou, H. Jiang, and Y. Zhu. A novel model for synthesizing parallel i/o workloads in scientific applications. In *Proceedings of IEEE International Conference on Cluster Computing*, pages 252–261. IEEE, 2008.

[13] M. Hlynka, P. Brill, and W. Horn. A method for obtaining laplace transforms of order statistics of erlang random variables. *Statistics and Probability Letters*, 80(1):9 – 18, 2010.

[14] M. Hovestadt, O. Kao, A. Keller, and A. Streit. Scheduling in hpc resource management systems: Queuing vs. planning. In D. Feitelson, L. Rudolph, and U. Schwiegelshohn, editors, *Job Scheduling Strategies for Parallel Processing*, volume 2862 of *Lecture Notes in Computer Science*, pages 1–20. Springer Berlin Heidelberg, 2003.

[15] P. J.Braam. The lustre storage architecture (tech. rep.). Technical report, Available: http://wiki.lustre.org/., 2004.

[16] Y. Kim, R. Gunasekaran, G. Shipman, D. Dillow, Z. Zhang, and B. Settlemyer. Workload characterization of a leadership class storage cluster. In *Petascale Data Storage Workshop (PDSW), 2010 5th*, pages 1–5, Nov 2010.

[17] L. A. N. Laboratory. *MPI IO TEST traces*, 2009.

[18] J. Liu, S. Byna, B. Dong, K. Wu, and Y. Chen. Model-driven data layout selection for improving read performance. 2014.

[19] J. Liu, Y. Chen, and Y. Zhuang. Hierarchical i/o scheduling for collective i/o. In *Cluster, Cloud and Grid Computing (CCGrid), 2013 13th IEEE/ACM International Symposium on*, pages 211–218, May 2013.

[20] J. Liu, Y. Zhuang, and Y. Chen. Hierarchical collective i/o scheduling for high-performance computing. *Big Data Research*, 2(3):117 – 126, 2015. Big Data, Analytics, and High-Performance Computing.

[21] M. R. MEDINA. *A self-tuning disk striping system for parallel input/output*. dissertation, University of Illinois at Urbana-Champaign, 2007.

[22] N. Nieuwejaar, D. Kotz, A. Purakayastha, C. S. Ellis, and M. L. Best. File-access characteristics of parallel scientific workloads. *IEEE Trans. Parallel Distrib. Syst.*, 7(10):1075–1089, Oct. 1996.

[23] S. Park and K. Shen. A performance evaluation of scientific i/o workloads on flash-based ssds. In *Proceedings of IEEE International Conference on Cluster Computing*, pages 1–5. IEEE, 2009.

[24] A. Riska, V. Diev, and E. Smirni. Efficient fitting of long-tailed data sets into hyperexponential distributions. In *Global Telecommunications Conference, 2002. GLOBECOM '02. IEEE*, volume 3, pages 2513–2517 vol.3, Nov 2002.

[25] P. Scheuermann, G. Weikum, and P. Zabback. Data partitioning and load balancing in parallel disk systems. *The VLDB Journal*, 7(1):48–66, Feb. 1998.

[26] H. Simitci. Adaptive disk striping for parallel input/output. Technical report, Champaign, IL, USA, 2000.

[27] H. Tang, X. Zou, J. Jenkins, D. A. B. II, S. Ranshous, D. Kimpe, S. Klasky, and N. F. Samatova. Improving read performance with online access pattern analysis and prefetching. In *Euro-Par 2014 Parallel Processing - 20th International Conference, Porto, Portugal, August 25-29, 2014. Proceedings*, pages 246–257, 2014.

[28] E. Varki, A. Merchant, J. Xu, and X. Qiu. Issues and challenges in the performance analysis of real disk arrays. *IEEE Trans. Parallel Distrib. Syst.*, 15(6):559–574, June 2004.

[29] F. Wang, Q. Xin, B. Hong, S. A. Brandt, E. L. Miller, D. D. E. Long, and T. T. Mclarty. File System Workload Analysis for Large Scale Scientific Computing Applications. In *In Proceedings of the 21st IEEE / 12th NASA Goddard Conference on Mass Storage Systems and Technologies*, pages 139–152, 2004.

[30] X. Zou, S. Lakshminarasimhan, D. A. B. II, S. Ranshous, H. Tang, S. Klasky, and N. F. Samatova. Fast set intersection through run-time bitmap construction over pfordelta-compressed indexes. In *Euro-Par 2014 Parallel Processing - 20th International Conference, Porto, Portugal, August 25-29, 2014. Proceedings*, pages 668–679, 2014.

[31] X. Zou, K. Wu, D. A. B. II, D. F. Martin, S. Byna, H. Tang, K. Bansal, T. J. Ligocki, H. Johansen, and N. F. Samatova. Parallel in situ detection of connected components in adaptive mesh refinement data. In *15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, CCGrid 2015, Shenzhen, China, May 4-7, 2015*, pages 302–312, 2015.