International Journal of Grid and Distributed Computing
Vol. 9, No. 5 (2016), pp.99-110
http://dx.doi.org/10.14257/ijgdc.2016.9.5.10

Skewed Data Distribution for Active Storage Systems on Hybrid Servers

Xiangyu Li[1,3], Shuibing He[1,2,*] and Xianbin Xu[1]

[1]Computer School, Wuhan University, Wuhan, Hubei 430072, China
[2] State Key Laboratory of High Performance Computing,
National University of Defense Technology, Changsha, Hunan 410073, China
[3] Wuhan DonghubUniversity, Wuhan, Hubei 430212, China
**Corresponding author: heshuibing@whu.edu.cn*
xylee@whu.edu.cn, xbxu@whu.edu.cn

Abstract

With the popularity of new storage technologies, hybrid active storage system provides an efficient way to improve the performance of high-performance computing applications. However, current active storage efforts have neglected the storage performance gap between heterogeneous servers, largely affecting the overall system performance. In this paper, we propose SDD, a Skewed Data Distribution scheme for hybrid active storage systems. In contrast to traditional even data distribution schemes, SDD distribute data on servers with skewed amount of data based on their performance. We have implemented a prototype of our proposed data layout scheme in a parallel I/O system, and demonstrated its benefits with a typical data processing application. Experimental results show our proposed data placement scheme can significantly improve the overall active storage system performance.

***Keywords**: Active Storage, Parallel I/O System, Data Distribution, Hybrid Servers*

1. Introduction

Many scientific and engineering applications in the highperformance computing (HPC) domains produce vast amounts of experimental and simulation data [1,2]. For example, the applications in astrophysics, geographic systems, climate modeling, and medical image processing, usually generate tens of terabytes simulation data [3, 4]. These data are projected to be in excess of 1 Exabyte per year by 2018 [5]. The growing volumes of data put unprecedented pressure on modern computer systems.

Transferring such tremendous amount of data between computing and storage nodes takes a considerable amount of time. Even on today's highest-performing computer systems, the data movement is a critical performance bottleneck. Although the processor performance improves with a high speed per year owing to the advancements of VLSI technology, the network bandwidth and disk access latency still increase with a much slower rate. Data transfer and storage have become a serious bottleneck for today's data-intensive HPC applications.[1]

Active storage technology provides an efficient way to address the I/O bottleneck problem [6-10]. By processing data on servers instead of on computing nodes, active storage can largely reduce the data movement on I/O path and benefit from the aggregated processing power on multiple storage servers. Due to these advantages, active storage has attracted much attention as a new storage technology to accommodate growing volumes of data [1, 11, 12].

[1] Shuibing He is the corresponding author.

In HPC domains, most of the current active storage systems rely on the file system data distribution method to place data on multiple servers. Current parallel I/O systems usually use an even data distribution scheme to distribute data on servers. While being simple and effective for certain kinds of I/O workloads, this method is designed for homogeneous severs. In this case, each server has the same storage performance as the others, so that it could carry out the active storage operations at the same rate as other servers. For a big active storage task, which is converted to multiple sub-tasks on servers, this even data distribution scheme can make each server finish its sub-task at the same time, which can remove the unnecessary waiting time among different servers.

With the emergence of new storage media, active storage systems may run on heterogeneous servers. For example, with the rapid development of NAND flash technology, solid state disks (SSD) are widely used in a parallel I/O system [13, 14]. Compared with HDDs, SSDs are famous for the much higher data transfer rate and lower access latency. However, building a large I/O system solely on SSDs is too costly and also loses the benefits of hard disks (HDD). For instance, HDDs can provide high capacity and decent peak bandwidth for large sequential requests. Therefore, a parallel I/O system with hybrid servers, namely HDD servers (HServer) and SSD servers (SServer), is more cost-effective for practical storage systems [15-17].

Unfortunately, for such hybrid active storage systems, traditional even data distribution schemes may face significant performance challenges. We assume that each active storage operation works with the typical pattern as follows. It first reads data from the local disk of the server, and then and processes the data in the memory. After that, the result will be written to disk. With the even data distribution, each server will have the same amount of data to process. As a result, the high-performance servers will finish the I/O operation quickly while the low-performance nodes will finish slowly. Since each server will finish their active storage operations with different rates and the overall data processing time depends on the straggler of all storage nodes, traditional data placement schemes will degrade the overall system performance.

In contrast to traditional even data distribution schemes, SDD distributes data on servers with varied stripe sizes based on their performance. We have implemented a prototype of our proposed data layout scheme in a parallel I/O system, and demonstrated its benefits with two typical data processing applications. Experimental results show that our proposed data placement scheme can significantly improve the overall active storage system performance.

In this paper, we propose a SDD, a skewed data distribution scheme to allocate data on servers for hybrid active storage systems. As SServers have higher data processing capacity during active storage operations because they provide better storage performance than HServers, SDD places data on SServers with a large amount of data than on HServers. Compared to the traditional even data distribution, such skewed distribution mitigates the load imbalance among hybrid servers. To determine the proper data amount on each server, SDD relies on a cost model and a linear programming optimizing method. Specially, we make the following key contributions:

- We introduce a cost model to evaluate the completion time of an active storage operation in a hybrid active storage system.
- We propose a linear programming method to determine the proper data amount on each server based on the cost model.
- We propose a skewed data distribution scheme with the optimal data amount for hybrid active storage systems.
- We implemented a prototype of SDD under a parallel I/O system, and evaluated its performance with a typical application. Experimental results show that SDD can significantly improve the active storage system performance.

The rest of this paper is organized as follows. In Section 2, we describe the related work. Then we describe the design and implementation of SDD in Section 3. Section 4 gives the performance evaluation. Finally, we conclude the paper in Section 5.

2. Related Work

In this section, we briefly introduce the previous active storage researches and the data distribution methods related to our work.

2.1. Active Storage on Disk Devices

To benefit from the under-utilized hardware source of computer system, active storage technology is first proposed to exploit the computing intelligence inside disk drives. These techniques are originally designed for database applications, such as CASSM [18] and RAP [19]. Gradually, active storage technologies are proposed in other more general fields, such as data mining, multimedia and image processing [7, 8, 20]. To promote its development, new hardware architectures[20-22] and programming models [7] are studied. However, these efforts are only dedicated to utilize the power of embedded processor, thus the systems provide limited computation offloading capability.

2.2. Active Storage on File Systems

With the performance improvements on storage nodes, active storage is enabled in file systems. These studies are conducted in the distributed file system called ScFS [23], the parallel file system PVFS [2, 4, 24], the Lustre parallel file system [25], *etc*. Furthermore, due to the popularity of objectbased storage, many efforts are devoted to integrate active storage with the object-based storage systems [10, 26-29].

While all of the prior arts are effective, they are designed for homogeneous storage clusters, thus cannot be directly applied to heterogeneous servers. As opposite to these efforts, this work is designed for active storage systems with hybrid servers.

2.3. Data Distribution in Parallel I/O Systems

There are three representative data distribution schemes in parallel file systems [30]. One-dimensional horizontal distribution is the simple striping method that distributes a process's file across all available servers in a round-robin fashion; onedimensional vertical distribution performs no striping at all, and instead places the file data on one server; two-dimensional distribution distributes the file on a subset of servers. Each of them works well for a particular kind of I/O access patterns. For complex patterns, segment-level placement scheme logically divides a file into several segments such that an optimal stripe size is assigned for each segment with non-uniform access patterns [31]. Server-level adaptive placement strategies adopt different stripe sizes on different file servers to improve the overall I/O performance [32]. These efforts are devoted to homogeneous systems. For heterogeneous file systems, recent studies [16, 17, 33] uses variable stripes to place data on different servers.

The above studies focus on storage performance of each server, without considering the CPU impact on the overall active storage system performance. In contrast, SDD uses a holistic cost model considering both computation and storage factors to direct the data distribution of the hybrid active storage system.

3. The Skewed Data Distribution Scheme

In this section, we first illustrate the basic idea of the skewed data distribution scheme. We then introduce a data processing cost model in active storage system and describe the determination of the proper data amount on each server. Finally, we present the implementation of the proposed data distribution scheme.

3.1. The Basic Idea of SDD

The goal of the proposed data distribution scheme, SDD, is to optimize the data distribution on hybrid servers based on each server's processing performance. Instead of allocating the same amount of data on each server, as traditional even data distribution schemes do [30], SDD distributes a larger amount of data on SServers and a smaller amount of data on HServers, so that all servers can finish their active storage operations within the same time. As the active storage system performance depends on the straggler of the servers, this scheme can mitigate the load imbalance issue among servers and can significantly improve the overall system performance.

However, determining the proper amounts of data on heterogeneous servers is not easy due to three reasons. First, the server performance can be impacted significantly by I/O patterns, such as request size, I/O operation (read or write), number of processes, *etc*. Second, server performance is also related with the storage media. Even under the same I/O patterns, HServer and SServer have different performance behaviors. Third, in addition to the storage performance, processor on each server can affect the data processing time of the active storage operations.

Thus, we introduce a holistic cost model to evaluate the data processing time for active storage operations in a hybrid active storage system, as we will discuss in the following section.

3.2. Active Storage Data Processing Cost Model

Table 1. Parameters in Cost Analysis Model

Symbol	Description
m	Number of HServers
n	Number of SServers
p	Number of processes on each server
b_h	Number of blocks on HServer
b_s	Number of blocks on SServer
B	Number of blocks in the parallel file
α_h	HServer storage startup time
β_h	HServer unit data transfer time
α_s	SServer storage startup time
β_s	SServer unit data transfer time
p_h	HServer unit data calculation time
p_s	SServer unit data calculation time

The cost is defined as the overall data processing time for each active storage operation. Table 1 lists the related parameters. This model is designed for hybrid active storage environments. Note that the storage parameters for hybrid servers are different to evaluate the I/O processing time. Namely, the startup time and transfer time are different between HServer and SServer. Especially, α_S is much smaller than α_H, and β_S is much larger than β_H. This is because SSDs have no mechanical components, thus it has better bandwidth and lower data latency. Moreover, it is worth pointing out thatthe model will consider the data calculation time and data storage time during the data processing, which is more comprehensive.

Before introducing the details of the model, we make the following reasonable assumptions.

1) We assume a parallel file is fully distributed on all the $m+n$servers, so that each server can contribute to the aggregated data processing capacity. Due to symmetry, we assume perfect load balance of data processing within SServers and HServers (but not across the two types of servers). Assume there are B data blocks in the parallel file, then

$$m \times b_h + n \times b_s = B \quad (1)$$

2) We assume a large active storage task is divided into sub-tasks on all servers, and each server carries out the sub-task concurrently. Therefore, the completion time of the active storage task depends on the slowest sub-task among all servers.
3) For each server, we assume it only conducts the data processing with the data located on itself. Therefore, no server will process data remotely from other servers, hence we do not need to consider the network transfer time due to data migration.
4) We assume each server executes the active storage operation in the following pattern. It first reads data from the storage device, and then performs the calculation in memory. While many similar data processing patterns exist, we focus on this pattern in this paper because it is widely used in practical active storage systems. Furthermore, we assume each server calculating the data with a fixed speed because many applications have this feature during their execution [34]. Therefore, p_h and $p_s$can be a constant for the active storage operations.

The data processing cost is defined as the overall time to complete the active storage operation, which mainly includes two parts: the data I/O time, $T_{I/O}$, and the data calculation time T_C. Generally, $T_{I/O}$ consists of T_S and T_T. The former is the storage startup time, and the latter is the actual data read/write time on storage media. T_C is the time spent on the data calculation in memory. In summary, the cost of one active storage operation can be described as follows.

$$T = T_S + T_T + T_C \quad (2)$$

The startup time TS is determined by the number of I/O operations on one server and the storage device media. We assume p processes exist on one server, each process will read the data from the disk to memory before data calculation, and then p startup operations are needed. Therefore, the startup time can be calculated as

$$T_S = \begin{cases} p \times \alpha_h, & \text{If the server is an HServer} \\ p \times \alpha_s, & \text{If the server is an SServer} \end{cases} \quad (3)$$

The data transfer time T_T of each server depends on the amount of data and the disk data transfer rate. For HServer and SServer, the data transfer rate is different. According to the parameters in Table I, T_T can be described as

$$T_T = \begin{cases} b_h \times \beta_h, & \text{If the server is an HServer} \\ b_s \times \beta_s, & \text{If the server is an SServer} \end{cases} \quad (4)$$

T_C is proportional to the amount of data needed to calculate. According to our assumptions, each server will carry out the active storage operation with all the data located on itself, then the amount of data are b_h and bs for HServer and SServer to process respectively. Assume the server unit data calculation time is p_h and p_s respectively, thus T_C can be defined as

$$T_C = \begin{cases} b_h \times p_h, & \text{If the server is an HServer} \\ b_s \times p_s, & \text{If the server is an SServer} \end{cases} \quad (5)$$

Based on Equation 3 to Equation 5, the overall cost of one active storage operation on each server can be obtained. Since the overall active storage performance is determined by the maximal completion time of each server, we can derive the final cost of the active storage operation. This cost model provides a detailed analysis of completion time for data processing in a hybrid active storage system. Although several parameters exist in the model, for most applications, the runtime variables such as B, p, p_h, and p_s are fixed for each run. In general, for a given system, m, n, α and β can be regarded as constants.

3.3. Determination of Data Amount on Each Server

The above cost model, show that T can be significantly impacted by the number of blocks on HServer and SServer, namely stripe sizes $b_h$and b_s. In other words, distributing different numbers of data blocks on servers leads to substantially various access costs. An optimal data distribution scheme should choose the proper $b_h$and $b_s$to get the best I/O performance. Thus, the optimization problem can be described as minimizing function F described in Equation 6 while satisfying the size constraints described in Equation 1.

$$F = \max\{p\,\alpha_h + b_h\beta_h,\ p + bs\} + \max\{b_h p_h, b_s p_s\} \quad (6)$$

According to the member values in the maximum expressions in Equation 6, this problem can be translated into four linear programming (LP) optimizing problems with two unknowns variables representing the number of data blocks on each HServer and SServer, namely $b_h$and b_s. Finally, the problem is to choose the values of $b_h$and $b_s$so as to minimize F as below. Case 1:

Case 1:

$$\text{Minimize} \quad F = p\alpha_h + b_h\beta_h + b_h p_h \quad (7)$$

$$\text{subject to} \begin{cases} mb_h + nb_s = B \\ p\alpha_s + b_s\beta_s \le p\alpha_h + b_h\beta_h \\ b_s p_s \le b_h p_h \end{cases} \quad (8)$$

Case 2:

$$\text{Minimize} \quad F = p\alpha_s + b_s\beta_s + b_h p_h \quad (9)$$

$$\text{subject to} \begin{cases} mb_h + nb_s = B \\ p\alpha_h + b_h\beta_h < p\alpha_s + b_s\beta_s \\ b_s p_s \le b_h p_h \end{cases} \quad (10)$$

Case 3:

$$\text{Minimize} \quad F = p\alpha_h + b_h\beta_h + b_s p_s \quad (11)$$

$$\text{subject to} \begin{cases} mb_h + nb_s = B \\ p\alpha_s + b_s\beta_s \le p\alpha_h + b_h\beta_h \\ b_h p_h < b_s p_s \end{cases} \quad (12)$$

Case 4:

$$\text{Minimize} \quad F = p\alpha_s + b_s\beta_s + b_s p_s \quad (13)$$

$$\text{subject to} \begin{cases} mb_h + nb_s = B \\ p\alpha_h + b_h\beta_h < p\alpha_s + b_s\beta_s \\ b_h p_h < b_s p_s \end{cases} \quad (14)$$

The finally optimized $b_h$and $b_s$are determined by the case where the objective function F achieves the minimal value among the four cases. As the linear programming optimization is expressed with two unknown variables, the search space is very small and solving the program requires acceptable time cost.

3.4. Skewed Data Distribution Scheme

Based on the optimalb_h and b_s, SDD is able to achieve the skewed data distribution for hybrid active storage systems. However, this method requires that we have a prior knowledge of applications access pattern. Fortunately, many HPC applications access their files with either regular data access patterns or predictable behaviors [35], thus it enables the proposed data distribution scheme based on an I/O profiling procedure.

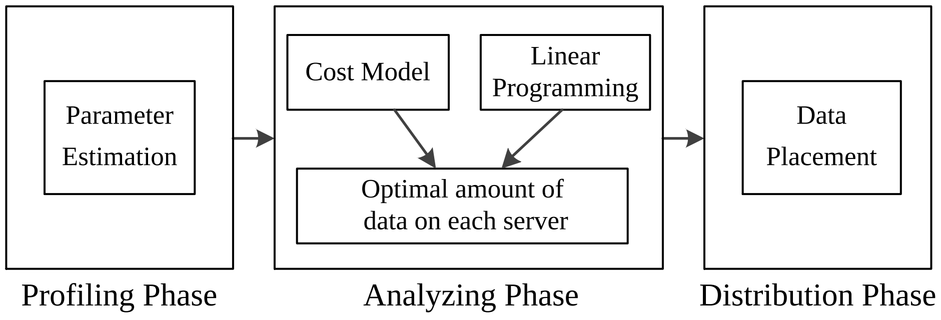

Figure 1. The Procedure of Skewed Data Distribution Scheme

Figure 1 depicts the detailed procedure of the optimal data distribution scheme, which includes three phases. In the *Profiling* phase, the related parameters in Table I are estimated. The storage parameters, such as α_h, β_h, α_s, β_s, the system parameters, such as m and n , and the application parameters, such as p, $p_h$and $p_s$can be regarded as constants. In the *Analyzing* phase, the cost model and the linear programming method are used to calculate the optimal numbers of data blocks on HServers and SServers. As the optimization is a two-variable linear programming problem, it only requires a relatively small time cost and can run very fast for most computer system. In the *Distribution* phase, the file is distributed on the hybrid servers with the optimal $b_h$and b_s. This can be performed by creating new files for later runs of the applications, or adjusting the file layout by copying operations in the existing parallel file systems.

3.5. Implementation

We implement the proposed data distribution scheme for a hybrid active storage system, which is based on MPICH2 [36] and OrangeFS [37] (A successor of PVFS). In the profiling phase, we use a trace collector to obtain the run-time statistics of data accesses during the application's execution. Based on the I/O trace, we obtain the related parameters to evaluate the data processing cost in a hybrid active storage system.

For the *placement* phase, we distribute the file data on hybrid servers with the optimal numbers of data blocks. For ease of implementation, we assign different number of blocks on servers by leveraging the existing varied-size stripping method supported by OrangeFS. For example, if we define the size of a data block as 64kB, which is the default stripe size of OrangeFS, a larger stripe size of 256KB will distribute four contiguous blocks on one server. In OrangeFS, a parallel file can either be accessed by the PVFS2 or the POSIX interface. For PVFS2 interface, we utilize the "pvfs2-xattr" command to set the data distribution method of file directories where the application files are located. For POSIX interface, we use the "setfattr" command to reach the same goal.

4. Performance Evaluation

4.1. Experimental Setup

We conduct the experiments on a Linux cluster, which consists of eight heterogeneous nodes. Two nodes are used as the computing nodes, four nodes are used as HServers and two nodes are used as SServers. The parameters of the nodes are summarized in Table 2. All nodes are equipped with Gigabit Ethernet interconnection. The operating system is Linux kernel 2.6.28.10, the MPI-IO library is MPICH21.4.1p1, and the parallel file system is OrangeFS 2.8.6. In the experiments, the hybrid OrangeFS file system is built on four HServers and two SServers unless otherwise specified.

Table 2. The Three Kinds of Nodes in a Linux Cluster

Node	CPU	Memory	Disk
Computing node	Two AMD Opteron	8GB	500GB HDD
HServer	Intel i5	4GB	250GB HDD
SServer	Intel i5	4GB	100GB SSD

To verify the efficiency of SDD, we compare it with the traditional data distribution (TDD). We test the active storage system performance under SDD and TDD. In TDD, the data is distributed on all servers with a default block size (64KB) in a round-robin way, which is the default data distribution approach for most current active storage systems. Obviously, this scheme does not consider the storage performance difference among hybrid servers and leads to an even data distribution. In contrast, SDD distributes data on hybrid servers with skewed data amount on each server.

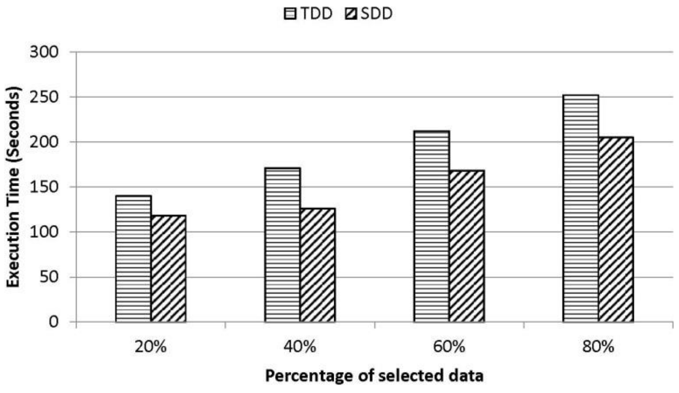

Figure 2. Performance Comparison of Data Selection Application under Different

We use a typical application, data selection, to evaluate the system performance. This application is widely used to search the desired data under a given condition from a large data set.

4.2. Data Selection

In this test, the client first downloads the data selection code onto the servers, and then executes the data selection operation on each server. The execution time of the application consists of all the above parts. In our tests, the data set is a data sequence consisting of millions of data, each of which is 0-9, and the large-scale sequence is stored as a 6GB parallel file on multiple servers.

Figure 2 describes the application execution time under different data selection conditions running TDD and SDD respectively. The percentage means how much data should be selected from the original data set. The results show that SDD is always better than the traditional data distribution scheme TDD. This is because SDD adopts storage-aware data distribution scheme to allocate data on multiple servers, so that high-performance SServers are assigned to process more data and low-performance HServers are assigned less. This skewed data allocation can significantly eliminate the load-imbalance issues among servers in current active storage systems. These results show that our proposed data distribution scheme is an efficient way to improve the hybrid active storage system performance for data-intensive applications.

To show the efficiency of our proposed scheme, we also evaluate SDD under various server configurations. In these tests, we change the HServer and SServer ratio to 5:1 and2:4.

Figure 3 shows the active storage completion time with different server configurations. As can be seen from the results, SDD can improve the performance compared to TDD. When the ratio is 5:1, the system performance can be improved by 17.2% to 26.4%. When the ratio is 2:4, the improvement is 21.4% to 32.3%. We can find that SDD obtains better performance benefits as the number of SServers increases. The main reason is that traditional data distribution schemes waste more hardware potential as more SServers are added into the system, but SDD can make full utilization of them. By using the skewed data distribution determined by the costmodel and linear programming method, SDD can significantly improve the hybrid active storage system performance with various server configurations.

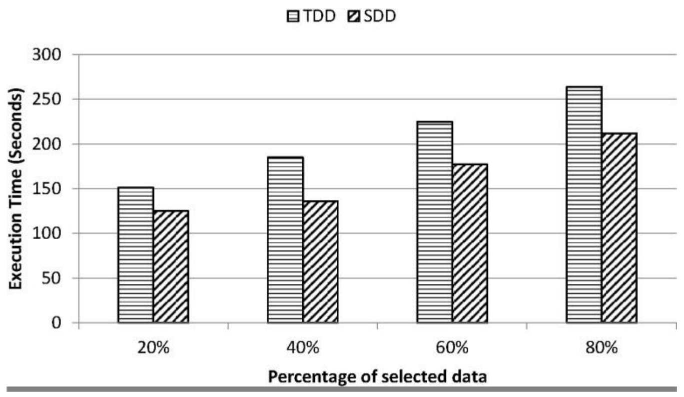

Figure 3. Performance Comparison of Data Selection Application under Different Schemes

5. Conclusion

Active storage technology provides an efficient way to improve the performance of data-intensive high-performance computing applications. With the popularity of new storage technologies, such as solid-state drives (SSD), hybrid active storage systems become possible and feasible in storage system designs. In this paper, we propose SDD, a skewed data distribution scheme to improve the performance of hybrid active storage systems. SDD distributes data on different servers with skewed amount of data based on their performance. To determine the proper data amount on each server, SDD relies on a cost model and a linear programming optimizing method. Compared to even data distribution schemes, SDD mitigates the load imbalance among hybrid servers. We have implemented a prototype of SDD in a parallel I/O system. Experimental results show that our proposed data distribution scheme can obtain considerable performance improvements for hybrid active storage systems.

Acknowledgements

This study is supported in part by the National Science Foundation of China under Grant No.61572377, the NaturalScience Foundation of Hubei Province of China under Grant No.2014CFB239, and the Open-fund from HPCL under Grant No.201512-02.

References

[1] D. Tiwari, S. Boboila, S. S. Vazhkudai, Y. Kim, X. Ma, P. J. Desnoyers, and Y. Solihin, "Active flash: Towards energy-efficient, in-situ data analytics on extreme-scale machines," in *Proceedings of the 11th USENIX Conference on File and Storage Technologies* (FAST'13), **(2013)**.

[2] C. Chen and Y. Chen, "Dynamic active storage for high performance i/o," in *Proceedings of the 41st International Conference on Parallel Processing (ICPP)*, **(2012)**, pp. 379–388.

[3] R. Latham, R. Ross, B. Welch, and K. Antypas, "Parallel i/o in practice," Tech. Rep. Tutorial of the International Conference for High Performance Computing, Networking, Storage and Analysis, **(2013)**.

[4] S. W. Son, S. Lang, P. Carns, R. Ross, R. Thakur, B. Ozisikyilmaz, P. Kumar, W.-K. Liao, and A. Choudhary, "Enabling active storage on parallel i/o software stacks," in *Proceedings of the IEEE 26th Symposium on Mass Storage Systems and Technologies* (MSST). IEEE, **(2010)**, pp.1–12.

[5] Y. Kim, S. Atchley, G. R. Valle, and G. M. Shipman, "Lads: Optimizing data transfers using layout-aware data scheduling," in *Proceedings of the 13th USENIX Conference on File and Storage Technologies* (FAST'15), **(2015)**.

[6] M. Xiaonan and A. L. N. Reddy, "Mvss: An active storage architecture," *IEEE Transactions on Parallel and Distributed Systems*, vol. 14, no. 10, **(2003)**,pp. 993–1005.

[7] A. Acharya, M. Uysal, and J. Saltz, "Active disks: Programming model, algorithms and evaluation," *ACM SIGPLAN Notices*, vol. 33, no. 11, **(1998)**,pp.81–91.

[8] E. Riedel, G. A. Gibson, and C. Faloutsos, "Active storage for large-scale data mining and multimedia," in *Proceedings of the 24rd International Conference on Very Large Data Bases*, **(1998)**, pp. 62–73.

[9] H. Tang, A. Gulbeden, J. Zhou, W. Strathearn, T. Yang, and L. Chu, "The panasasactivescale storage cluster-delivering scalable high bandwidth storage," in *Proceedings of the ACM/IEEE SC2004 Conference on Supercomputing*, **(2004)**, pp. 53–62.

[10] S. He, X. Xu, and Y. Yang, "Oasa: An active storage architecture for object-based storage system," *International Journal of Computational Intelligence Systems*, vol. 5, no. 6, **(2012)**,pp. 1173–1183.

[11] Y. Xie, K. Muniswamy-Reddy, D. Feng, D. Long, Y. Kang, Z. Niu, and Z. Tan, "Design and evaluation of oasis: An active storage framework based on t10 osd standard," in *MSST*. IEEE, **(2011)**, pp. 1–12.

[12] B. Rich and D. Thain, "Datalab: Transactional data-parallel computing on an active storage cloud," in *Proceedings of the 17th International Symposium on High Performance Distributed Computing*, **(2008)**, pp.233–234.

[13] A. Caulfield, L. Grupp, and S. Swanson, "Gordon: Using flash memory to build fast, power-efficient clusters for data-intensive applications," in *Proceedings of the Fourteenth International Conference on Architectural Support for Programming Languages and Operating Systems*, **(2009)**.

[14] J. Ou, J. Shu, Y. Lu, L. Yi, and W. Wang, "Edm: An enduranceaware data migration scheme for load balancing in ssd storage clusters," in *Proceedings of 28th IEEE International Parallel and Distributed Processing Symposium*, **(2014)**.

[15] M. Zhu, G. Li, L. Ruan, K. Xie, and L. Xiao, "Hysf: A striped file assignment strategy for parallel file system with hybrid storage," in *Proceedings of the IEEE International Conference on Embedded and Ubiquitous Computing*, **(2013)**, pp. 511–517.

[16] S. He, X.-H. Sun, and A. Haider, "HAS: Heterogeneity-Aware Selective Data Layout Scheme for Parallel File Systems on Hybrid Servers,"in*Proceedings of 29th IEEE International Parallel and Distributed Processing Symposium*, **(2015)**, pp. 613–622.

[17] S. He, X.-H. Sun, Y. Wang, A. Kougkas, and A. Haider, "AHeterogeneity-Aware Region-Level Data Layout Scheme for Hybrid Parallel File Systems," in *Proceedings of the 44th International Conference on Parallel Processing*, **(2015)**.

[18] E. A. Ozkarahan, S. A. Schuster, and K. C. Smith, "Rap: An associative processor for data base management," in *Proceedings of the AFIPS Joint Computer Conferences*. ACM New York, NY, USA, **(1975)**, pp. 379–387.

[19] S. Y. W. Su and G. J. Lipovski, "Cassm: A cellular system for very large data bases," in *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, **(1975)**, pp. 456–472.

[20] K. Keeton, D. A. Patterson, and J. M. Hellerstein, "A case for intelligent disks (idisks)," *ACM SIGMOD Record*, vol. 27, no. 3, **(1998)**, pp. 42–52.

[21] S. Chiu, W.-k. Liao, and A. Choudhary, "Design and evaluation of distributed smart disk architecture for i/o-intensive workloads," in *Proceedings of International Conference on Computational Science*.Springer, **(2003)**, pp. 230–241.

[22] M. Franklin, R. Chamberlain, M. Henrichs, B. Shands, and J. White,"An architecture for fast processing of large unstructured data sets," in *Proceedings of the IEEE International Conference on Computer Design*, **(2004)**, pp. 280–287.

[23] M. Sivathanu, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau, "Evolving rpc for active storage," in *ACM SIGPLAN Notices*, vol. 37. ACM, **(2002)**, pp. 264–276.

[24] C. Chen, Y. Chen, and P. C. Roth, "Dosas: Mitigating the resource contention in active storage systems," in *Proceedings of the IEEE International Conference on Cluster Computing*, **(2012)**, pp. 164–172.

[25] J. Piernas, J. Nieplocha, and E. J. Felix, "Evaluation of active storage strategies for the lustre parallel file system," in *Proceedings of the 2007 ACM/IEEE conference on Supercomputing*. ACM New York, NY, USA, **(2007)**, pp. 1–10.

[26] L. Huston, R. Sukthankar, R. Wickremesinghe, M. Satyanarayanan, G. R. Ganger, E. Riedel, and A. Ailamaki, "Diamond: A storage architecture for early discard in interactive search," in *Proceedings of the 3rd USENIX Conference on File and Storage Technologies*. USENIX Association, **(2004)**, pp. 73–86.

[27] R. O. Weber, "Information technology–scsi object-based storage device commands-2 (osd-2), revision 5," Tech. Rep. Technical report, INCITS Technical Committee T10/1729-D, Jan **(2009)**.

[28] A. Devulapalli, I. Murugandi, D. Xu, and P. Wyckoff, "Design of an intelligent object-based storage device," **(2009)**.

[29] T. M. John, A. T. Ramani, and J. A. Chandy, "Active storage using object-based devices," *2008 Ieee International Conference on Cluster Computing*, **(2008)**, pp. 472–478.

[30] H. Song, Y. Yin, Y. Chen, and X.-H. Sun, "A Cost-IntelligentApplication-Specific Data Layout Scheme for Parallel File Systems," in*Proceedings of the 20th International Symposium on High Performance Distributed Computing*, **(2011)**, pp. 37–48.

[31] H. Song, Y. Yin, X.-H. Sun, R. Thakur, and S. Lang, "A Segment-LevelAdaptive Data Layout Scheme for Improved Load Balance in Parallel File Systems," in *Proceedings of the 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing* (CCGrid), **(2011)** , pp. 414–423.

[32] H. Song, H. Jin, J. He, X.-H. Sun, and R. Thakur, "A Server-Level Adaptive Data Layout Strategy for Parallel File Systems," in *Proceedings of the IEEE 26th International Parallel and Distributed Processing Symposium Workshops and PhD Forum*, **(2012)**, pp. 2095–2103.

[33] S. He, X.-H. Sun, B. Feng, and F. Kun, "Performance-aware data placement in hybrid parallel file systems," in *Proceedings of the 14 th International Conference on Algorithms and Architectures for Parallel Processing*, **(2014)**, pp. 563–576.

[34] X. Jiong, Y. Shu, R. Xiaojun, D. Zhiyang, T. Yun, J. Majors, A. Manzanares, and Q. Xiao, "Improving MapReduce Performance through Data Placement in Heterogeneous Hadoop Clusters," in *Proceedings of the 19th International Heterogeneity in Computing Workshop*, 19-23 April **(2010)**, pp. 1 – 9.

[35] Y. Liu, R. Gunasekaran, X. Ma, and S. S. Vazhkudai, "AutomaticIdentification of Application I/O Signatures from Noisy Server-SideTraces," in *Proceedings of the 12th USENIX conference on File and Storage Technologies*, **(2014)**, pp. 213–228.

[36] A. N. Lab, "MPICH2:A High Performance andWidelyPortableImplementationofMPI."[Online].Available:http://www.mcs.anl.gov/research/project-detail.php?id=2

[37] "Orange File System," http://www.orangefs.org/.

Authors

Xiangyu Li, is a Ph.D. candidate of the Computer School, Wuhan University, Wuhan, China. He received a B.A. degree from Huazhong University of Science and Technology, China, in 2003and a M.S. degree in computer science and technology from Wuhan University, China, in 2008. He is especially interested in file and storage systems, high performance computing, distributed system, and computer network.

Shuibing He, received the Ph.D. degree in computer science and technology from Huazhong University of Science and Technology, China, in 2009. He is now an assistant professor at Computer School of Wuhan University, China. His current research areas include parallel I/O systems, file and storage systems, high-performance computing, and distributed computing.

XianbinXu, graduated from the department of computer architecture in Huazhong University of science and technology and worked at Huazhong University of science and technology from 1977 to 1985.He got the Ph.D. degree from Computer School of Wuhan University. He is now a professor at Computer School of Wuhan University, China. His research interests focus on network storage, data grid, and distributed system.