Wuhan University
Journal of Natural Sciences

2016, Vol.21 No.3, 249-256

Article ID 1007-1202(2016)03-0249-08
DOI 10.1007/s11859-016-1167-4

Capability-Aware Data Placement for Heterogeneous Active Storage Systems

☐ **LI Xiangyu[1,2,3], HE Shuibing[1,2†], XU Xianbin[1], WANG Yang [4]**

1. School of Computer, Wuhan University, Wuhan 430072, Hubei, China;
2. State Key Laboratory of High Performance Computing, National University of Defense Technology, Changsha 410073, Hunan, China;
3. School of Computer Science, Wuhan Donghu University, Wuhan 430212, Hubei, China;
4. Shenzhen Institute of Advanced Technology, Chinese Academy of Science, Shenzhen 518055, Guangdong, China

Abstract: By moving computations from computing nodes to storage nodes, active storage technology provides an efficient for data-intensive high-performance computing applications. The existing studies have neglected the heterogeneity of storage nodes on the performance of active storage systems. We introduce CADP, a capability-aware data placement scheme for heterogeneous active storage systems to obtain high-performance data processing. The basic idea of CADP is to place data on storage nodes based on their computing capability and storage capability, so that the load-imbalance among heterogeneous servers can be avoided. We have implemented CADP under a parallel I/O system. The experimental results show that the proposed capability-aware data placement scheme can improve the active storage system performance significantly.

Key words: active storage; parallel I/O system; CADP; data placement

CLC number: TP 393

Received date: 2015-12-10

Foundation item: Supported by the National Science and Technology Foundation of China (61572377), the Natural Science Foundation of Hubei Province (2014CFB239), the Open Fund from HPCL (201512-02), the Open Fund from SKLSE (2015-A-06), and the US National Science Foundation(CNS-1162540)

Biography: LI Xiangyu, male, Ph.D. candidate, research direction: file and storage systems, high performance computing, distributed system, and computer network. E-mail: xylee@whu.edu.cn

† To whom correspondence should be addressed. E-mail: heshuibing@whu.edu.cn

0 Introduction

Over the past decades, many scientific applications in the high-performance computing (HPC) domains, such as astrophysics, geographic systems, climate modeling, medical image processing, and high-energy physics, have become increasingly data-intensive [1]. For example, the astro program in astronomy generates tens of gigabytes of data in one run [2]. In some climate modeling and combustion simulation applications, the data set sizes range between 100 TB and 10 PB [3].

Moving such large data volumes between computing nodes and storage nodes takes a large amount of time, even on today's highest-performing computer systems. The main reason is that, although the performance of each hardware component of a computer system is continuously increasing with the advancements of VLSI technology, the I/O bandwidth between computing nodes and storage nodes has not improved as the same rate as the data requirements of applications. Data storage and analysis have become a serious bottleneck for data-intensive applications.

Active storage provides a promising solution to addressing the limited I/O bandwidth issue [4-8]. The main idea of active storage is to move computation from computing nodes to storage nodes. By offloading appropriate data operations to storage nodes and directly processing the data on storage nodes, active storage can not only reduce the network traffic, but also provide significantly aggregative processing capability when multiple devices are used in parallelism. Due to its efficiency in reducing network and disk traffic, active storage has

received intensive attention [9-11].

While effective to improve I/O system performance, most of the current active storage studies are designed for homogeneous I/O environments, that is, the storage nodes are usually built with the same hardware platforms in terms of CPU, memory and storage device. Since each node is homogeneous and with the same data process capability, active storage system usually uses an even data placement scheme to distribute data on multiple nodes, so that each node can finish the active storage operations nearly at the same time.

However, in a practical active storage system, the storage nodes may run with heterogeneous hardware resource: the CPU, memory and storage device of each node can be greatly different. Therefore, each storage node has a different data process capability. For the same amount of data needed to process, the high-performance storage nodes will finish the tasks quickly while the low-performance nodes will finish slowly. Since each node will finish their active storage operations with different rate and the overall data processing time depends on the straggler of all storage nodes, traditional data placement schemes will degrade the overall system performance.

It is common that storage nodes have heterogeneous hardware resources in a practical I/O systems. The first scenarios appears when a component of a storage node fails and it needs to be replaced by a new one [12]. As VLSI technology improves quite rapidly, it is quite probable for the new storage nodes to be faster and larger than the ones already in the I/O system. The second scenario appears when the system needs to grow its storage capability and new nodes have to be acquired. It will also be difficult to buy the same nodes as the ones in the original configuration, and thus newer storage nodes will be added. In both cases, it will make the traditional active storage system with sub-optimal performance.

In this paper, we propose CADP (capability-aware data placement), a novel data placement scheme to distribute data for heterogeneous active storage systems. CADP places data on each node based on a holistic metric-processing ratio, which considers the computing capability and storage capability of the node, so that the load-imbalance among heterogeneous servers can be avoided. In summary, we make the following contributions.

- We develop a novel metric to evaluate the data processing capability of each storage node in a heterogeneous cluster.
- We propose a data placement scheme, CADP, which distributes the data on each storage node based on their data processing capability.
- We implement a prototype of CADP under a parallel I/O system, and evaluated its performance with typical applications. The experimental results show that CADP can significantly improve the active storage system performance.

The rest of this paper is organized as follows. In Section 1, we describe the related work. Then, we describe the design and implementation of CADP in Section 2. Section 3 gives the performance evaluation. Finally, we conclude the paper in Section 4.

1 Related Work

1.1 Device-Level Active Storage

By offloading computing operations to storage devices, active storage technology can largely improve the computer system performance. Active storage is first proposed to exploit the computing intelligence inside disk drives. These techniques are either designed for general applications [5,6,13], or some special fields (e.g, database [14,15]). Both hardware architectures [13,16,17] and programming models [5,6] are studied to address the I/O bottleneck problem. A stream-based programming model of active disks was proposed in Ref.[5], which divides an application into a host part and a disk-resident part. Ref.[6] presented a detailed analysis of active disks for scan-intensive database applications. However, these efforts are only dedicated to utilize the power of embedded processor, thus the systems provide limited computation-offloading capability.

1.2 System-Level Active Storage

With the performance improvements on storage nodes, active storage can also be applied to file system. Sivathanu *et al* [18] extended the remote procedure call (RPC) to implement the prototype of active storage. Ma *et al* [4] proposed a multi-view storage system architecture with virtual file system technology to provide a flexible active storage. Piernas *et al* [19] gave an active storage strategy implemented in Lustre parallel file system. Son *et al* [3] enabled the active storage implementation in PVFS. Chen *et al* [1,20] considered data dependences and data contention issues in active storage system.

Due to the benefits of object storage technology, many researchers made efforts to integrate active storage into the object-based storage systems. Huston *et al* [21] used an active storage architecture for interactive search

of non-indexed data. This system does not comply with the T10 OSD standard [22]. To promote the development of object-based storage technology, several studies integrated the active storage technology into the object-based storage system based on the OSD standard [8,9, 23-25].

Most of the above studies are designed for homogeneous storage clusters. In contrast, this study proposes a data placement scheme to improve the active storage performance in heterogeneous I/O systems. While recent work SDD [26] introduces a data distribution scheme for heterogeneous active storage systems, it does not consider the heterogeneity of processor and the hardware parameters in a real system are hard to be obtained. In contrast, CADP addresses these issues by utilizing the relative execution time of the active storage operation.

1.3 Data Placement in Parallel File Systems

Parallel file systems usually provide several data placement policies [27], such as simple stripe, two dimensional stripe, and variable stripe, each suitable for a special kind of workloads. For complex workloads, 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 [28]. Server-level adaptive placement strategies adopt different stripe sizes on different file servers to improve the overall I/O performance [29]. These efforts are devoted to homogeneous systems. For heterogeneous file systems, recent studies [30-34] uses skewed data distribution to place data on HDD and SSD-based storage nodes.

All these studies only consider two kinds of nodes and they focused on storage performance without considering the CPU and memory impacts on the overall system performance. As opposite to these efforts, CADP can be applied to a more general heterogeneous system and it fully considers the holistic processing capability of each storage nodes to improve the system performance.

2 Capability-Aware Data Placement

In this section, we first introduce the basic idea of our proposed data placement scheme. Then we describe the data processing capability measurement and the algorithm used to determine the optimal data block placement on each server. Finally, we present the implementation of CADP.

2.1 The Basic Architecture of CADP

There are two typical models to deploy computing and storage nodes for data-intensive applications. The first one is to separate computing nodes from storage nodes, which is widely used in MPI applications. The second one is to perform computing and storage operations in the same node, which is adopted by MapReduce/Hadoop applications. In this paper, we focus on the first model since it is the most common architecture in HPC systems.

Figure 1 illustrates the system architecture for which the proposed data placement scheme is designed. In these systems, the user data are distributed to multiple servers with a data placement scheme. The metadata sever (MDS) responds to manage the block allocation information, and the data would be processed in each node when the active storage code is executed. As the servers are homogeneous, the main idea of the capability-aware data placement scheme is to place data on each node according to their data processing performance, instead of with an even data distribution widely used in current active storage systems. To this end, there are two components, profiling ratios (PR) and data placements (DP), located in MDS. PR is responsible for gathering the execution time of active storage codes downloaded to each node, and using these times to compute the profiling ratio of the node, which is fed to DP to generate the data placement plan. Applications interact with the client of parallel file system (PFS) for normal I/O operations, and the PFS API is instrumented to contact with MDS where PR and DP are located to obtain the allocation scheme for the write data. According to the placement scheme, the write data are partitioned into several regions, each assigned to a node based on its relative profiling ratio as shown in Fig. 1.

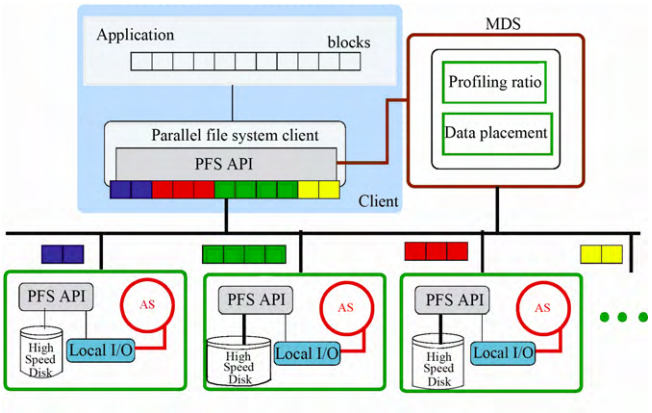

Fig. 1 An overview of the data placement framework for heterogeneous active storage system

The capability-aware active storage system works as follows. First, applications pass the active storage codes to the servers through an active storage client API.

Once the server installs these codes, it will begin to execute the active storage operations with a given-size input data. To get the exact performance profile of the node, measurement codes exercise the local I/O, which is delegated to the PFS APIs. Finally, the performance profile of each node is sent back to PR on MDS, and DP makes the final data placement plan for a large file according to the guidance of PR as we mentioned above.

2.2 Heterogeneous Data Processing Capability Measuring

To achieve the proposed data placement scheme, we first need to evaluate each node's data processing capability. In a practice I/O system, determining this factor is challenging for the following two reasons. First, the data processing time is related with the hardware resource of each node. As we discussed previously, each storage node in a heterogeneous active storage environment may have different CPU, memory and storage device. Since the hardware resource includes multiple components, only using computing power or storage performance can not accurately evaluate the overall data processing capability of each node.

Second, the data processing speed of each storage node also relies on the application itself. For different applications, the data processing speed can vary significantly because one application can require different hardware resource. For example, one application can be more CPU-intensive, but another application can be more I/O-intensive because it has a large fraction of I/O operations on storage devices. Thus, the heterogeneity measurements in the storage cluster may change while we execute active operations from different applications.

To address above issues, we introduce a simple but effective metric, processing ratio, to measure each node's data processing capability in a heterogeneous active storage system. processing ratio is a holistic parameter considering both hardware and application factors. We get the processing ratios through a profiling procedure as Algorithm 1, which includes the following steps.

① The active storage operations (data processing codes) of the HPC application are downloaded to each storage node.

② We carry out the operations separately on each node. To fairly compare the data processing speed, we limit all nodes to process the same amount of data. For example, in our experiments we limit the input data size on each node to 256MB to fully reflect its performance capability while not consuming too much time.

③ We record the data processing time on each node. The longest processing time is used as a reference to normalize the data processing time measurements.

④ The normalized values, called processing ratios (pr), are employed by the data placement algorithm to allocate file blocks for the given HPC application.

Algorithm 1 Processing capability determination

1. **procedure** CALCULATING PROCESSING RATIO
2. ▷the following loop can be done in parallel
3. **for** each storage node i($i\in[1, n]$) **do**
4. Download processing operations to node i
5. Execute operations with a fixed-size input data
6. Record its data processing time T_i
7. **end for**
8. $S \leftarrow \varnothing$
9. $\Gamma \leftarrow \max\{T_1, T_2, \cdots, T_n\}$
10. **for** ($\forall i\in[1, n]$) **do**
11. $\mathrm{pr}_i \leftarrow \dfrac{\Gamma}{T_i}$
12. $S \leftarrow S \bigcup \{\mathrm{pr}_i\}$ ▷S gathers all pr_i
13. **end for**
14. **return** S
15. **end procedure**

We download the data processing codes as active storage operations instead of pre-installing some measurement codes in advance because of the following reasons. First, both the files to be stored and the node processing ratios are correlated to the measurement codes as well as their input data. Second, any pre-defined codes will make us loss the flexibility to configure the codes according to the files to be stored. Given these benefits, by moving computations from computing nodes to storage nodes, active storage technology provides a promising solution for data-intensive high-performance computing applications.

We use an example to illustrate how to calculate the processing ratio of each node. Suppose there are four heterogeneous storage nodes (i.e., node A, B, C and D) in a heterogeneous active storage system. After the data process operations on each node, the processing time of the application on each node is 50, 100, 50 and 20 seconds, respectively. The processing time of node B is the longest, thus its processing ratio is set to 1, which is a reference used to determine processing ratios of other nodes. Therefore, the processing ratios of node A, C, and D are 2, 2 and 5, respectively.

2.3 Capability-Aware Data Placement Scheme

Based on the data processing capability of each node, we devise a heuristic iterative algorithm to determine the appropriate data placement on each node. Assuming there are m blocks in a parallel file needing to be processed, the goal of the algorithm is to distribute this blocks on the underlying n nodes. The traditional data placement algorithm adopts a fixed-size block (file stripe) to distribute data on multiple nodes. In particular, these blocks are located on the nodes in a round-robin way, so that each node has the same number of blocks of the file. Instead of using an even data distribution on each node, the proposed capability-aware data placement scheme improves the existing approach by differentiating the weight of each node to dispatch data blocks.

Algorithm 2 illustrates the data placement procedure of the CADP scheme. The basic idea of this scheme is to allocate a region of blocks according to the node's relative processing ratio. Specifically, after initializing varphi, the sum of processing ratios of all nodes, a, and R, the region start point and the allocation scheme (Lines 3-4), the algorithm enters a loop to allocate a region to each node (Lines 5-12). The size of the region is first determined by the node's relative processing ratio (Lines 6-7), and then the location of the region is identified and gathered in vector R (Lines 8-10). Finally, the starting point of next region is reset (Line 11). After the regions for all nodes are achieved, the algorithm returns the scheme (Line 13).

Algorithm 2 Capability-Aware date placement scheme

1. **procedure** PLACEMENT ($W[1, m]$) ▷ $W[1; m]$ is the block array to be stored
2. $\varphi \leftarrow \sum_{i=1}^{n} \mathrm{pr}_i$ ▷ pr_i : global variables
3. $a \leftarrow 0$ ▷ (a,b) represents a region
4. $R \leftarrow \varnothing$
5. **for** ($\forall\, i \in [1, n]$) **do**
6. $\omega_i \leftarrow \frac{\mathrm{pr}_i}{\varphi}$
7. $m_i \leftarrow m \times \omega_i$ ▷ m_i: region size
8. $b \leftarrow \lceil a + m_i \rceil$
9. ▷ R gathers the placement scheme
10. $R \leftarrow R \cup \{W[a, b]\}$
11. $a \leftarrow b$
12. **end for**
13. **return** R
14. **end procedure**

Note that although the CADP algorithm is relatively simple, it is a generalization of those existing approaches that target the nodes with homogeneous capability. Logically, our algorithm would default to the current data placement schemes if all the storage nodes have the same data processing capability.

2.4 Implementation Issues

We implement the proposed data placement scheme in a parallel I/O system. The I/O middleware is MPICH2 and the parallel file system is OrangeFS. 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 processing ratios to evaluate the holistic data processing capability of heterogeneous storage nodes.

During the data placement phase, we distribute the file data with the optimal data layout. For ease of implementation, we assign different number of blocks on each node by specifying the stripe size on each node. For example, for a default stripe size 64 KB, one node will be allocated two blocks if we configure the stripe size as 128 KB. The OrangeFS file system supports an API for implementing specific variable stripe distribution. In OragneFs, 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 of directories where the application files are located. For POSIX interface, we use the "setfattr" command to reach the similar capability-aware data placement goal.

3 Performance Evaluation

3.1 Experimental Setup

We conducted the experiments on a Linux cluster. We choose two nodes as the computing nodes, each having two AMD Opteron(tm) processors and 8 GB memory. To simulate heterogeneous storage servers, we choose two types of file servers. The low-performance storage servers are configured with Intel i5, 250 GB HDD, and 4 GB memory. The high-performance storage servers are configured with Intel i7 processor, 100 GB SSD, and 16 GB memory. All nodes are equipped with Gigabit Ethernet interconnection. The operating system is Ubuntu 9.04, 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 HDD-based servers and four SSD-based servers unless otherwise specified.

To demonstrate the effectiveness of our proposed data placement scheme, we evaluate the system performance under three schemes: traditional storage (TS), traditional active storage (TAS), and heterogeneous active storage (HAS). In TS, the servers are responsible for normal I/O operations. The active data processing operations are carried on the clients; In TAS, the data is distributed on servers with a default block size (64 KB) in a round-robin way. Each node has an even data distribution. This is the default data placement scheme adapted by current active storage systems; In HAS, we implement the proposed data placement scheme CADP on the active storage system.

We use two typical applications, data compression and data selection, to evaluate the system performance. The first application is responsible for compressing user's data with a configurable size. Data compression is a representative operation in file system and is widely used in a space-constrained storage environment. In our test, we use a parallel file to store the user's data, and each storage node is only limited to process data on itself.

3.2 Data Compression

In TS, the clients first fetches the given-length data from the parallel file system to their local memory through a READ interface, and then carry out the data compression operations on the clients. After that, the clients write the result back to the parallel file system. The execution time of the application consists of data read time through network, data compression time on the clients, and data writing back time to the storage server. In both TAS and HAS, the clients first download the compression code onto the storage server, and then execute the data compression on the server. Finally, the server stores the result onto itself. The execution time of the application consists of all the above mentioned parts.

Figure 2 shows the execution time of the data compression application running with TS, TAS, and HAS, respectively. The file size is 2 GB. We test the performance of the application under 10%, 30%, 50%, and 70% of the data is compressed. These results show that, when the data amount is very small (10% data is compressed), TAS and HAS shows comparable performance as TS. This is because the network and storage data movement is not very large, the reduced I/O time is not obvious. However, when large data movement occurs, namely 50% to 70% of the data needed to be compressed, TAS and HAS have much better performance than TS. The improvements are attributed to the reduced data movement through network and the increased data processing capability on multiple servers. Compared with TAS, HAS has a better behavior. This is because TAS could lead to an imbalanced resource utilization of servers due to the even data placement. The proposed data placement scheme considers the data processing capability of heterogeneous storage servers and addresses the limitation of existing active storage systems well.

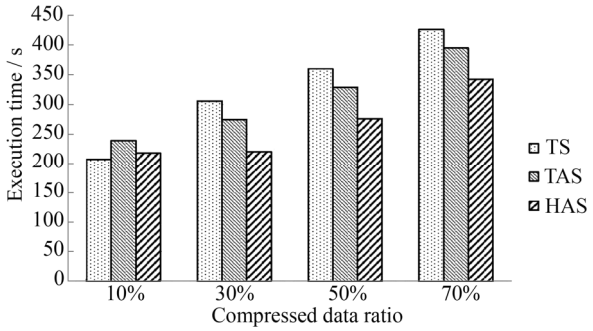

Fig. 2 Performance comparison of data compression application under different schemes

3.3 Data Selection

In TS, the clients first fetches the given-length data from the servers to their local memory, and then carry out the data selection operations on the clients. The execution time of the application consists of data read time through network and data selection time on the clients. Similarly, in both TAS and HAS, the clients first downloads the data selection code onto the server, and then executes the data selection operations on them. 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 4 GB parallel file on multiple servers.

Figure 3 describes the application execution time under different data selection conditions running TS, TAS, and HAS, respectively. The ratio means how much data should be selected from the original data set. From the figure we can observe that the active storage system is always better than the traditional storage system. More

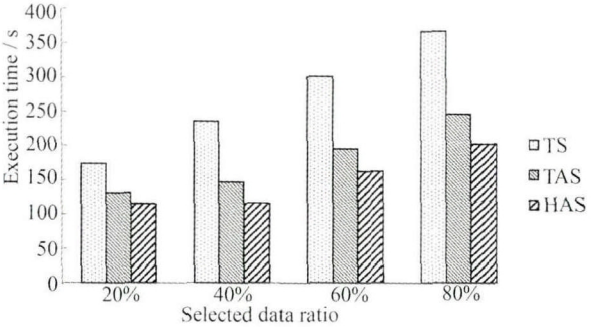

Fig. 3 Performance comparison of data selection application under different schemes

important, heterogeneous-aware active storage has the best performance. This is because HAS adopts the capability-aware data placement scheme to distribute data on multiple servers, which can effectively eliminate the load-imbalance issues in current active storage systems. These results show that our proposed data placement scheme is an efficient way to improve the I/O system performance for data-intensive high-performance computing applications.

4 Conclusion

The great processing capability of storage nodes makes it feasible to implement active storage technology in parallel I/O systems. However, the heterogeneity of storage servers leads to critical challenges to the data placement schemes in current active storage systems. In this research, we introduce CADP, a capability-aware data placement scheme for heterogeneous active storage system to achieve high-performance data processing. CADP places data on storage nodes based on a holistic metric-processing ratio, which considers the computing capability and storage capability of the node, so that the load-imbalance among heterogeneous servers can be avoided. Experimental results show that the proposed capability-aware data placement scheme can significantly improve the active system performance.

References

[1] Chen C, Chen Y. Dynamic active storage for high performance I/O [C] // *Proceedings of the* 41*st International Conference on Parallel Processing.* Washington D C: IEEE Press , 2012: 379-388.

[2] Kandemir M, Son S W, Karakoy M. Improving I/O performance of applications through compiler-directed code restructuring [C] // *Proceedings of the* 6*th USENIX Conference on File and Storage Technologies*, *FAST* ' 08. San Jose: USENIX Association, 2008: 159-174.

[3] Son S W, Lang S, Carns P, *et al*. Enabling active storage on parallel I/O software stacks [C]// *Proceedings of the IEEE* 26*th Symposium on Mass Storage Systems and Technologies*, *MSST*. Washington D C: IEEE Press, 2010: 1-12.

[4] Ma X N, Reddy A L N. MVSS: An active storage architecture [J]. *IEEE Transactions on Parallel and Distributed Systems*, 2003, **14**(10): 993-1005.

[5] Acharya A, Uysal M, Saltz J. Active disks: Programming model, algorithms and evaluation [J]. *ACM SIGPLAN Notices*, 1998, **33**(11): 81-91.

[6] Riedel E, Gibson G A, Faloutsos C. Active storage for large-scale data mining and multimedia [C] // *Proceedings of the* 24*rd International Conference on Very Large Data Bases*, VLDB. New York: Morgan Kaufmann Press, 1998: 62-73.

[7] Tang H, Gulbeden A, Zhou J, *et al.* The panasas active scale storage cluster-delivering scalable high bandwidth storage [C] // *Proceedings of the ACM/IEEE SC2004 Conference on Supercomputing*. Washington D C: IEEE Press, 2004:53-62.

[8] He S B, Xu X B, Yang Y H. Oasa: An active storage architecture for object-based storage system [J]. *International Journal of Computational Intelligence Systems*, 2012, **5**(6): 1173-1183.

[9] Xie Y, Muniswamy-Reddy K, Feng D, *et al*. Design and evaluation of Oasis: An active storage framework based on T10 OSD standard [C] // *Proceedings of the IEEE* 27*th Mass Storage Systems and Technologies*, *MSST*. Washington D C: IEEE Press, 2011: 1-12.

[10] Tiwari D, Boboila S, Vazhkudai S S, *et al*. Active flash: Towards energy-efficient, in-situ data analytics on extreme-scale machines [C] // *Proceedings of the* 11*th USENIX Conference on File and Storage Technologies*, *FAST*'13. San Jose: USENIX Association, 2013:119-132.

[11] Rich B, Thain D. Datalab: Transactional data-parallel computing on an active storage cloud [C] // *Proceedings of the* 17*th International Symposium on High Performance Distributed Computing*, *HPDC*'08. Boston: Association for Computing Machinery Press, 2008: 233-234.

[12] Cortes T, Labarta J. Taking advantage of heterogeneity in disk arrays [J]. *Journal of Parallel and Distributed Computing*, 2003, **63**(4): 448-464.

[13] Keeton K, Patterson D A, Hellerstein J M. A case for intelligent disks (IDISKs) [J]. *ACM SIGMOD Record*, 1998, **27** (3): 42-52.

[14] Su W Y S, Lipovski G J. Cassm: A cellular system for very large data bases [C] // *Proceedings of the International Conference on Very Large Data Bases*, *VLDB*. Framingham : Association for Computing Machinery, 1975: 456-472.

[15] Ozkarahan A E, Schuster S A, Smith K C. Rap: An associative processor for data base management [C]// *Proceedings of the AFIPS Joint Computer Conferences*. Washington D C: IEEE Press 1975: 379-387.

[16] Chiu S, Liao W K, Choudhary A. Design and evaluation of distributed smart disk architecture for I/O-intensive workloads [C] //*Proceedings of International Conference on Computational Science*, *ICCS*'03. Berlin: Springer-Verlag, 2003: 230-241.

[17] Franklin M, Chamberlain R, Henrichs M, *et al*. An architecture for fast processing of large unstructured data sets [C]// *Proceedings of the IEEE International Conference on Computer Design*, *ICCD*'04. San Jose: Institute of Electrical and Electronics Engineers, 2004: 280-287.

[18] Sivathanu M, Arpaci-Dusseau A C, Arpaci-Dusseau R H. Evolving RPC for active storage [J]. *ACM SIGPLAN Notices*, 2002, **37**(10): 264-276.

[19] Piernas J, Nieplocha J, Felix E J. Evaluation of active storage strategies for the lustre parallel file system [C] // *Proceedings of the* 2007 *ACM/IEEE Conference on Supercomputing*, *SC*'07. New York: Association for Computing Machinery, 2007: 1-10.

[20] Chen C, Chen Y, Roth P C. Dosas: Mitigating the resource contention in active storage systems [C] // *Proceedings of the IEEE International Conference on Cluster Computing*, *CLUSTER*'12. Washington D C: IEEE Press, 2012: 164-172.

[21] Huston L, Sukthankar R, Wickremesinghe R, *et al*. Diamond: A storage architecture for early discard in interactive search [C] // *Proceedings of the 3rd USENIX Conference on File and Storage Technologies*, *FAST*'04. San Francisco: USENIX Association, 2004: 73-86.

[22] Weber R O. Information technology—SCSI object-based storage device commands-2(osd-2), Revision 5[R]. Oklahoma: INCITS Technical Committee *T*10/1729-*D*, 2009.

[23] Qin L, Feng D. Active storage framework for object-based storage device [C] // *Proceedings of the IEEE* 20*th International Conference on Advanced Information Networking and Applications*. Washington D C: IEEE Press, 2006: 97-101.

[24] Devulapalli A, Murugandi I, Xu D, *et al*. *Design of an Intelligent Object-Based Storage Device* [M]. New York: Springer -Verlag, 2009.

[25] John T M, Ramani A T, Chandy J A. Active storage using object-based devices [C] // 2008 *IEEE International Conference on Cluster Computing*. Washington D C: IEEE Press, 2008: 472-478.

[26] Li X Y, He S B, Xu X B. Skewed data distribution for active storage systems on hybrid servers [J]. *International Journal of Grid and Distributed Computing*, 2016, **9**(5): 51-62.

[27] Song H, Yin Y, Chen Y, *et al*. A cost-intelligent application-specific data layout scheme for parallel file systems [C] // *Proceedings of the* 20*th International Symposium on High Performance Distributed Computing*. San Jose: IEEE Press, 2011: 37-48.

[28] Song H, Yin Y, Sun XH, *et al*. A segment-level adaptive data layout scheme for improved load balance in parallel file systems [C] // *Proceedings of the* 11*th IEEE/ACM International Symposium on Cluster*, *Cloud and Grid Computing*, CCGrid'11. Washington D C: IEEE Press, 2011: 414-423.

[29] Song H, Jin H, He J, *et al*. A server-level adaptive data layout strategy for parallel file systems [C] // *Proceedings of the IEEE* 26*th International Parallel and Distributed Processing Symposium Workshops and Ph.D. Forum*, PDPSW'12. Washington D C: IEEE Press, 2012: 2095-2103.

[30] He S B, Sun X H, Feng B, *et al*. Performance-aware data placement in hybrid parallel file systems [C] // *Proceedings of the* 14 *th International Conference on Algorithms and Architectures for Parallel Processing*, *ICA*3*PP*. New York: Springer-Verlag, 2014: 563-576.

[31] He S B, Liu Y, Sun X H. PAS: A performance and space-aware data layout scheme for hybrid parallel file systems [C] // *Proceedings of the Data Intensive Scalable Computing Systems Workshop*, *DISK*'14. Washington D C: IEEE Press, 2014: 41-48.

[32] He S B, Sun X H, Haider A. HAS: Heterogeneity-Aware selective data layout scheme for parallel file systems on hybrid servers [C] // *Proceedings of* 29*th IEEE International Parallel and Distributed Processing Symposium*, *IPDPS*'15. Washington D C: IEEE Press, 2015: 613-622.

[33] He S B, Sun X H, Wang Y, *et al*. A heterogeneity-Aware region-level data layout scheme for hybrid parallel file systems [C] // *Proceedings of the* 44*th International Conference on Parallel Processing*, *ICPP*'15. Washington D C: IEEE Press, 2015: 340-349.

[34] He S B, Wang Y, Sun X H. Boosting parallel file system performance via heterogeneity-aware selective data layout [J]. *Journal IEEE Transactions on Parallel and Distributed System*, 2015, **99**: 1-14.

□