Evaluation of A Performance Model of Lustre File System

Tiezhu Zhao[1], Verdi March[2,3], Shoubin Dong[1], Simon See[2,4]
[1]Guangdong Key Laboratory of Computer Network, South China University of Technology, Guangzhou, 510641, P.R.China
[2]Asia-Pacific Science and Technology Center (APSTC), Sun Microsystems
[3]Department of Computer Science, National University of Singapore
[4]Department of Mechanical & Aerospace Engineering, Nanyang Technological University
zhao.tiezhu@mail.scut.edu.cn, Verdi.March@Sun.COM , sbdong@scut.edu.cn , Simon.See@Sun.COM

***Abstract*—As a large-scale global parallel file system, Lustre file system plays a key role in High Performance Computing (HPC) system, and the potential performance of such systems can be difficult to predict because the potential impact to application performance is not clearly understood. It is important to gain insights into the deliverable Lustre file system IO efficiency. In order to gain a good understanding on what and how to impact the performance of Lustre file system. This paper presents a study on performance evaluation of Lustre file systems and we propose a novel relative performance model to predict overhead under different performance factors. In our previous experiments, we discover that different performance factors have a closed correlation. In order to mining the correlations, we introduce relative performance model to predict performance differences between a pair of Lustre file system equipped with different performance factors. On average, relative model can predict bandwidth within 17%-28%. The results show our relative prediction model can obtain better prediction accuracy.**

Keywords-performance evaluaion; parallel file system; model; lustre

I. INTRODUCTION

Parallel file system is a key part of any complete massively parallel computing environment and widely used in clusters dedicating to I/O-intensive parallel applications. Lustre parallel file system is best known for powering seven of the ten largest high-performance computing (HPC) clusters in the world with tens of thousands of client systems, petabytes (PBs) of storage and hundreds of gigabytes per second (GB/sec) of I/O throughput. Many HPC sites use Lustre as a site-wide global file system, servicing dozens of clusters on an unprecedented scale [1]. Currently, in the cloud computing era, the performance research of Lustre file system increasingly attracted the attention of industry and research communities. Further details on Lustre are available in [9][10][11][12].

The rapid development of HPC application is aggressively pushing the demand of parallel file system in terms of high aggregated I/O bandwidth, mass storage capacity and high data fault-tolerant etc. HPC platforms need to be coupled with efficient parallel file systems, such as Lustre file system, that can deliver commensurate IO throughput to scientific applications. Although the various performance characteristics in HPC workloads have been researched via experimental analysis and empirical analysis, the potential performance of such systems can be difficult to predict because the potential impact to application performance in parallel file system environment is not clearly understood and most internal details of the basic components of parallel file system are not public. It is important to gain insights into the deliverable Lustre file system IO efficiency.

As we known, the construction of parallel file system is much more expensive and complex. When a parallel file system is not properly tuned or configured, this cost may not be paid off. So, issues on how to optimize the design of a parallel file system, how to evaluate the performance of a parallel file system, how to tune the performance of a parallel file system and how to predict the performance trend are more and more concerned by both storage industry and research communities.

Storage system can be complex to manage. Management consists of storage device (volume or LUN) sizing, selecting a RAID level for each device, and mapping application workloads to the storage device. Automating management is one way to offer the administrator some relief and help reduce the total cost of ownership in the data center. In particular, one could automate the mapping of workloads to storage devices. Whereas, storage administration currently continue to be overly complex and costly, and face with many challenges involved in deciding the mapping from application data sets to storage devices, balancing loads, matching workload characteristics to device strengths. Unfortunately, storage administration currently relies on experts who use rules-of-thumb to make decision [18][19][20]. One need a mechanism for predicting the performance of any given workload and automates the prediction process.

Based on the above observations, we propose an in-depth performance evaluation of Lustre file system and our evaluation mainly covers the number of OSSes, storage connection approaches, the type of disks, the type of journal for OST and the number of threads/OST etc.. We deliver relational analysis for the performance overhead under different performance factors and discover that different performance factors have a closed correlation. In order to mining the potential performance correlations, we conduct a novel relative performance prediction model to predict performance under different factors.

The remainder of this paper is organized as follows. We firstly introduce related work in section II. Then, we conduct an in-depth survey on performance factor of Lustre file system in section III. In section IV, we present the relative performance prediction model and carry out detailed prediction performance analysis and conclude the paper in section V.

978-0-7695-4106-8/10 $26.00 © 2010 IEEE
DOI 10.1109/ChinaGrid.2010.38

II. RELATED WORK

Currently, works in parallel/distributed file system can be divided into five categories: (1) Metadata management and query optimization. Metadata management is critical in scaling the overall performance of large-scale data storage systems and a large-scale distributed file system must provide a fast and scalable metadata lookup service. E.g. Wang et. al. proposed a two-level metadata management method to achieve higher availability of the parallel file system while maintaining good performance [2]; (2) Performance parameter Analysis and tuning. Yu et. al. indicated excessively wide striping can cause performance. To mitigate striping overhead and benefit collective IO, authors proposed two techniques: split writing and hierarchical striping to gain better IO performance [3]. Yu et. al. presented an extensive characterization, tuning, and optimization of parallel I/O on the Cray XT supercomputer (named jaguar), and characterized the performance and scalability for different levels of storage hierarchy [4]; (3) Optimizing data distribution strategy. e.g. Li et. al. modeled the whole storage system's architecture based on closed Fork-Join queue model and proposed an approximate parameters analysis method to build performance model [5]. Yu et. al. adopted a user-level perspective to empirically reveal the implications of storage organization to parallel programs running on Jaguar and discovered that the file distribution pattern can impact the aggregated I/O bandwidth [6]; (4) Optimizing data access path. Juan Piernas et. al. adopted a novel user-space implementation of Active Storage for Lustre and the user-space approach has proved to be faster, more flexible, portable, and readily deployable than the kernel-space version [7]; (5) Availability and scalability. Zhang developed a new mechanism named Logic Mirror Ring (LMR) to improve the reliability and availability of the parallel file system. a logic mirror ring is built over all I/O nodes to indicate the mirror relationship among the nodes [8].

III. SURVEY ON PERFORMANCE FACTORS

Prior to the introduction of relative model, we firstly conduct a detailed survey on performance factors of Lustre file system by referring to extensive literatures such as [1],[3],[6], [7], [13], [14], [15], [16]. This part provides some details of performance factors and we categorize these factors as follows: the number of OSSes; the number of OSS/MDS threads; the type of journal for OST; the type of disks; storage connection method; striping pattern (stripe size, stripe count and stripe offset); read/write cache effect; the size and the number of inodes for OST/MDT; data distribution strategies etc. The detailed performance factors and the basic architecture of Lustre can be seen in Figure 1.

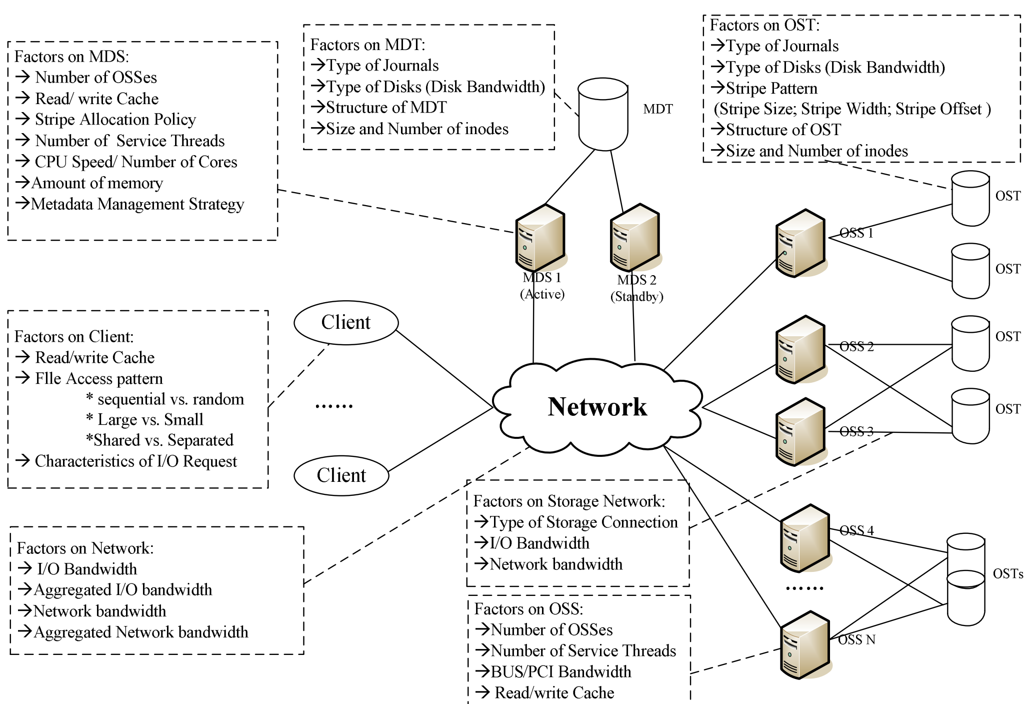

Figure 1. Performance factors and architecture of Lustre

Figure 1 shows the basic architecture of Lustre and the performance factors on each component including the interconnection network and storage network. And our following performance model mainly focuses on some important performance factors including the number of OSSes, he type of journal for OST, the type of disks, the number of OSS/MDS threads and storage connection approaches.

IV. RELATIVE PERFORMANCE MODEL

Conventional storage models tend to predict the performance of a workload on a given storage system [28][29] [30][31][32][33]. They are hard to capture the application-device feedback of a closed workload (which is affected by storage performance), can't obtain or concisely express the workload characteristics and easy to lost necessary information. Fortunately, M. P. Mesnier et. al. found that, for a given workload, the performance of one device is often the best predictor of the performance of another and proposed a relative fitness model to overcome the shortcoming of conventional model, which is a new black-box approach to modeling the performance of storage devices and predicts performance differences between a pair of devices [19][20]. In our previous experiments, we discover that different performance factors have a closed correlation. In order to mining the correlations, we introduced relative performance model to capture the differences between Lustre systems and learn to predict performance scaling factors.

A. Model Setup

The template is designed so that author affiliations are not repeated each time for multiple authors of the same affiliation. Please keep your affiliations as succinct as possible (for example, do not differentiate among departments of the same organization). This template was designed for two affiliations.

1) Model hypothesis

Hypothesis1: For a given workload, there are different workload characteristics on different Lustre system, which equipped with different performance factors.

Hypothesis2: Resource utilization, server&target utilization and performance can improve the prediction accuracy.

2) Model parameter

P_i : the performance metrics vector of $i\text{-}th$ Lustre system, performance metrics consist of bandwidth (MB/sec), throughput (IO/sec) and latency (msec).

$$P_i = [Bandwidth \quad Throughput \quad Latency]^T$$

W_i : the workload characteristics vector on $i\text{-}th$ Lustre system, mainly including No of threads/OST (THREAD), No of objects/OST (OBJ), read/write request size (RS), read-write ratio (RWR), queue depth (also known as multi-programming level or outstanding requests), I/O randomness (defined as random- sequential request ratio), request arrival rate , I/O inter-arrival delay, spatial locality, burstiness, spatiotemporal correlation etc. [19][20] [34] [35].

$$W_i = [THREAD_i \quad OBJ_i \quad RS_i \quad RWR_i \quad \cdots]^T$$

$RUtil_i$: the resource utilization vector of $i\text{-}th$ Lustre system, including CPU utilization (C), memory utilization (M), disk capacity utilization (D), cache hit rate etc.

$$RUtil_i = [C_i \quad M_i \quad D_i \quad \cdots]^T$$

$STUtil_i$:the server and target utilizations vector of $i\text{-}th$ Lustre system, which consist of OSS utilization (OSS), OST utilization (OST), MDS utilization (MDS), MDT utilization (MDT).

$$STUtil_i = [OSS_i \quad OST_i \quad MDS_i \quad MDT_i]^T$$

3) Model setup

a) Construcing relative performance model

We construct relative performance model for each pair of Luster systems to capture the differences between Lustre systems and learn to predict performance scaling factors, as shown in Figure 2.

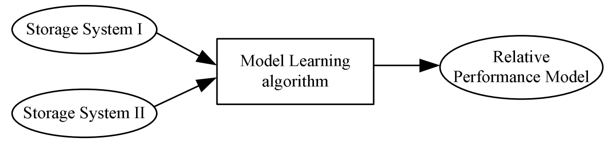

Figure 2. Construct relative model

For a specific system j , we firstly train a function ϕ_j to express Lustre system j and take as input the workload characteristics W_j and output a performance metric P_j when applications run on system.

$$P_j = \phi_j(W_j) \tag{1}$$

In order to capture the changes in workload characteristics from system i to system j , we define a function $\varphi_{i\to j}$ to predict the workload characteristics W_j given W_i .

$$W_j = \varphi_{i\to j}(W_i) \tag{2}$$

By combining formula (1) and (2), we obtain the composition of ϕ_j and $\varphi_{i\to j}$.

$$P_j = \phi_j(\varphi_{i\to j}(W_i)) \tag{3}$$

In addition to workload characteristics, performance, resource utilization and server&target utilization also are beneficial to predict performance. In other words, the performance, resource utilization and server&target utilization of one system can be used in predicting the performance of another.

$$P_i = \phi_j(\varphi_{i\to j}(W_j, P_j, RUtil_j, STUtil_j)) \tag{4}$$

Whereas, rather than learn two function, the composition of ϕ_j and $\varphi_{i\to j}$ can be expressed as a single composite function $CF_{i\to j}$.

$$P_j = CF_{i\to j}(W_i, P_i, RUtil_i, STUtil_i) \tag{5}$$

Because the performance ratios are better predictors for a new workload, we can predict performance ratios P_j/P_i , rather than predict raw performance value P_j .

$$\frac{P_j}{P_i} = RF_{i\to j}(W_i, P_i, RUtil_i, STUtil_i) \tag{6}$$

b) Using model to predict performance

After reducing the relative model, we using this model to predict performance, that is, we can train this relative model to predict system I's performance as a function of system II's workload characteristics, performance, resource utilization, server&target utilization, as shown in Figure 3.

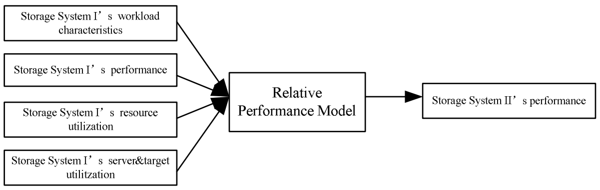

Figure 3. Predict performance

By solving formula (6), we can predict the performance value P_j.

$$P_j = RF_{i\to j}(W_i, P_i, RUtil_i, STUtil_i) \times P_i \quad (7)$$

B. *Experiment Setup*

Evaluation experiment are run on 2x Sun Fire X4240 (OSS nodes) which is equipped with 2x AMD Dual-core Opteron™ Processor 3GHz, DDR2 8GB of memory and 24x SAS Disks (300GB), 48x SATA Disks (1TB). The interconnect network is 10 Gigabit TCP/IP network. In software environment, we use Lustre 1.6.6 and Redhat Enterprise Linux 5.2.

As Lustre separates data from meta-data and selectively sends appropriate requests to OSS or MDS server, any standard sequential I/O benchmark that is running on Lustre clients will be translated to object protocol on servers. We need a tool that can mimic object I/O protocol of Lustre servers. OBDFilter is a layer in Lustre Object Storage Server I/O stack. OBDFilter-survey is a script that exercises this layer to create, delete, read, write objects and displays the performance of OSS. Since it correctly mimics standard Lustre I/O on OSS, it is chosen as benchmarking tool.

Based on survey on performance factors, our experiment mainly consider five factors: the number of OSSes (1OSS vs. 2 OSS), the type of journals (internal journal vs. external journal), the type of disks (SAS disk vs. SATA disk), storage connection approaches (directly connected vs. daisy-chain connected) and the number of threads and design 4 groups of cases to reveal the implication of Lustre file system under different factors situation and the detailed configuration of test cases can be seen in Table 1.

TABLE 1. THE CONFIGURATION OF CASES

Group	Case	Num of OSSes	Type of Journals	Storage Connection	Type of disks
1	1.1	1	external	direct	SAS
	1.2	2	external	direct	SAS
2	2.1	1	internal	direct	SAS
	2.2	1	external	direct	SAS
3	3.1	2	external	direct	SAS
	3,2	2	external	direct	SATA
4	4.1	2	external	direct	SATA
	4.2	2	external	daisy-chain	SATA

C. *Performance prediction*

Based on experiment setup, OBDFilter-survey is selected as the benchmarking tool to mimic object I/O protocol of Lustre servers. No of threads/OST take on a value from 8 to 128 (power of two), No of objects/OST varies from 1 to 8 (power of two), record size is set 1024 KB and I/O request size is 16 GB (large enough to avoid cache effects). Table 2 and Table 3 show a part of experiment test data, which represent test data of case1.1-W and case1.2-W, respectively.

TABLE 2. TEST DATA OF CASE1.1-W

Bandwidth (MB/Sec)	objects/OST			
Threads/OST	***1***	***2***	***4***	***8***
8	381	404	407	389
16	659	688	671	646
32	590	820	837	782
64	598	884	827	848
128	837	915	888	893

TABLE 3. TEST DATA OF CASE 1.2-W

Bandwidth (MB/Sec)	objects/OST			
Threads/OST	***1***	***2***	***4***	***8***
8	430	492	441	425
16	622	702	657	643
32	728	803	816	754
64	846	942	922	869
128	817	937	1116	1006

Based on the previous work, we discover that different performance factors have a closed correlation. In order to mining the correlations, we introduced relative performance model. In this part, we choose Classification And Regression Trees (CART) as the model learning algorithm. CART modeling is a machine learning tool that can approximate real functions in multi-dimensional Cartesian space. It can also be thought of as a type of non-linear regression [32]. CART models predict a desired value based on predictor variables. In our case, the predictors are workload characteristics, performance, and resource utilization and server&target utilization; and the predicted variable is a relative performance value.

In order to better understand our relative model, we construct an example of training samples (Table 4) from the subset of Table 2 and Table 3.

TABLE 4. AN EXAMPLE OF TRAINING SAMPLES FOR CART MODEL

Threads/OST	Objects/OST	$RF_{i->j}$
8	1	1.13
16	2	1.02
32	1	1.23
64	2	1.06

Figure 4 shows the steps taken by CART in building a regression tree from the samples in Table 4. CART determines which split is "best" by inspecting the similarity of the samples in the leaf nodes. In this example, (b) (objects/OST) is a better first split than (a) (threads/OST). For the next split, CART then selects the threads/OST. A detailed discussion of CART is available in [36] [20].

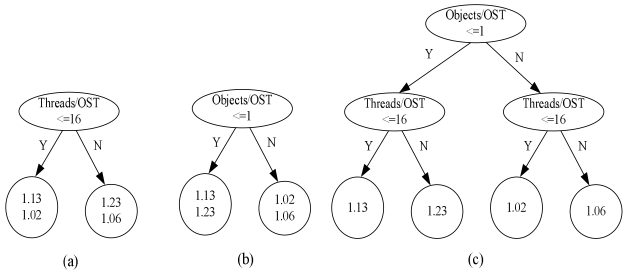

Figure 4. An example of the steps taken by CART

Assuming a new workload is configurated with 8 threads/OST and 2 objects/OST, running on $i\text{-}th$ and $j\text{-}th$ Lustre system, respectively. And the actual performance of $i\text{-}th$ and $j\text{-}th$ Lustre system are 414 MB/sec and 446 MB/sec, respectively. So, we can obtain the predicted value 414*1.02=422.28 MB/sec, then the relative error is (446-422.28)/464*100%=8.99%.

Using the similar way, we can conduct performance prediction using case1.1-case1.2, case2.1-case2.2, case3.1-case3.2, case4.1-case4.2. And the average relative errors of prediction results can be found in Table 5, Table 6 and Table 7 (Notes: A.R.E=Average Relative Error)

TABLE 5. AVERAGE RELATIVE ERRORS OF PREDICTION RESULTS (W)

Predictor variable	**Predicted variable**	**A.R.E.**
case1.1-W	case1.2-W	18.52%
case2.1-W	case2.2-W	27.31%
case3.1-W	case3.2-W	19.09%
case4.1-W	case 4.2-W	23.45%

TABLE 6. AVERAGE RELATIVE ERRORS OF PREDICTION RESULTS (REW)

Predictor variable	**Predicted variable**	**A.R.E.**
case1.1-ReW	case1.2-ReW	21.40%
case2.1-ReW	case2.2-ReW	25.21%
case3.1-ReW	case3.2-ReW	17.15%
case4.1-ReW	case4.2-ReW	23.38%

TABLE 7. AVERAGE RELATIVE ERRORS OF PREDICTION RESULTS (R)

Predictor variable	**Predicted variable**	**A.R.E.**
case1.1-R	case1.2-R	24.16%
case2.1-R	case2.2-R	27.18%
case3.1-R	case3.2-R	21.75%
case4.1-R	case4.2-R	19.49%

As shown in Table 5 to 7, on average, our relative performance model can predict bandwidth within 17%-28%. The results show our relative prediction model can obtain better prediction accuracy. And these results also confirm the previous conclusion from Ref. [19][20], which found that, for a given workload, the performance of one device is often the best predictor of the performance of another.

V. CONCLUSION

In this paper, we present an in-depth efficient performance evaluation of Lustre file system. In our previous experiments, we discover that different performance factors have a closed correlation. In order to mining the correlations, we propose a relative performance model to predict performance differences between a pair of Lustre system equipped with different performance factors. At the beginning, we conduct a survey on performance factors which is the basic of our experiment and model analysis. Then, we introduce our relative performance model to capture the differences of Lustre systems and predict performance. In the experiment, we designed four group cases covering some important performance factors, such as the number of OSSes, storage connection approaches, the type of disks, the type of journal for OST and the number of threads/OST. Our relative performance model can obtain better prediction accuracy and the results confirm the previous conclusion which found that, for a given workload, the performance of one device is often the best predictor of the performance of another.

ACKNOWLEDGMENT

This work is supported by China Education and Research Grid Project (ChinaGrid), which is granted by the Ministry on Education of P.R. China.

We thank Atul Vidwansa from Sun Microsystems Inc. for his sincere help, especially in the experiment setup.

REFERENCES

[1] Sun Microsystems, Inc., "LUSTRE™ FILE SYSTEM", Oct. 2008.

[2] F. Wang,Y.L. Yue, D. Feng etc. "High Availability Storage System Based on Two-level Metadata Management", Proceedings of the 2007 Japan-China Joint Workshop on Frontier of Computer Science and Technology(FCST 2007), 2007, pp.41-48.

[3] W. Yu, J. S. Vetter, R. S. Canon etc., "Exploiting Lustre File Joining for Effective Collective IO", CCGrid 2007, 2007.

[4] W. Yu, J. S. Vetter, H. S. Oral, "Performance Characterization and Optimization of Parallel I/O on the Cray XT", IPDPS 2008, 2008.

[5] H. Y. Li, Y. Liu, Q. Cao, "Approximate parameters analysis of a closed fork-join queue model in an object-based storage system", Eighth International Symposium on Optical Storage and 2008 International Workshop on Information Data Storage, 2008.

[6] W. Yu, H. S. Oral, R. S. Canon etc., "Empirical Analysis of a Large-Scale Hierarchical Storage System", Euro-Par 2008, LNCS 5168, 2008, pp. 130-140.

[7] J. Piernas, J. Nieplocha, E. J. Felix, "Evaluation of Active Storage Strategies for the Lustre Parallel File System", SC'07, Nov. 2007.

[8] H. Zhang, W. Wu, X. Dong etc., "A High Availability Mechanism for Parallel File System", APPT 2005, LNCS 3756, 2005, pp. 194-203.

[9] Lawrence Livermore National Laboratory (LLNL), "I/O Guide for LC", Aug. 2007.

[10] Sun Microsystems, Inc., "Solving the HPC I/O Bottleneck: Sun™ Lustre™ Storage System", April 2009.

[11] Sun Microsystems, Inc. and Oak Ridge National Laboratory (ORNL), "Peta-Scale IO with the Lustre File System", Feb. 2008.

[12] DataDirect Networks, Inc., "Best Practices for Architecting a Lustre-based Storage Environment", May 2008.

[13] H. Z. Shan, J. Shalf, "Using IOR to Analyze the I/O performance for HPC Platforms', Cray User Group Conference 2007, Jun. 2007.

[14] W. Yu, S. Oral, J. Vetter etc., "Efficiency Evaluation of Cray XT Parallel IO Stack",Cray User Group Meeting (CUG 2007), 2007.

[15] W. Yu, J. Vetter, "ParColl: Partitioned Collective I/O on the Cray XT", ICPP 2008, 2008.

[16] J. Logan, P. Dickens, "Towards an Understanding of the Performance of MPI-IO in Lustre File Systems", 2008 IEEE International Conference on Cluster Computing, 2008.

[17] Sun Microsystems, Inc., "Lustre™ 1.6 Operations Manual", May 2009.

[18] G. R. Ganger, J. D. Strunk, A. J. Klosterman, "Self-* Storage: brick-based storage with automated administration", Technical Report CMU–CS–03–178, Carnegie Mellon University, August 2003.

[19] M. P. Mesnier, M. Wachs, R. R. Sambasivan, "Modeling the Relative Fitness of Storage", SIGMETRICS'07, 2007, pp.37-48.

[20] M. P. Mesnier, M. Wachs, R. R. Sambasivan etc. , "Relative Fitness Modeling", Communications of the ACM, Vol.52, No. 4, 2009, pp.91-96.

[21] J. L. Deng, "Grey group decision in grey rationale space", Journal of Grey System, Vol. 10, No. 3, 1998, pp.177-182.

[22] N. M. Xie, S. F. Liu, "Research on evaluations of several grey relational models adapt to grey relational axiom", Journal of Systems Engineering and Electronics, Vol. 20, No. 2, 2009, pp. 304-309.

[23] M. Lu, K. Wevers, "Grey System Theory and Applications: A Way Forward", Journal of Grey System, Vol. 10, No. 1, 2007, pp.47-54.

[24] J. L. Deng, "On judging the admissibility of grey modeling via class ratio", The Journal of Grey System, Vol. 5, No. 4, 1993, pp.249-252.

[25] H. Z. Shan, J. Shalf, "Using IOR to Analyze the I/O performance for HPC Platforms", Cray User Group Conference 2007, 2007.

[26] W. Yu, S. Oral, J. Vetter etc., "Efficiency Evaluation of Cray XT Parallel IO Stack", Cray User Group Meeting (CUG 2007), 2007.

[27] C. Ruemmler and J. Wilkes, "An introduction to disk drive modeling", IEEE Computer, Vol.27, No.3, March 1994,pp.17–28.

[28] E.Varki, A. Merchant, J. Xu etc., " Issues and challenges in the performance analysis of real disk arrays", IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 15, NO. 6, JUNE 2004. pp.559-574.

[29] A. Thomasian, C. Liu, "Comment on "Issues and Challenges in the Performance Analysis of Real Disk Arrays", IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 16, NO. 11, NOVEMBER 2005, pp.1103-1104.

[30] M. Uysal, G. A. Alvarez, and A. Merchant, "A modular, analytical throughput model for modern disk arrays", International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems, 2001, pp.183–192.

[31] T. Kelly, I. Cohen, M. Goldszmidt etc., "Inducing models of black-box storage arrays", Technical report HPL-2004-108, June 2004.

[32] M. Wang, K. Au, A. Ailamaki etc., "Storage device performance prediction with CART models", The 12th Annual International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS'04),2004, pp.588-595.

[33] E. Anderson, "Simple table-based modeling of storage devices", SSP Technical Report HPL–SSP–2001–4. HP Laboratories, July 2001.

[34] M. Wang, A. Ailamaki, and C. Faloutsos, "Capturing the spatio-temporal behavior of real traffic data", Performance Evaluation, Vol.49, No.4, 2002, pp.147–163.

[35] M. Wang, T. Madhyastha, N. H. Chan etc., "Data mining meets performance evaluation: fast algorithms for modeling bursty traffic"' International Conference on Data Engineering, 2002, pp. 507–516.

[36] L. Brieman, J. H. Friedman, R. A. Olshen, and C. J. Stone, "Classification and Regression Trees", Wadsworth, 1984.