DEIM Forum 2011 C10-5

Power-Proportionalityの概念に基づく ストレージシステムの実機実験による性能評価

レーヒェウハン[†]　引田　諭之[†]　横田　治夫[†]

† 〒152-8552　東京都目黒区大岡山 2–12–1,
東京工業大学大学院 情報理工学研究科 計算工学専攻
E-mail: †{hanhlh,hikida}@de.cs.titech.ac.jp, ††yokota@cs.titech.ac.jp

あらまし　近年，Power-proportionality という概念がストレージシステムの重要な評価指標の一つとして認識されている．この概念に基づき，Power-proportionality を満たすためのデータ配置や負荷分散などの研究が盛んに行われており，システムにいい影響を与えると評価されてるが，直接に同様な環境における評価実験はまだ行われていない．そこで，本稿では二つの代表的な手法を選択し実機実験による性能評価を行い，実験結果を報告する．

キーワード　電力性能比，ストレージシステム，省電力，実機実験，PARAID，RABBIT，HDFS

Performance Comparison of Power-Proportional Approaches in Storage Systems through Empirical Experiment

Hieu Hanh LE[†], Satoshi HIKIDA[†], and Haruo YOKOTA[†]

† Department of Computer Science, Graduate School of Information Science and Engineering,
Tokyo Institute of Technology, 2–12–1 Ookayama, Meguro, Tokyo, 152–8552 Japan
E-mail: †{hanhlh,hikida}@de.cs.titech.ac.jp, ††yokota@cs.titech.ac.jp

Abstract　Nowadays, the concept of power-proportionality has been recognized as one of important metrics in storage systems. Based on this idea, a number of proposals focusing on data placement, load balancing and so on have been proposed and evaluated to be successful in providing such characteristic to storage system. In this paper, the empirical experimental evaluation on performances of representative data placement proposals in this area is reported.

Key words　power-proportionality, storage system, power-aware, empirical experiment, PARAID, RABBIT, HDFS

1. Introduction

Recently, power savings in datacenters has gained a lot of interest due to the increase in power delivery. As indicated in [1], the electricity used in all datacenters in the US in 2006 has more than doubled since 2000. Furthermore, of all end-use components, such as site infrastructure, network equipment, servers, etc., disk storages have been recognized as the greatest growth at a compound annual growth rate (20%). This trend will continue as data intensive applications demand fast and reliable to online data resources.

Until now, almost all of disk-based power-aware storage systems have tried to save energy by leveraging the difference in power consumption among different operation modes of disks [2], [3]. In these approaches, the system is able to decide to spin down disks from their usual high energy consumption mode (active/spin-up) into a low energy mode (inactive/standby) after observing an idle period when no request comes. However, this technique is implemented at the trade off performance (response time) penalty. Disks can only serve requests while they are in active mode. Once the request arrives into the inactive disk, it should be wait for a certain penalty time for the disk to be waked up before being served.

As the results, rather than simple power saving, achieving power-proportionality in datacenters has been gained more and more attention and considered as an important design factor [4]. The basic concept in power-proportionality is that computing equipment, such as storages, servers, networks should consume energy in proportion to the amount of work performed. Specially for storage, power-proportionality can be achieved through data placement method by controlling the total number of active disks (or nodes) in the system. Taking this concept into consideration, a number of data placement policies have been proposed in order to provide this metric to disk-based storage systems [5], [6], [7], [8]. These approaches are common in the way of managing energy in a group of disks instead of controlling a single disk individually. The technique that supports these approaches is data replication, which is currently very useful in datacenters for good availability, fault tolerance and load balancing. In power-aware systems, replication also benefits the possibility of selecting a replica in current active disks rather than choosing another replica stored in a disk which is in sleep mode. In order to achieve this idea, at first the systems divide all their storage disks into certain small groups. And each group, which contains different number of disks, is ensured to store all necessary data for guaranteeing the availability of systems. By controlling power management in the granularity of group, those approaches are considered to be success in providing the power-proportional metric to storage systems.

Although a number of proposals, introduced like above, are evaluated to be success at certain circumstances, however they have been still not compared with each other in the same environment yet. Hence, in order to identify the characteristics of each proposal, the performance comparison of so called methods is needed. In this paper, we decide to choose two representative proposals, i.e. PARAID [5] and RABBIT [8] and perform empirical experiment on actual machines to compare their performance in the context of power-proportionality. While PARAID inspires many other researches by its idea of controlling system's power over small groups, RABBIT is the novel work adapted to implement on HDFS which is very popular in distributed computing area.

Furthermore, taking availability and scalability into consideration, we perform the experiments over HDFS (Hadoop Distributed File System) [9], a popular distributed file system for high scalability and availability. In our knowledge, in the current time, the idea of PARAID is just implemented in simulation or under the small scale of several RAID disks [5].

From the experiment results, it is found out that both two methods can provide power-proportionality to system in distributed environment where distributed I/O is needed. Moreover, because of gaining better-balanced data distribution, RABBIT is measured to achieve better performance than PARAID in our experiment.

The paper is organized as follow. Section 2 gives a detail description of PARAID, RABBIT, and a brief review of other approaches in power-proportional storage systems. Section 3 is planned to describe and report the experiment results on actual machines. The conclusion and future works will be summarized in Section 4.

2. Power-Proportional Storage System Approaches

In this section, the data placement and power management to implement power-aware storage systems used in PARAID and RABBIT are described. And then, some other proposals in this approach are also introduced briefly.

Although not like RABBIT, PARAID was originally designed inside a RAID unit, the idea can be expanded to distributed environment that contains a large number of nodes connected through network. In this context, a node is defined as an array of disks managed together with respect to energy control. Thus, a node is a collection of disks and there is no disk sharing between nodes. In this part, we describe the modified skewed data placement in PARAID in order to apply in distributed environment.

Both of these proposals are based from the idea of dividing the total number of disks in the system into small groups. Consequently, the system then contains a certain number of gears that include number of groups. A low gear with small number of powered disks is supposed to consume low power and vice versa.

Given a dataset D with total B blocks, a total number of nodes N are divided into G groups. Each group contains a different number of nodes. In detail, each node is symbolized as $n_{(g,i)}$, where g $(1 \leqq g \leqq G)$, i $(1 \leqq i \leqq N)$ indicate i-th of node at g-th group. For example, nodes $n_{(1,1)}$, $n_{(1,2)}$ belong to Group 1, while nodes $n_{(2,3)}$, $n_{(2,4)}$ and $n_{(2,5)}$ belong to Group 2. Note that, to normally operation, the system need at least all disks in Group 1 are powered.

2.1 PARAID

PARAID is the first work to introduce the concept of gear-shifting based on load of system within a RAID unit. It utilized the idea of skewed striping pattern to adapt to the system load by varying the number of powered disks.

2.1.1 Data Placement

PARAID takes advantage of unused storage in disks to replicate and stripe data blocks in a skewed fashion. Then, disks could be organized into a number of sets of RAIDs. Each set defining as gear, contains a different number of disks and can serve all requests via either its primary blocks or replicated block.

In PARAID, at first, all data D of B blocks are allocated evenly to all nodes. We denote by $B_i(\frac{1}{m})$ an $\frac{1}{m}$ fraction of B_i. After replication, each node will hold a certain replica in addition to its original data as follows: (a) Each node in Group 1 nodes gets an equal fraction of the replicated

Table 1 PARAID data placement

	Group 1 nodes		Group 2 nodes				
Node	$n_{(1,1)}$	$n_{(1,2)}$	$n_{(2,3)}$	$n_{(2,4)}$	$n_{(2,5)}$	$n_{(2,6)}$	$n_{(2,7)}$
Original	B_1	B_2	B_3	B_4	B_5	B_6	B_7
Replica	$B_3(\frac{1}{2})$	$B_3(\frac{1}{2})$					
	$B_4(\frac{1}{2})$	$B_4(\frac{1}{2})$	$B_4(\frac{1}{3})$				
	$B_5(\frac{1}{2})$	$B_5(\frac{1}{2})$	$B_5(\frac{1}{4})$	$B_5(\frac{1}{4})$			
	$B_6(\frac{1}{2})$	$B_6(\frac{1}{2})$	$B_6(\frac{1}{5})$	$B_6(\frac{1}{5})$	$B_6(\frac{1}{5})$		
	$B_7(\frac{1}{2})$	$B_7(\frac{1}{2})$	$B_7(\frac{1}{6})$	$B_7(\frac{1}{6})$	$B_7(\frac{1}{6})$	$B_7(\frac{1}{6})$	

data from each node of other groups; (b) Remaining nodes keep replicas of original data from specific other non-Group 1 nodes in skewed way. Specifically, the original data B_i of a non-Group 1 node $n_{(g,i)}(g > 2)$ are replicated equally to other non-Group 1 nodes. This is done by selecting $\frac{1}{i-1}$ blocks of B_i for each node $n_{(g,j)}$ $(j < i)$. Table 1 indicates a simple example of placing data through a storage cluster with 7 nodes and two groups. Here, the system is able to operate in two gears. Gear 1 contains only active disks in Group 1 while Gear 2 needs all disks in both Group 1 and Group 2 to be spin-up.

Through the above techniques, PARAID is considered to be able to provide the power-proportional characteristic that performance is proportional with the power consumption of the system.

2.1.2 Power Management

In PARAID, there exists a Monitor to decide when a gear shift is needed based on the current system load.

a) Up-shifts:

In order to decide when to up-shift, the Monitor must know whether the current gear has reached a predetermined utilization threshold, in the meaning of busy RAID period (milliseconds) within a certain time window. An active disk is marked busy from the point when a request enters the disk queue to when the completion callback function is invoked. Since multiple requests can overlap, a request that be completed in t milliseconds then the device serve that request is mark prior t milliseconds busy.

The Monitor keeps a moving average of utilization $0 \leqq U \leqq 1$ and a moving standard deviation S for each gear to track the system load. If the utilization plus its standard deviation is greater than the threshold $0 \leqq T \leqq 1$, an up-shift is performed.

$$U + S > T \tag{1}$$

b) Down-shifts:

To decide when to down-shift, the utilization of the lower gear $0 \leqq U' \leqq 1$ needs to be computed, with also the associated moving standard deviation S'. If their sum is below the threshold T, the lower gear can now handle the current load.

$$U' + S' < T \tag{2}$$

Table 2 RABBIT data placement

	Group 1 nodes		Group 2 nodes				
Node	$n_{(1,1)}$	$n_{(1,2)}$	$n_{(2,3)}$	$n_{(2,4)}$	$n_{(2,5)}$	$n_{(2,6)}$	$n_{(2,7)}$
Data	$\frac{B}{2}$	$\frac{B}{2}$	$\frac{B}{3}$	$\frac{B}{4}$	$\frac{B}{5}$	$\frac{B}{6}$	$\frac{B}{7}$

2.2 RABBIT

RABBIT is a power-proportional distributed file system (PPDFS) that uses a cluster-based storage data placement to provide power-proportional factor.

2.2.1 Data Placement

Supposedly r replicas of B blocks of dataset D are desired to be stored to n nodes with G gears. At first, one replica of all B blocks are evenly stored in first primary p nodes at Group 1 (also called primary group). Consequently, each node in Group 1 contains $\frac{B}{p}$ blocks. The remaining $(r-1)$ replicas are distributed to $(N-p)$ nodes in the way that the node $n_{(g,i)}$, where $g > 2$ and $p < i <= N$, stores $\frac{B}{i}$ blocks. Here, in the constrain of keeping number of replica r small with fixed number of nodes RABBIT can guarantee that the number of blocks stored by i-th node must not be less than $\frac{B}{N}$ for all $i \leqq N$ when N nodes are active. Obeying this constrain makes it possible for the load to be shared equally among active nodes. Thus, the performance of the system is suggested to be linear with the number of powered nodes, i.e. implicitly corresponds to the power consumption of the system.

Table 2 shows an example of data placement in RABBIT for a 7-node cluster with two primary nodes and 2-replica. It is well recognized that 5 nodes in Group 2 store are enough for storing the necessary data of second replication. Like PARAID, the system in RABBIT can operate in two gears. The first gear includes active disks in Group 1 and the second gear needs the disks in Group 2 joining the systems.

2.2.2 Power Management

In [8], the authors have not proposed the mechanism of automatic control the gear shifting yet. However, the idea in PARAID, i.e. based on utilization of system is considerable.

2.3 Others

Like PARAID, GRAID [6] is a green storage architecture aiming for improving energy efficiency and reliability within RAID unit. In this proposal, the data mirroring redundancy of RAID10 is extended by incorporating a dedicated log disk. The function of this log disk is to store all updates since last mirror-disk update. Using these information for log disk, the system only needs to update the mirroring disks only peri-

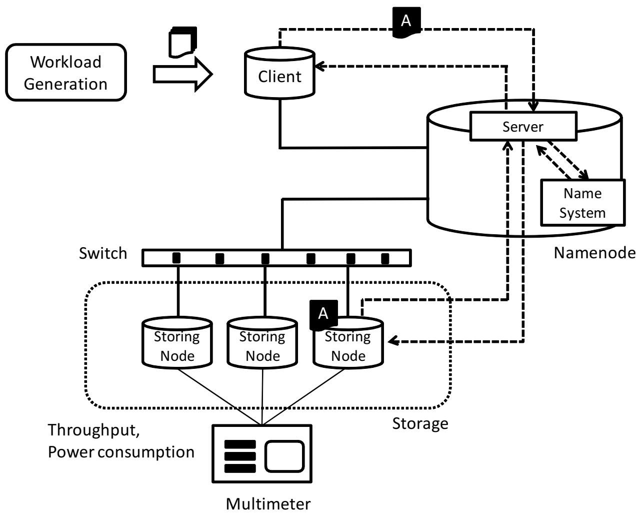

Figure 1　An overview of system framework

odically, thus being able to spin down all the mirroring disks to lower power mode most of the time to save energy.

Also designed as a power-proportional, distributed file system as RABBIT, SIERRA [7] is a replicated object store that allows storage servers to be put into low power states when load is low. This allows servers hosting inactive replicas to be powered down. For example, in three-way replicated system like in normal HDFS or Google File System, SIERRA allows up to $\frac{2}{3}$ of the storage servers to be in standby. Comparing with RABBIT, SIERRA's data placement is simpler as all storage servers keep the same amount of data.

3. Experiments

The purpose of the experiments is to compare the performance of PARAID and RABBIT in the context of power-proportionality. Because RABBIT is still not implemented write function yet so in the cover of this paper, we focused on read performance only. Furthermore, as PARAID was originally designed within RAID unit, in this experiment, in order to be comparable with RABBIT, the idea of data placement in PARAID was extended to be implemented on HDFS like in RABBIT.

At first, two methods are compared in the context of power-proportional by using a short synthetic workload started from one client node. And then, an comparative evaluation's result of these two methods when applying benchmark that utilizes distributed I/O over storage system is discussed. In this part, the module of automatic gear shift discussed at Sec. 2. 1. 2 is still not implemented yet and left as future work. Memory at each node is deleted after each running in this experiments.

3. 1 Synthetic Workload Experiment

3. 1. 1 Experiment Environment

An overview of the experiment framework is shown in Fig. 1. There are four main elements in our framework.

Firstly, in storage, we use a number of Storing nodes which play a role of Datanode as in HDFS. Each Storing nodes is an autonomous disk designed for low power consumption.

Next, in order to implement data placement and to manage information about data location such as which Storing node is containing what data, a Namenode is used. At this Namenode, the source codes relating to data placement of normal HDFS are touched to make it available to implement the layout policy of RABBIT and PARAID. Furthermore, storing data is mounted to this Namenode based on the Filesystem in Userspace project FUSE [10]. Over this system, a web server service is set up to serve requests from clients.

To generate requiring data requests to server, one Client node is used. At this node, some other tasks such as logging to extract each request's response time are also implemented.

Finally, to measure the power of Storing nodes, we used Hioki 3334 digital multimeter. This multimeter is made connect with another Windows XP node for saving measurement results.

The interconnect between Client node and Namenode is Extreme Network Summit 16101 Gigabit Ethernet switch. Client and Namenode, the switch and Storing node is connected directly through CAT-5e cable.

In this system, at first, the Namenode needs to write whole dataset into the storages and after that, the server is ready to serve requests. As presented in Fig. 1, supposing that there is a request requiring data A is started from Client. When it arrives at server, the server service makes a request to Namespace a service of HDFS to get the information about which Storing nodes is containing data A. In next step, the server directly connects to the responsible Storing node to download data A, and send back to Client.

The specification of Client, Namenode and Storing nodes are summarized in Tab. 3.

3. 1. 2 Experiment Results

In this part, the effective of each system relating to the ratio between each request's throughput and system's power consumption is presented. At first, the power consumption of one Storing node is measured. And then an experiments on system's performance and the power consuming in that serving period with given synthetic workload is discussed.

a) Power consumption of one Storing node

To investigate the system's power performance, the information about power consumption of a single Storing node is needed. Table 4 summaries the result for the Storing node using in this experiment.

As noted in previous section, the model Storing node using in this experiments are designed for low power consumption so, comparing to other normal system, its power consump-

Table 3 Nodes specification

	Client	Namenode	Storing node
CPU	Intel Core 2 Quad 2.40GHz	Intel Pentium 4 3.00GHz	Transmeta Efficeon TM8600 1.0GHz
Memory	SDRAM 2048MB	SDRAM 1024MB	DRAM 512MB
HDD	SATA 320GB	SATA 250GB	IDE 100GB
Network Interface Card	1000Mb/s	1000Mb/s	100Mb/s
OS	Linux 2.6.18	Linux 2.6.18	Linux 2.6.18
Java	JDK-1.6.0	JDK-1.6.0	JDK-1.6.0
HDFS	-	0.20.2	0.20.2

Table 4 Power consumption for a single Storing nodes

Node status	Power consumption [Watt]
Normal	10.35
Standby (hard disk's spindle motor is off)	9.45
Sleep (hard disk is shut down)	9.30
Hibernation (only hard disk is on)	3.36

Table 5 Workload Parameters

Workload Parameters	Value
Experiment duration	30 minutes
Read:Write	1:0
File size	1000 [MB]
Number of files	1000
Total amount of data	1 [GB]
Data access distribution	Zipf
Access time distribution	Poisson
Zipf parameter	1.2
Average arrival period (request/s) λ	5
Number of requests	9000 ($\lambda \times 60 \times 30$)

tion was quite small. Also, from this result, the hard disk in our model is discovered to consume just about 1[W] in practice. As the result, it is very difficult to evaluate PARAID and RABBIT in the context of power-proportional if only powering off the hard disk in low gear, i.e. small number of active Storing nodes. So, we decided to hibernate all inactive Storing nodes in low gear because at that status they use up to 3.4[W] (30% of total node).

b) Power-proportional characteristic of whole system

The parameters of synthetic workload using in this experiment are shown in Tab. 5. For each RABBIT and PARAID, we run two kinds of scenario. The first one is running the system with total 7 Storing nodes. The second scenario is running with 2 active Storing nodes while 5 others were manually set to hibernate. Each scenario was run twice and we measure the average power consumption also extract the average throughput from log files at Namenode.

Figures 2 show a relationship between average throughput and average power consumption results at storage system for

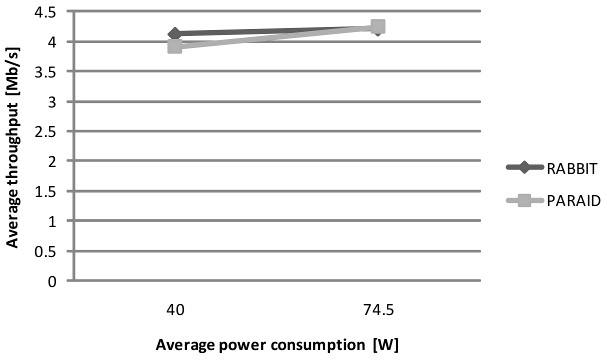

Figure 2 System performance

cases of 2, 7 active Storing nodes.

X-axis of this figure shows the different power consumption between two scenarios, i.e, running 2 and 7 active Storing nodes. The latter consumed about 35[W] more than the former. That is because there were 5 Storing nodes is hibernated, and each nodes at hibernate status saved approximately 7[W], as in Tab. 4.

Also it is seen from y-axis that the throughput of system in both methods (RABBIT and PARAID) were not much improved comparing to the case only 2 active Storing nodes were active (8% at maximum). However, the power consumption was increased about 86%. Consequently, it can be concluded that in this scenario both 2 methods were failed to provide power-proportionality.

Overall, it is hard to identify the difference between two methods. The considerable main reason is due to there is no big difference in performance between two methods when the setting configuration of system changes from low mode to high mode (2 to 7 active Storing nodes). This is because in this scenario read requests were initialized by a single client, and the load to system was not high enough to fully take advantage of parallel processing which is one of important feature in distributed environment. In next part, the performance of two methods is evaluated by using a benchmark that uses HDFS to perform distributed I/O over storage.

3. 2 Distributed I/O Experiment

In this part, the comparative evaluation result of two methods in distributed I/O manner is reported.

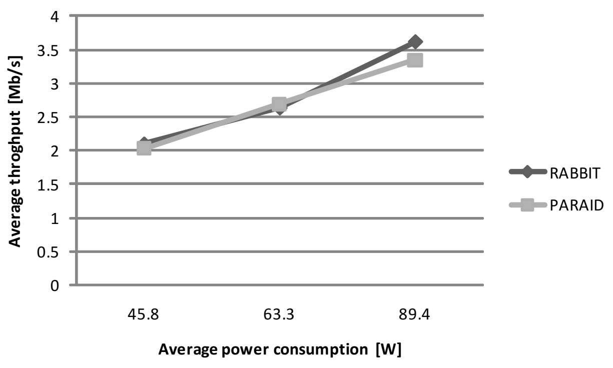

Figure 3 Performance vs. Power Consumption

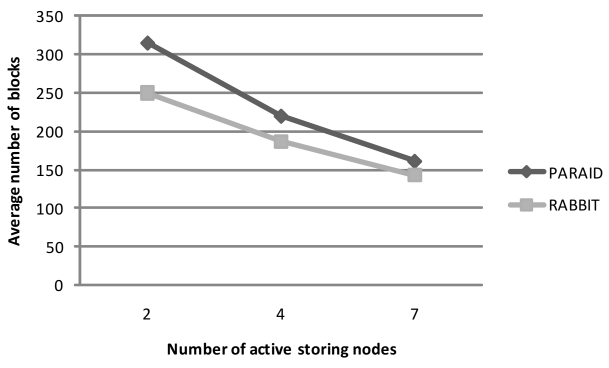

Figure 4 Average number of blocks storing at each Storing nodes

3. 2. 1 Experiment Environments

The performance of two methods is tested using TestDFSIO, a bechmark requiring distributed I/O over HDFS, with 2 GB dataset containing 20 files and each file's size was 100 MB. Here, the block size of HDFS was changed to 4 MB for guaranteeing one file's data was allocated over all Storing nodes in storage. During the experiment, the corresponding power consumption of storage partition were measured by using multimeter Hioki 3340. In this experiment, the same type of Namenode and Storing nodes used in previous experiment were used and the number of active Storing nodes was fixed at 2, 4 and 7 nodes. Inactive nodes were set to hibernation.

3. 2. 2 Experiment Results

The relationship between system's throughput and power consumption is shown in Fig. 3. In this figure, the x-axis shows the average power consumption when the number of active nodes was set to 2, 4 and 7 accordingly. The y-axis shows the average throughput in [MB/s] taken from benchmark TestDFSIO.

From this figure, it can be seen that the performance of system was quite linearly increased when changing the configuration of storage system by raising the number of active nodes. In both methods, the average throughput was improved at least 1.65 times. This result shows the effective of two data placement policies in order to provide power-proportionality in distributed environment.

Comparing two methods, it is observed that RABBIT slightly overcame PARAID at most 8% when all Storing nodes were active. Further investigation was performed to find out the reason behind this result.

Figures 4 and 5 show the average storing blocks at each Storing nodes and the standard deviation when the number of active Storing nodes was changed. From this result, in this experiment, it is seen that RABBIT obtained better distribution of total blocks to active Storing nodes than PARAID due to smaller values at both cases. It shows the importance of well balancing data distribution over Storing nodes in data placement in order to provide high performance.

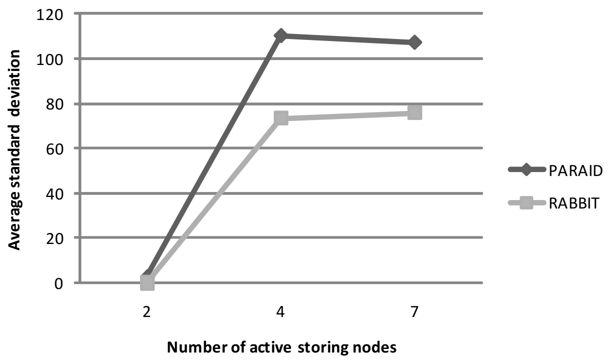

Figure 5 Average standard deviation

4. Conclusions and Future Work

The power-proportionality has been found out to be an important design metric in power-aware storage systems. Currently, almost all of researches focused on dividing total number of disks to small groups by leveraging replication techniques. By managing energy in granularity of small group, the systems can achieve power-proportionality.

This paper described two most representative efforts focusing on data placement and energy control. Here in this experiments, the idea of PARAID which was originally design for RAID unit has been extended to be applicable in distributed environment like HDFS for the first time. In order to comparatively evaluate these two methods in the context of power-proportionality, the experiment on read performance was performed. From the experiment results, for application requiring distributed I/O over storage system, both methods were seen to succeed in providing power-proportional metric as average throughput raised when approving higher power consumption. Because of smaller deviation in distributing data over Storing node, RABBIT was confirmed to have higher, at most 8%, throughput than PARAID.

In the future, we would like to perform other comparative evaluation between these two methods, such as write performance with larger data set. Furthermore, note that in our experiment, the model of Storing node was design for low energy consumption, as the result, in the future work, the other models closed to models using in normal datacenters.

Acknowledgements

This work is partly supported by Grants-in-Aid for Scientific Research from Japan Science and Technology Agency (A) (#22240005) and MEXT (#21013017).

References

[1] U.S. Environment Protection Agency ENERGY STAR Program, "Report to Congress on Server and Data Center Energy Efficiency," `http://www.energystar.gov/ia/partners/prod_development/downloads/EPA_Datacenter_Report_Congress_Final1.pdf`, 2007.

[2] D. Colarelli, and D. Grunwald, "Massive Arrays of Idle Disks for Storage Archives," in the Proceeding of the 2002 ACM/IEEE conference on Supercomputing, pp. 1–11, Los Alamitos, CA, USA, 2002, IEEE Computer Society Press.

[3] E. Pinheiro, and R. Bianchini, "Energy Conservation Techniques for Disk Array-Based Servers," in the Proceedings of the 18th Annual International Conference on Supercomputing, pp. 68–78, New York, NY, USA, 2004, ACM.

[4] B.L. Andre, and H. Urs, "The Case for Energy-Proportional Computing," Computer, vol. 40, no. 12, pp. 33–37, 2007.

[5] W. Charles, O. Mathew, Q. Jin, W.A.I. Andy, R. Peter, and K. Geoff, "PARAID: A Gear-Shifting Power-Aware RAID," Transaction on Storage, vol. 3, October 2007.

[6] B. Mao, D. Feng, H. Jiang, S. Wu, J. Chen, and L. Zeng, "GRAID: A Green RAID Storage Architecture with Improved Energy Efficiency and Reliability," in the Proceeding of IEEE International Symposium on Modeling, Analysis and Simulation of Computers and Telecommunication Systems, 2008. MASCOTS 2008., pp. 1–8, September 2008.

[7] E. Thereska, A. Donnelly, and D. Narayanan, "SIERRA: a Power-Proportional, Distributed Storage System," , 2009.

[8] A. Hrishikesh, C. James, G. Varun, G. Gregory R., K. Michael A., and S. Karsten, "Robust and Flexible Power-Proportional Storage," in the Proceeding of the 1st ACM Symposium on Cloud Computing, pp. 217–228, New York, NY, USA, 2010, ACM.

[9] "Hadoop," `http://hadoop.apache.org`.

[10] "Fuse: File system in userspace," `http://fuse.sourceforge.net`.