論文 / 著書情報
Article / Book Information

Title	Data layout management for energy-saving key-value storage using a write off-loading technique
Author	Masaki KAN, Dai Kobayashi, Haruo YOKOTA
Journal/Book name	2012 IEEE 4th International Conference on Cloud Computing Technology and Science, , , pp. 75-81
Issue date	2012, 12
DOI	http://dx.doi.org/10.1109/CloudCom.2012.6427514
URL	http://www.ieee.org/index.html
Copyright	(c)2012 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other users, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works for resale or redistribution to servers or lists, or reuse of any copyrighted components of this work in other works.
Note	このファイルは著者（最終）版です。 This file is author (final) version.

Data layout management for energy-saving key-value storage using a write off-loading technique

Masaki KAN*†, Dai KOBAYASHI*, Haruo YOKOTA†,
* *Cloud System Research Laboratories, NEC Corportaion*
1753, Shimonumabe, Nakahara, Kawasaki, Kanagawa, 211-8666, Japan
Email: kan@bq.jp.nec.com, daik@ay.jp.nec.com
† *Department of Computer Science, Tokyo Institute of Technology*
2-12-1 Ookayama, Meguro-ku, Tokyo 152-8552, Japan
Email: kan@de.cs.titech.ac.jp, yokota@cs.titech.ac.jp

***Abstract*—The objective of our research has been to achieve energy-saving key-value store system. We introduce architecture of our distributed key value store and control framework for energy saving. To reduce power consumption, the control framework should reduce the time required before node power-down. Since this time depends heavily on the amount of data migration, we employ an approach which reduce that amount. We describe and evaluate three control strategies that can be executed on this control framework, each consisting of a data allocation algorithm and data control layout algorithm for use when nodes are to be powered down. With a prototype system developed for evaluation purposes, the time required before node power down is reduced by use of our proposed data allocation algorithm, by means of its taking into account the possible presence of Write off-loading nodes.**

***Keywords*-Cluster computing, Power-propotionality, Data layout, Key value store**

I. Introduction

Power consumption in data center is major problem. Research has shown that the energy used worldwide by data centers is 1.1 - 1.5% of total electricity use [1]. In Japan, in the aftermath of the 2011 Tohoku earthquake and tsunami saving electricity at data centers has become even more important than before. In recent years, much research has been focused on reducing energy consumption. Because the cloud computing paradigm increases importance of large-scale distributed storage,we focus on a large-scale distributed storage system for use data centers.

Distributed Key-Value Store systems, or KVSs, are scalable, high performance data storage systems that offer a type of large-scale distributed storage. They consist of multiple storage nodes connected via networks. Each stored data object, referred to as a "value" is associated with a unique identifier called a key. The stored objects are replicated for enhanced availability and system reliability.

Energy consumption in distributed storage systems can be reduced by making substantial subsets of nodes into low-power modes. Disk-array based systems can decrease power consumption by reducing the spin-rate of some of their disk drives [2]. In cluster-based systems, powers of disk-drives power consumption is not a dominant factor overall.In addition, in these systems, more than half of the power consumption of any individual computer will be independent of its workload [3]. Thus, rather than powering down individual components within nodes in response to variations in workload, a better balance between active-time and non-active time power consumption can be achieved by using control which powers down individual nodes as a whole.

A. Motivation

Distributed storage system consists of multiple storage nodes connected via networks. The objects are multiply replicated (typically three) and stored in some nodes.

Even when some nodes was powered down to reduce energy consumption, the storage system must maintain the data store availability and reliability. In other word, the storage system should be able to respond to read/write access for all objects properly.

Write-offloading is one of powerful approach to keep availability, reliability and power proportionality of data store systems. With write offloading, when write access requests are made to multiple sleeping nodes each containing the same object, each of the requests is forwarded to an available node, in which the relevant information contained in the request information contained in the request is stored in a temporary update log. The update logs are then employed after the sleep nodes wake-up. Originally the write offloading was proposed by Narayanan et al.[4] for power management of disk array systems .

To enhance power reduction, the time required before node power down should be reduced. This time depends on the amount of data migration, the time required to calculate which objects should be selected for migration, and the time required to put nodes into a sleep mode. In this paper, we focus on the amount of data migration because the time in our architecture, this element accounts for an overwhelming percentage of the time required to achieve node power-down.

The requirement for keeping the amount of data migration as low as possible is a difficult one to meet. The consistent hashing used by Amazon Dynamo, for example is a scalable approach, but it is sufficient for coping with the huge amount of data that would have to the migrated in an attempt to scale storage nodes dynamically. While it is known that Write-offloading can be used to measurably reduce the amount of data migration, the potential for combining it with data layout management has yet to be thoroughly determined.

B. Data layout management of distributed data store

Our focus is developing a power-proportional Distributed Key-Value Store system and an energy-saving control framework for it.

In this paper, we propose control strategies which are consists of data allocation algorithm and data-control layout algorithm for use in reducing the amount of data migration needed for node power-down.

We have developed a prototype distributed key value store (DKVS) system, results of an evaluation of it show our proposed strategy to be capable of reducing by roughly 90% the time required to shift a low power state.

The three primary contributions of this paper are:

- We consider problems involved in needed for node. We propose a new data allocation algorithm that takes into consideration the potential presence of Write-offloading nodes.
- We show the useful of Write-offloading mechanisms by simulation and physical experiments.
- The introduction and evaluation of three control strategies that enables power-saving distributed key value store.

The reminder of this paper is organized as follows. Section II describes in detail of our power-proportional KVS approach and issue of data layout control needed for node power-down. Section III describes the architecture of our proposed control framework. Section IV describes data allocation and control strategies. Section V evaluates the strategies and the effect of power-saving in our prototype system. Section VI discusses related research, and Section VII summarize our work.

II. Issues of energy-saving data store

Distributed data store systems are equipped with only basic functions for energy-saving operations, and data location/control and node power up/down management methods for energy-saving distributed data storage have yet to be sufficiently developed. In this section, we discuss requirements of data location/control management methods, and issues of system operation.

Energy saving control methods for distributed data storage must be able both to reduce power consumption as much as possible while minimizing the adverse affects of that reduction on access performance.

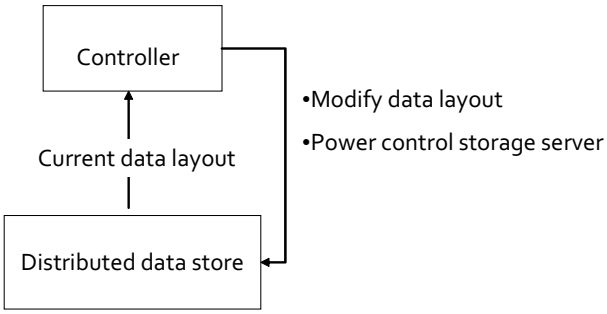

Figure 1. System Architecture

To reduce power consumption, time required before node power-down control has to be reduced. This time depends on the amount of data migration, the time required to calculate which objects should be selected for migration, and the time required to put nodes into a sleep mode. In this paper, we focus on the amount of data migration because the time in our architecture, this element accounts for an overwhelming percentage of the time required to achieve node power-down.

When multiple nodes have been powered down, the data distribution much be such that any data object will have at least one replica in an active node, and when update process have been performed by means of Write-offloading, there must be redundancy in the storage of information regarding the updates. For this purpose, data migration must be completed before power-down are commenced. Further, migration will an influence on storage access performance because it requires the use of the network bandwidth. Migration processes require the use of resources of computers that operate meta servers, and accesses object replicas. While suppressing the rate of data movement might seem to be an option, that would significantly degrade power-reduction.

Energy saving control should not cause additional performance bottlenecks. This means that the distribution of data in the state following the power-down of multiple nodes should be such that the number of access requests to active nodes should be relatively even across the range of those nodes, which implies that the number of objects contained in those nodes should also be relatively even across the range. This is made more difficult by the fact that the load resulting from update information requests will be extremely heavy on nodes which have received such information, resulting in overall skew that can easily create a performance bottleneck.

III. System Architecture Design

In this section, we introduce a control system architecture for energy-saving distributed data storage. Figure 1 shows the overall architecture of our system.

This architecture creates a framework for adjusting node power-levels, maintaining effective data distribution, and adjusting the number of active nodes to best accord with load requirements. Our controller has been designed and

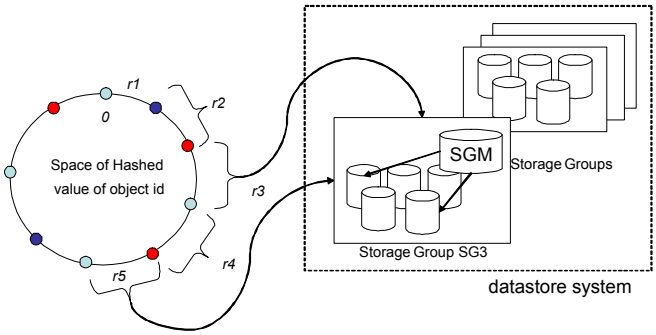

Figure 2. Hierarchical data location management on the DKVS

developed for our prototype DKVS system. Because we designed it for use with general-purpose API, however, it can easily be modified for application to non-DKVS systems, such as an open-source key-value system.

A. DKVS: Distributed Key-Value Store

In this section, we give an overview of an energy-saving DKVS system in which individual nodes can be powered down in their entirety.

Our DKVS system can power down storage nodes. It stores an *object* linked with a unique key in one system. This object consists of multiple properties. The application programmer describes multiple field records by using multiple properties. In addition, the DKVS has consistency only at the object level. The DKVS has multiple replicas on multiple individual servers and deals with access by using a primary-backup consistency protocol. The objects have multiple versions' data in several servers.

The DKVS's client program can call nodes that have a target object via the client's library. This library has an asynchronous cache of the system nodes' list and the objects' location. This library specifies the nodes that have objects by accessing meta-servers (discussed below) or this cache.

We think that there are two requirements for achieving energy saving operation: (1) balance adaptive data allocation with system scalability, and (2) avoid overmuch nodes' power-up by write-offloading. Consequently, we make the DKVS have the following two functions:

- Hierarchical Data Location Management on the DKVS (section III-A1)
- Node Power Down (migrate to ACPI S3/S4) and write-offloading (section III-A2)

1) Hierarchical Data Location Management on the DKVS: To keep scalability, many distributed key-value store systems use consistent hashing for the data allocation scheme. However, these systems lack the flexibility in data allocation needed to reduce power consumption. To manage data allocation, the single meta-data server architecture has risks such as performance bottleneck and single point of failure. Therefore, a large-scale system prefers consistent hashing to specify data allocation nodes on several client servers.

To stop some nodes of systems and provide the storage's function continuously, systems should put at least one replica of each object into an available node for processing all read requests. If nodes of the access object are powered down, the system should boot one of the nodes. It takes a very long time to boot a node. Distributed key-value store systems with a straightforward consistent hashing mechanism decide on an object's allocation on the basis of the key's hash values. Therefore, the number of the candidates of nodes that can be shifted to power-down mode becomes lower. The straightforward solution fixes the functions of each node, such as some nodes for primary objects and some nodes for backup objects. Because this approach is not effective at the point of load distribution, we think that the data allocation mechanism should cope with both scalability and flexibility to reduce power consumption.

Figure 2 shows a simple overview of hierarchical data location management on the DKVS used to cope with both scalability and flexibility. The DKVS divides storage nodes into multiple storage groups (**SG**). One node of each storage node is appointed as the SG's master node (**SGM**) in each SG. In addition, the RootMaster node (**RM**) checks the state of SGMs for fault detection, handles some processes to restore faulty SGMs, and does some other minor work. The DKVS divides the hash value spaces of object keys and assigns these spaces to SGs. Because the SGM manages data allocation within a SG, it can control data allocation freely (not depend on consistent hashing). One SG can undertake multiple hash value spaces. This hierarchical management can power down more servers to allocate an object's replica suitability when the system's load is low.

2) Node power down and write off-loading: To power down a node, ACPI power modes are available. Unlike disk-based storage, in-memory storage loses stored data during power down because DRAM is volatile. ACPI power mode such as S4, called "hibernation," writes a memory image onto non-volatile media such as hard disk drives and shutdown nodes.

Even when some nodes are power down, systems must process access to sleep nodes for reliability and availability. Putting at least one replica of each object into an available node can process all read requests. However, write requests must be processed on the necessary number of nodes so as not to lose updating when some node failures occur.

One of useful method for keeping the redundancy of updating data is write offloading, which allows the updating of objects to be redirected to other available nodes temporarily for a few minutes or up to a few hours. These objects are reclaimed slowly in the background after the target node wakes up.

It is easy to implement write off-loading at in-memory

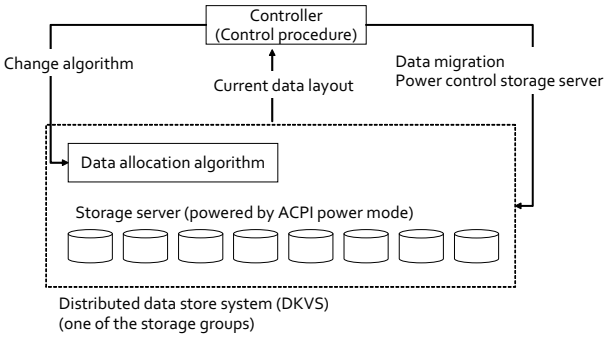

Figure 3. System architecture for the controller

KVS. The write off-loading target is decided before node power-down and written with data location information or decided statically. The client library has the role of transferring access to write off-loading nodes on the basis of the updated data location information when some nodes are powered down. The write off-loading node records the accesses into log objects for the power down node. When the node wake ups, the write off-loading node applies the update log before the node re-entry to the system.

The write off-loading mechanism in the DKVS allocates a write off-loading target node every data node. A mechanism to allocate a write off-loading target node every object is one other approach. However, this approach needs to search and get write off-loading logs from all nodes during the write off-loading log apply process that occurs when nodes wake up. If sleeping nodes have logs to apply, these nodes must be woken-up. In this way, this approach needs a more complicated procedure, which causes more electrical power to be wasted (by node wake up/power down).

The DKVS's write off-loading mechanism for allocating a write off-loading target node every data node needs to check only one write off-loading target node per sleeping node. Moreover, ensuring a write order to a write off-loading log is more simple. In addition, write off-loading target nodes should not shift to a powered down state for simplicity. The write off-loading log apply process causes write-offloading target nodes to be woken up, in a similar way.

B. Controller

Figure 3 shows the system architecture of the controller for a distributed data store. In this figure, the distributed data store system is one of the storage groups in the DKVS. The controller acquires the current data layout from a data store system, and requests the DKVS's data migration calls and node power control calls. The storage servers support ACPI power modes, are changed by requests from the controller. The client of the data store system allocates new objects depending on data allocation algorithms, which are specified by the data store system. In this system, these algorithms are implemented on the client library and SGM.

```
1: plan ← calcControlPlan()
2: for all migration objects o of plan do
3:    call migration o
4: end for
5: for all power control servers S of plan do
6:    call power control request to S
7: end for
8:
9: calcControlPlan() {calculate new data layout plan}
10: estimates the number of nodes for the current load.
11: select S (power control target nodes)
12: plan ← S
13: extract o (to move objects)
14: for all migration objects o do
15:    select migration target node and input target node
       information and o to plan
16: end for
17: return controlPlan
```

Figure 4. Control procedure

1) control procedure: Figure 4 shows the control procedures in the controller. The performance of the distributed data store and the effect of energy saving depends on the concrete algorithms in this procedure and data allocation algorithms. These algorithms are described next. In this paper, we call the combination of calculating the control plan and the data allocation algorithm *strategy*. This strategy can be replaced by developers. At the line 11, the controller should evenly select S from groups of data nodes which share a write off-loading target node.

2) data allocation algorithm: The data allocation algorithm is a decision method that decides the (initial) nodes to be allocated. This is the problem of selecting x nodes from active nodes. x is the replication number of objects.

Though this data allocation algorithm generally can select nodes from powered down nodes, this algorithm does not select them in this paper because there are no apparent advantages in this methods. One of the possible merits is that data distribution is balanced after node power restitution to increase the candidates of allocated objects. However, because the update access of newly created objects needs the write off-loading process, the system has to apply a write off-loading log during the restitution of nodes and increase the size of the write off-loading log.

In this paper, we focus on selecting candidates for allocation from active nodes for this apply log process because we see virtually no improvement compared with the data migration as re-balance processing at the restitution of nodes.

3) data migration at node power control: Data migration during the node power control should keep the data layout that has redundancy of update access and avoid unacceptable levels of decreased performance.

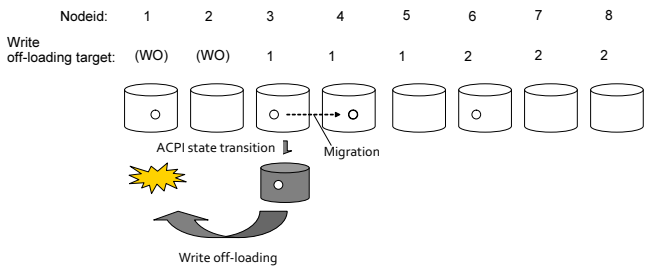

Figure 5. Degrade redundancy by write off-loading

Table I
COMPARISON OF STRATEGIES

	No Write-offloading	Free data allocation with WO	Consideration WO node
Data allocation algorithm	Random	Random	Data allocation algorithm that takes into consideration the potential presence of write off-loading nodes
Data migration at power control	All data at sleeping node	Replicas of sleeping node ' s data (at WO node) and replicas that share WO node (at sleeping node)	Migrate replicas that share WO node (at sleeping node)

Figure 5 shows an example of redundancy being degraded by the write off-loading mechanism. The system in this example is a single storage group in the DKVS, and it consists of two write off-loading target nodes and six standard data nodes. The nodes (Nodeid 3 - 5) are assigned to the nodeid 1 node to write off-loading, and Nodeid 6 - 8 are assigned to the nodeid 2 node, too. The white circle is one of the objects. This object allocates the nodeid 1, 3, and 6 nodes. When the nodeid 3 node shifts to power-down mode, the update information on this object replica of nodeid 3 is transferred to node 1. However, because the nodeid 1 node has this object replica, this update information is recorded to only two nodes. Therefore, the replica on the nodeid 3 node should be migrated to other nodes (the nodeid 4 node in this example).

IV. DATA ALLOCATION/MIGRATION STRATEGIES

In this section, we discuss the data allocation algorithm and migration strategies at node power-down on our proposal architecture.

Table I shows a short summary of these strategies. We describe three strategies in the following sections in detail.

A. No-write off-loading strategy

This strategy is a straight forward approach that does not use the write off-loading scheme.

The data allocation algorithm in this strategy is randomly selects x nodes from active nodes during object creation. In this paper, we employ this method for simplicity of implementation. Consistent hashing-based method, such as Amazon Dynamo, may be selected in this strategy.

The general distributed data store, which cannot use a write off-loading scheme, needs to move all data stocked in a node that will be changed to a sleeping state because if clients issue update request for a object on sleeping nodes, sleeping nodes have access this object must be change to active. This power state change causes large latency time. Therefore, all object replicas should be allocated to active nodes.

The general distributed data store, which cannot use a write off-loading scheme, needs to move all data stocked in a node that will be changed to a sleeping state because, if clients issue an update request for an object on the sleeping nodes, these nodes that have access to this object must be changed to active. This power state change causes a large latency time. Therefore, all object replicas should be allocated to active nodes.

This strategy demands a large quantity of data migration. Moreover, the efficiency of storage capacity is low because sleeping nodes cannot work as having backup replicas.

B. Free data allocation with WO

Free data allocation with the WO strategy uses the write off-loading scheme, does not intermediate during data creation, and resolves a paradox of data layout at the node power control.

The data allocation algorithm in this strategy can be employed from several algorithms. In this paper, we randomly select selection-based methods in the same way as the no write off-loading strategy.

At the power control step, this strategy needs the following objects.

- Replicas of scheduled sleeping node ' s data (on WO node)
- Replicas that share WO node (on scheduled sleeping and slept nodes)

Figure 6 shows the data migration algorithm in this strategy. This algorithm is a step of calcControlPlan() in Figure 4.

C. Consideration WO node

In this section, we propose a data allocation algorithm and strategy to reduce the amount of data migration at power down control. One straightforward approach is to remove write off-loading target nodes from allocated nodes. However, this approach degrades the efficiency of the storage capacity.

We propose a new data allocation algorithm that takes into consideration the potential presence of write off-loading nodes. Figure 7 describes a data allocation algorithm in this strategy used during new object creation. X is the number of replicas. This algorithm takes into consideration

```
create blank migobjects (migration objects list)
o ← extract objects on scheduled sleeping nodes.
for all o do
    if WO target node has o then
        migobjects ← o
    end if
end for
cr ← select replicas (except one replica) share WO target node
on scheduled sleeping and asleep nodes.
for all cr do
    migobjects ← cr
end for
objids ← select all objids that are allocated on sheduled
sleeping nodes.
for all objids do
    if the objects are allocated to all replicas to on scheduled
    sleeping and asleep nodes then
        migobjects ← this object
    end if
end for
for all migobjects do
    set migobjects to migrate to active nodes (keep flat data
    distribution if at all possible )
end for
```

Figure 6. Data migration control algorithm

```
create blank allocatenodes list
candidateNodes = activeNodeList
while allcatenodes.size < X do
    n ← random select from candidateNodes
    put n to allocatenodes
    remove nodes that are assigned n as WO target from
    candidateNodes
    remove node n's WO target node from
    candidateNodes
end while
return allocatenodes
```

Figure 7. A data allocation algorithm that takes into consideration the potential presence of write off-loading nodes

the potential presence of Write off-loading nodes. If one of the object's replicas is allocated on scheduled sleeping nodes, the controller does not need this object's replica to migrate.

However, If this number is smaller than X and the system selects WO target nodes continuously in the random selection phase, $candidateNodes$ becomes blank. Therefore, in order to avoid this case, the system should take a followed step as exceptional measures. In this step, the system removes write off-loading target nodes from $candidateNodes$, if the number of write off-loading target nodes of remaining $candidateNodes$ is smaller than the replication number which doesn't decide on allocate nodes.

If the number of WO target nodes in the SG is bigger than X, this algorithm functions well. However, if this number is smaller than X and the system selects WO target nodes continuously during the random selection phase, the

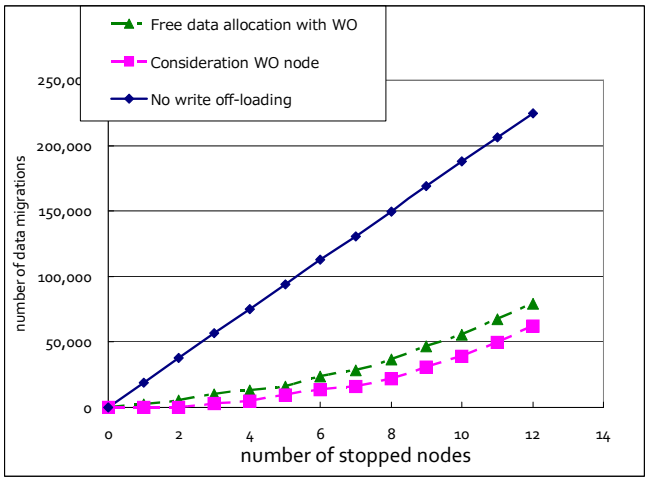

Figure 8. Result of data migration simulation

$candidateNodes$ become blank. Therefore, to avoid this, the system should take the following step as an exceptional measure. In this step, the system removes write off-loading target nodes from $candidateNodes$ if the number of write off-loading target nodes of the remaining $candidateNodes$ is smaller than the number of object replicas, which does not decide on which nodes are allocated.

V. Evaluation

In this section, the proposed strategies are evaluated by using simulation and prototype systems.

A. simulation

1) Amount of data migration: To evaluate the effect of energy saving, we developed a mock-up distributed prototype system. Figure 8 shows the amounts of data migration at power control with three strategies. The parameters of distributed data store system in this graph were 1 storage group and 16 storage nodes (including 4 write off-loading target nodes), and this system was assumed to be able to store 100,000 objects.

This result indicates that the write off-loading scheme reduced the large amount of migration objects. For example, when 12 nodes were stopped, the free data allocation strategy reduced the number of migration objects by 75%, and the consideration data allocation strategy reduced this by 83%. As a result, the reduction of data migrations reduced the time before stopping nodes.

Because the increase in the amount of data migrations increased time until the nodes stop, the system wasted some electrical power. For example, table II shows wasted power for 12 node stops when we assumed that the migration speed in this system was 1,000 objects/sec and the idle power of the storage node was 100 w. Besides, if using less electrical power at the data center is requested on a very short notice,

Table II
ESTIMATION OF WASTING POWER CONSUMPTION

	No write-offloading	Free data allocation with WO	consideration data allocation with WO
time until power control (sec)	224.47	79.28	62.35
power consump-tion(w)	269,364	95,132	74,814

the write off-loading approach and the consideration data allocation strategy reduce handling time. These parameters are determined by reference to our DKVS prototype system's performance. The migration speed depends it's implementation and data object's size. Because the high speed data migration hurts this storage performance [5], improving this implementation of data migration is not sufficient.

2) Effect of data store system: We evaluate storage capacity per node on a mock-up distributed prototype system. Because no write offloading strategy has all accessible objects on active nodes, the number of objects increases as the number of stopped nodes increases. Thus, because the nodes cannot hold objects on memories, performance risks increase. By using the write off-loading mechanism, the system can make full use of the storage capacity of the power-saving nodes.

B. prototype system experiments

1) experimental setup and condition: The experiments were all run using a system consisting of nine standard servers, each with a dual core Intel Xeon E5504 processor and 12 GB DDR2-800 memory. All of the servers were running Debian GNU Linux 5.0.4 (x86-64). The servers were connected with TCP/IP on Gigabit Ethernet networks. This experimental DKVS system consisted of eight standard nodes (one SG; two WO target nodes) and one RM node. The controller operated on one of the DKVS nodes (SGM).

In order to measure the nodes' power consumption, we use Raritan Dominion PXs which are intelligent power tap product. These taps can measure power consumptions every node per 3 seconds.

Table III shows the features of the nodes' power consumption. When these nodes are in a hibernate state (ACPI S4), They use about 22 (w) per one node for their BMC controllers. Each server's peak power consumption was about 180 (w), and our prototype DKVS standard nodes used only about 140 (w) at peak load because they were an underutilized CPU resource.

We use the Yahoo! Cloud Serving Benchmark (YCSB) [6] in our evaluation's workload. The YCSB emulates a synthetic workload generator that can be parameterized to vary the read/write ratio, access distributions, etc.

Table III
FEATURES OF NODE'S POWER CONSUMPTION

peak power	about 180[w] (140[w] at DKVS's standard node)
idle power	about 100[w]
hibernate	about 22 [w]

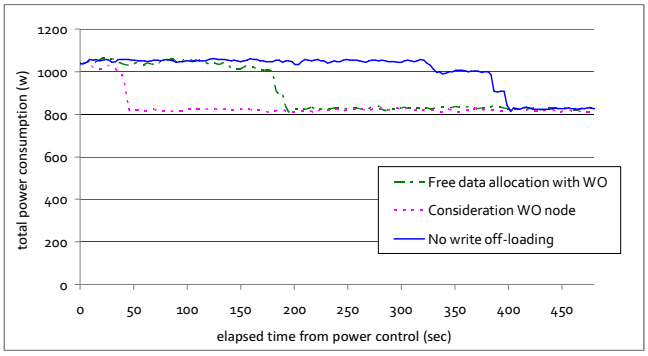

Figure 9. Time-series power consumption on physical servers

Table IV
POWER CONSUMPTION DURING 480 SECONDS FROM POWER CONTROL

strategies	power consumption(w)
Free data allocation with WO	439,018
Consideration WO node	404,271
No write off-loading	483,999

2) Evaluation of reducing power consumption with three strategies: Figure 9 shows time-series power consumption with three strategies. The x-axis is the progress time from start to calculating the control plan on the basis of several strategies. These three strategies use same paek power consumption before controlling the DKVS. YCSB's workload parameters are workload B (the read heavy workload, update 5%, read 95%), object number = 100,000, object size = 1 KB(10 fields, field length = 100 bytes), and the distribution of access is performed by zipfan.

These results show that consideration WO node strategy can reduce the time it takes to hibernate two nodes. Because the consideration WO node strategy in this condition does not need to migrate objects, the system takes time only to calculate a plan and to shift nodes to hibernate mode until shifting to the state of low power consumption. Because the other two strategies have to migrate many objects, they take a large amount of time until shifting to a low power consumption state.

Table IV shows power consumption during 480 seconds from the time of power control. Free data allocation with the WO strategy/consideration WO node strategy reduced power consumptions by more than 79,728/44,982 watts in comparison with the no write off-loading strategy.

Figure 10 shows a time-series of YCSB's throughputs. The x-axis is the same scale as that in Figure 9. The throughputs of all strategies before power control are about 140,000

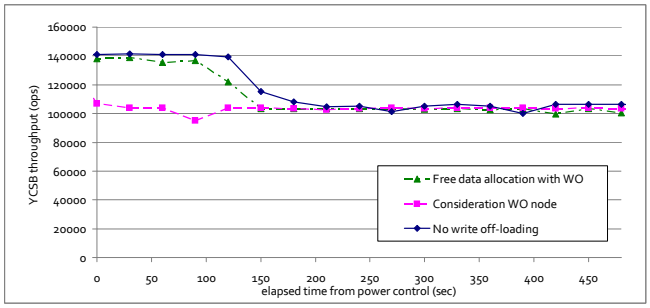

Figure 10. Performance-throughput

operations per second (ops), and the throughputs after power control are about 100,000 ops. In this experiment, because YCSB's workload clients requested as many maximum loads as possible, the DKVS's performance went down due to the hibernation of two nodes by the power control.

Considering the data allocation strategy with WO caused performance to go down about 45 seconds before power consumption went down. This time is almost the time required to put nodes into hibernation. Free data allocation with WO and no write off-loading gradually fell to 100,000 ops during the data migration phase. The reason is that the number of objects on the scheduled sleep nodes reduced as data migrations progressed. Therefore, the DKVS system could not make good use of the scheduled sleep nodes until immediately before the nodes powered down. Free data allocation with WO had less of a bad influence than did no write off-loading because the scheduled sleep nodes had some objects that remained in sleep mode for the write off-loading mechanism. However, because the no write off-loading strategy had to migrate all objects on the scheduled sleep nodes, these nodes could not work immediately before the nodes powered down.

VI. Related Work

The importance of power-proportional systems in data centers is mentioned by Barroso et al. [7].

For data storage systems, spinning down disks is a typical approach to decreasing power consumption, such as in a MAID [2]. However, the early works are related to archival storages. Write off-loading [4] makes power-proportional control with on-line storage systems possible.

Studies with cluster-based storage systems, especially for the Hadoop system, are currently active. GreenHDFS [8] divides nodes into hot and cold zones and controls data placement by using access frequency. Leverich et al. [9] and Amul et al. [10] made some nodes turn off. [10] mentioned that write off-loading is also useful for cluster-based storage; however, they did not evaluate it.

Our architecture and controller is similar to SCADs Director [5]. The objective of this research is to meet strict performance service-level objectives. Our controller focuses on energy-saving and reducing migration objects.

VII. Future Work and Conclusion

Our aspiration is to realize an information society friendly to humans and the earth. The power-proportional high performance data store will be an important component to this earth friendly society. In this paper, we introduced and evaluated a data allocation algorithm that reduce the amount of data migrations to some powered down nodes. We showed that the write off-loading mechanism can make good use of the storage capacity of sleeping nodes. Moreover, a physical evaluation showed that our proposed data allocation algorithm reduces the amount of data migration at power control and can switch to a low-power state.

Acknowledgment

A part of this work belongs to "Green IT Project" which NEC contracted with New Energy and Industrial Technology Development Organization (NEDO) of Japan.

References

[1] J. Koomey, *Growth in data center electricity use 2005 to 2010*. Oakland, CA: Analytics Press, August 2011. [Online]. Available: http://www.analyticspress.com/datacenters.html

[2] D. Colarelli and D. Grunwald, "Massive arrays of idle disks for storage archives," in *Proc. of the Supercomputing '02*, 2002, pp. 1–11.

[3] D. Tsirogiannis, S. Harizopoulos, and M. A. Shah, "Analyzing the energy efficiency of a database server," in *Proc. of SIGMOD '10*, Jun. 2010, p. 231.

[4] D. Narayanan, A. Donnelly, and A. Rowstron, "Write off-loading: practical power management for enterprise storage," in *Proc. of FAST'08*, 2008, pp. 17:1–17:15.

[5] B. Trushkowsky, P. Bodík, A. Fox, M. J. Franklin, M. I. Jordan, and D. A. Patterson, "The SCADS director: scaling a distributed storage system under stringent performance requirements," in *Proc. of FAST'11*, 2011, pp. 12–12.

[6] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears, "Benchmarking cloud serving systems with YCSB," in *Proc.of SoCC'10*, 2010, pp. 143–154.

[7] L. A. Barroso and U. Hölzle, "The case for Energy-Proportional computing," *Computer*, vol. 40, no. 12, pp. 33–37, Dec. 2007.

[8] R. T. Kaushik and K. Nahrstedt, "Evaluation and Analysis of GreenHDFS : A Self-Adaptive , Energy-Conserving Variant of the Hadoop Distributed File System," in *Proc. of CloudCom 2010*, 2010.

[9] J. Leverich and C. Kozyrakis, "On the energy (in)efficiency of hadoop clusters," *SIGOPS Oper. Syst. Rev.*, vol. 44, no. 1, pp. 61–65, Mar. 2010.

[10] H. Amur, J. Cipar, V. Gupta, G. R. Ganger, M. A. Kozuch, and K. Schwan, "Robust and flexible power-proportional storage," in *Proc. of SoCC'10*, 2010, pp. 217–228.