論文 / 著書情報
Article / Book Information

Title	Efficient Gear-shifting for a Power-proportional Distributed Data-placement Method
Author	Hieu Hanh LE, Satoshi HIKIDA, Haruo YOKOTA
Journal/Book name	Proceedings of IEEE BigData 2013, , , pp. 76-84
Issue date	2013, 10
DOI	10.1109/BigData.2013.6691557
URL	http://www.ieee.org/index.html
Copyright	(c)2013 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.

Efficient Gear-shifting for a Power-proportional Distributed Data-placement Method

Hieu Hanh LE*
Department of Computer Science,
Tokyo Institute of Technology
Tokyo, Japan
hanhlh@de.cs.titech.ac.jp

Satoshi HIKIDA
Department of Computer Science,
Tokyo Institute of Technology
Tokyo, Japan
hikida@de.cs.titech.ac.jp

Haruo YOKOTA
Department of Computer Science,
Tokyo Institute of Technology
Tokyo, Japan
yokota@cs.titech.ac.jp

***Abstract*—Energy-aware commodity-based distributed file systems for efficient Big Data processing are increasingly moving towards power-proportional designs. However, current data placement methods for such systems have not given careful consideration to the effect of gear-shifting during operations. If the system wants to shift to a higher gear, it must reallocate the updated datasets that were modified in a lower gear when a subset of the nodes was powered off, but without disrupting the servicing of requests from clients. Inefficient gear-shifting that requires a large amount of data reallocation greatly degrades the system performance. This paper proposes a data placement method known as *Accordion*, which uses data replication to arrange the data layout comprehensively and provide efficient gear-shifting. Compared with current approaches, Accordion reduces the amount of data transferred, which significantly shortens the period required to reallocate the updated data during gear-shifting. The effect of this reduction is larger with higher gears so Accordion is suitable for smooth gear-shifting in multigear systems. Moreover, the times when the active nodes serve the requests are well distributed so Accordion is capable of higher scalability than existing methods based on the I/O throughput performance. Accordion does not require any strict constraint on the number of nodes in the system therefore our proposed method is expected to work well in practical environments. Extensive empirical experiments using actual machines with an Accordion prototype based on the Hadoop Distributed File System demonstrated that our proposed method significantly reduced the period required to transfer updated data, i.e., by 66% compared with an existing method.**

***Keywords*—energy-aware; power-proportionality; data-placement; HDFS**

I. Introduction

Energy-aware distributed file systems for efficient Big Data processing are increasingly moving towards power-proportional designs [1]. In such systems, the power proportionality can be achieved using data placement methods to control the total number of active nodes used to store data in the system. Based on this concept, several data placement policies have been proposed that position data on organized nodes, which aim to provide power proportionality [2]–[4]. These methods share an approach where data are replaced with replicas on organized nodes. First, the system divides all of the nodes into a set of small separated groups. These groups are then configured to operate in multiple gears where each gear contains a different number of groups. A high gear has a large number of groups of active nodes. Thus, the system consumes more power in a high gear, although it can deliver higher performance with suitable processing. Using this mechanism, the system can manage its power consumption and performance so it can deliver power proportionality.

However, the power-proportional data placement methods used by current approaches do not consider the effects of gear-shifting on the performance of the system. Initially, the system may have to update the datasets modified in a low gear when a subset of the nodes was powered off. When the system moves to a higher gear, it needs to guarantee power proportionality to serve the requests of clients. Therefore, the system has to replicate the updated data internally on the reactivated nodes in the background to share the load equally among all of the active nodes. Inefficient gear-shifting degrades the performance of power-proportional distributed file systems greatly.

In this paper, we propose a flexible power-proportional data placement method known as Accordion, which aims to improve the efficiency of gear-shifting by reducing the amount of re-transferred data. Accordion shares the node organization of other methods because it supports a multigear operational mode where each gear is combined from different numbers of different groups of nodes. However, the locations of the primary data in the dataset are different with Accordion. At first, the primary data in the dataset are located equally among all of the nodes in the system. Next, the data are replicated so each of its replicas is located on nodes that belong to different groups. Thus, when the modified dataset is updated (or appended) in low gear, part of the primary data in the updated dataset is already stored on the active nodes so, only the remainder of the updated dataset, which can be assumed to be written to the deactivated nodes, needs to be written temporally. Therefore, only this temporal part of the updated dataset needs to be re-transferred when the system shifts to a higher gear. In other existing methods, the overall updated dataset needs to be reallocated during gear-shifting. As a result, Accordion is expected to reduce the cost of updating during gear-shifting because it reduces the period of data movement.

Moreover, the efficiency of this reduction is increased when gear-shifting is performed in a high gear. The amount of data reallocated is reduced with a lower number of deactivated nodes, which reduces the gear-shifting period further. Thus, the degradation of the system performance is prevented when Accordion-based systems shift to a higher gear. Consequently, Accordion is capable of providing smooth gear-shifting for

*Hieu Hanh LE is currently working for Hitachi Ltd.

multigear power-proportional distributed file systems.

Furthermore, Accordion also benefits from a load balancer that aims to share the workload among all of the active nodes. One of the main goals of the load balancer is to allow all of the active nodes to serve the same amount of data when responding to requests. However, another important factor that affects the performance of the file system greatly is the timing distribution when active nodes serve requests. It should be well balanced in order to provide a higher I/O throughput performance. For example, if two nodes serve 100 MB of data, generally it would be more efficient if two nodes served 50 MB simultaneously rather than sequentially. The different locations of the primary data mean that Accordion can balance the amount of data requested and the service timing on the active nodes.

Additionally, the flexibility to determine the configuration is important in practical applications, such as the number of active nodes in each gear in power-proportional file systems. Rabbit, the first method to achieve power proportionality in a Hadoop Distributed File System (HDFS) [5], has a strict constraint where the number of nodes in each group is defined by an exponential function, which provides ideal power proportionality based on an equal work policy [4]. In Accordion, this constraint is more flexible because the total number of nodes in the system should be an even number. Therefore, it is easier to apply Accordion in a practical environment.

The main contributions of this paper are as follows.

- We propose a distributed data placement method known as Accordion, which utilizes data replication to deliver efficient gear-shifting in power-proportional distributed file systems. The different approach of locating the primary data in Accordion reduces the amount of data updated during gear-shifting, hence shortens the time required to reallocate data.
- Accordion is applicable to smooth gear-shifting in multigear systems because it increases the performance efficiency when gear-shifting is performed in a higher gear.
- Accordion increases the I/O throughput performance because it improve the parallelism effectively distributes both the timing of serving requests and the amount of data requested.
- Accordion is expected to work well in practical environments because it does not require a strict constraint on the number of nodes in file systems.
- We performed extensive empirical experiments using a maximum of 24 nodes with an Accordion prototype on HDFS to evaluate the efficiency of gear-shifting and the performance scalability of power proportionality. The empirical experimental results showed that Accordion significantly improved the performance by 66% compared with red Rabbit.

The remainder of this paper is organized as follows. Related studies are discussed in Sect. II and the design of Accordion is described in Sect. III. Section IV presents a performance evaluation of our proposed method. Our conclusions and future work are discussed in Sect. V.

II. Related Work

Rabbit [4] was the first method to provide power proportionality to an HDFS by focusing on the read performance. Rabbit uses an equal work–data-layout policy based on data replication. The primary replicas are stored evenly among the primary nodes. The remaining replicas are stored on additional and increasingly large subsets of nodes. Each node in the subsets has a fixed order and it stores a number of blocks that is inversely related to its order, which guarantees that the system can distribute the workload equally among all of the active nodes. However, Rabbit still does not support the write workload so it cannot consider the cost of reflecting the updated data in a low gear.

Sierra [3] was designed as a power-proportional distributed storage system for general data centers where a replicated object store supports the write-and-read workloads during multigear operations. This method guarantees the write availability in a low gear by exploiting the concept of write off-loading [6], which was motivated originally by the aim of saving power by spinning down unnecessary disks. This method allows write requests on spun-down disks to be redirected to other active disks in the file system. Thus, this technique increases the spin-down duration, thereby providing additional power savings. This method may be considered as a solution for multigear file systems to deal with updated data in a low gear. Sierra can deal with write requests in a low gear, but it is still not optimized to reflect the updated data efficiently when the system moves to a higher gear.

In previous studies [7], we conducted a simple evaluation of Rabbit and PARAID [2] to identify an appropriate data placement approach to support efficient gear-shifting in power-proportional systems. PARAID uses a skewed pattern to replicate and stripe data blocks to the disks. This facilitates adaptation to the system load by varying the number of active disks in the system. PARAID focuses only on the RAID unit and it is unreliable when adapting to a distributed environment.

We also proposed an architecture known as NDCouplingHDFS [8] to facilitate the efficient reflection of updated data in a power-proportional HDFS. NDCouplingHDFS focuses on coupled metadata management and data management on each HDFS node, which localizes the range of data maintained by the metadata in an efficient manner. This reduces the cost of managing the metadata generated during changes in the system configuration. However, the data placement method was not considered in this study.

Other studies, rather than power-proportional designs, have aimed to reduce the total power consumption based on a trade-off with performance in general storage system [9]–[12].

III. Proposed Method

In this section, we describe the data placement method used by Accordion in detail. Next, we explain the load balancer, which aims to distribute the workload among all of the active nodes and provide power proportionality. Finally, we present the processes that reflect updated data, which are written to the system in a low gear, when the system performs gear-shifting.

Algorithm 1 Algorithm used to assign roles to nodes in the cluster.

Input: All the nodes of the system which are organized in G groups ($Node_g^i$, $g \in [1, G]$, $i \in [1, N_g]$)
Output: $Role(Node_g^i)$

```
for Group_g from Group_1 to Group_G do
    if N_g is even then
        middle = N_g / 2
        for i from middle to 1 do
            order = middle − i + 1
            Role(Node_g^i) = Left_g^order
        end for
        for i from middle + 1 to N_G do
            order = i − middle
            Role(Node_g^i) = Right_g^order
        end for
    else if N_g is odd then
        middle = N_g / 2 + 1
        for i from middle to 1 do
            order = middle − i + 1
            Role(Node_g^i) = Left_g^order
        end for
        for i from middle to N_G do
            order = i − middle
            Role(Node_g^i) = Right_g^order
        end for
    end if
end for
```

A. Accordion design

Accordion was designed to provide power proportionality and a high data I/O throughput in cluster file systems that use commodity computer servers such as HDFS, Google File System [13]. In Accordion, the files are divided into a large number of blocks and a number of replicas of each data block are distributed among the nodes of the cluster. The mapping of the file names to block identifiers is maintained by a separate metadata service.

Like other approaches, Accordion aims to control the power consumption of the system by dividing the nodes in a cluster into several separate groups. A system that uses the Accordion data placement layout can then operate in a multigear mode where each gear contains a different number of groups. The higher gears have more groups of nodes.

Power proportionality is achieved by Accordion's data placement policy, which is based on skewed replication, and a careful consideration of the effects of reflecting the modified dataset when the system moves from low to high gears. The data placement policy is described in detail in the following parts and the symbols used are summarized in Tab. I.

1) Node role assignment: In a system that uses Accordion, changes in the operational modes of nodes are associated with changes in the shape of the bellows of an Accordion (musical instrument) after adjusting for effects on the transitions of a note or between multiple notes. When the system moves up to higher gears or down to lower gears, the active node ranges are expanded or reduced centrally.

In our method, the nodes are arranged geometrically in a horizontal array because the nodes that belong to lower groups are bounded by the nodes of higher groups. $Group_m$ is higher than $Group_n$ if $m > n$. For example, we assume that a system is operating in a two-gear mode with two groups of nodes, i.e., $Group_1$ and $Group_2$. The nodes from $Group_1$ are activated in Gear 1 and the nodes from $Group_1$ and $Group_2$ are turned on

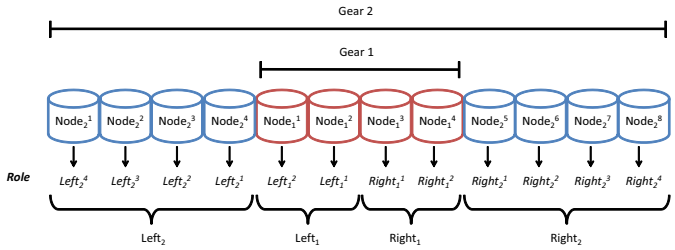

Fig. 1. Example of a two-gear system using Accordion where two groups of nodes have roles, where $G = 2$, $N_1 = 4$, $N_2 = 8$, and $N = 12$.

in Gear 2. In this case, $Group_2$ is higher than $Group_1$. Thus, the nodes of $Group_1$ are bounded by the nodes of $Group_2$.

To perform data replication among the nodes in the system, i.e., to identify the destination nodes for storing the backup data of a specific node, each node is assigned its corresponding role in the group. The role of each node is determined by the configuration, which is related to the number of groups and the total number of nodes in each group. The algorithm used to assign the roles to nodes is shown in Algorithm 1:

$$Role(Node_g^i) = Part_g^{order}, \tag{1}$$

where $Part$ is either $Left$ or $Right$ and $order \in [1, N_g + 1]$.

Definition 1: For a $Group_{gA}$ that contains an odd number of nodes (N_{gA} is an odd number), there is a node that has two roles, i.e., $Left_{gA}^1$ and $Right_{gA}^1$.

An image of a multigear system using Accordion with organized nodes and roles are shown in Fig. 1. Gear 1 requires the nodes from $Group_1$ (a set of $Left_1$ and $Right_1$) to be activated while Gear 2 requires the nodes from both $Group_1$ and $Group_2$ to be activated.

2) Skewed data replication in Accordion: The three goals of Accordion are to provide power proportionality in the read performance, reduce the cost of reflecting updated data when the system changes gear, and guaranteeing the data reliability in all file system operating modes. Thus, we need to apply the policies shown below to satisfy these goals.

a) Location of primary data: First, we distribute the primary data in the dataset to all of the nodes in the system. This means that each node stores the same amount of data.

b) Location of backup data: Starting with the highest $Group_G$, we replicate the data stored in this group to the next lower group $Group_{G-1}$. Thus, the node with the role $Part_{G-1}^{order}$ in $Group_{G-1}$ is allocated to the backup data for the data from the nodes in $Group_G$, which have roles in the following range, if they exist.

$$[Part_G^{(middle-order)rate_G+1}, Part_G^{(middle-order+1)rate_G}],$$

where $Part$ is $Left$ or $Right$, $rate_G = \frac{N_G}{N_{G-1}}$, and

$$middle = \begin{cases} \frac{N_G}{2} & N_G \text{ is even} \\ \lfloor \frac{N_G}{2} \rfloor + 1 & N_G \text{ is odd} \end{cases} \tag{2}$$

For example, in Fig. 1, the data from the range $[Left_2^3, Left_2^4]$ are replicated in the node with the role $Left_1^2$. This

TABLE I. NOTATIONS

Symbol	Description
N	Number of nodes in the cluster
G	Number of groups in the cluster
$Group_g$	A group of nodes with index g ($g \in [1, G]$)
N_g	Number of nodes in group G_g
$Node_g^i$	A node with index i in group G_g ($i \in [1, N_g]$)
$Role(Node_g^i)$	The role of a node $Node_g^i$, is $Part_g^{order_i}$, where $Part$ is $Left$ or $Right$
$Left_g$	$Left_g = \forall Node_g^i \in Group_g, Role(Node_g^i) = Left_g{}^{order}$
$Right_g$	$Right_g = \forall Node_g^i \in Group_g, Role(Node_g^i) = Right_g^{order}$
B	Total number of blocks in a dataset
$V(B)$	Storage requirements to store a dataset with B blocks
$V(B)_g$	Storage requirements to store a dataset with B blocks in $Group_g$

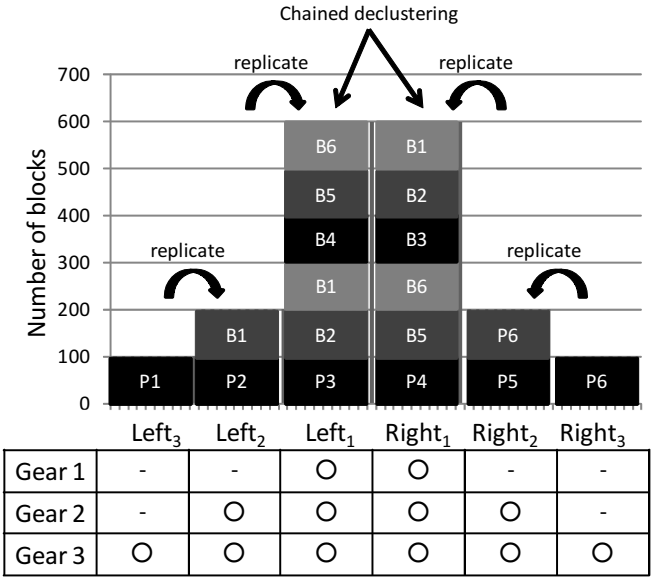

Fig. 2. An example of a three-gear system based on Accordion data placement.

means that the data in $Node_2^1$ and $Node_2^2$ are replicated in $Node_1^1$.

The process is finished when the backup data for the nodes in $Group_2$ are replicated to the nodes in $Group_1$.

c) Chained declustering at the smallest group: To guarantee the data reliability in the lowest gear, we apply the chained declustering policy to the smallest group ($Group_1$). Each node replicates its data to its neighbor node, which guarantees that all of the data in the dataset are replicated in the two neighbor nodes. In Fig. 1, the data (the primary data and the backup data) in $Node_1^1$ are replicated in $Node_1^2$, then from $Node_1^2$ to $Node_1^3$, and so on. In other power-proportional approaches, the data reliability in the lowest gear is often omitted because the data are not replicated.

Figure 2 shows an example of a three-gear system based on Accordion data placement where $B = 600$, $G = 3$, and $N = 60$ ($N_1 = N_2 = N_3 = 20$). $Group_i$ contains the nodes of $Left_i$ and $Right_i$. Gear 1 contains only the nodes in $Group_1$. To serve a request, Gear 2 requires the nodes in $Group_1$ and $Group_2$ to be activated, while Gear 3 requires all of the nodes in $Group_1$, $Group_2$, and $Group_3$ to be activated. The primary data in 600 blocks are stored equally among all of the nodes so each group locates 100 blocks initially. Next, the data in $Group_3$ are replicated in $Group_2$ and the data in $Group_2$ are replicated in $Group_1$. Finally, the chained declustering method is used to replicate the data in $Group_1$. Thus, the three groups contain 1200, 400, and 200 blocks.

3) Storage requirements: This section describes the total storage requirements for storing a dataset based on the skewed data replication policy used by Accordion. The total of blocks in a dataset is B and there are G groups, where each group contains N_g nodes. The total amount of data blocks $V(B)$ stored is calculated as:

$$V(B) = \sum_{g=1}^{G} V(B)_g, \tag{3}$$

where $V(B)_g$ is the number of data blocks located on the nodes of $Group_g$. Initially, all of the blocks are stored equally among all of the nodes in the cluster so $V(B)_G = B \times \frac{N_G}{N}$, where $N = \sum_{g=1}^{G} N_g$ is the total number of nodes in the system. In addition to the original data, the nodes in $Group_{G-1}$ are also used to locate the replication data from the nodes in $Group_G$. Thus,

$$V(B)_{G-1} = B\frac{N_{G-1}}{N} + B\frac{N_G}{N} = B(\frac{N_{G-1}}{N} + \frac{N_G}{N}). \tag{4}$$

The amount of data in the remaining groups from Group (G-2) to Group 1 are similarly calculated. Finally, chained declustering is applied to $Group_1$ so the storage required by $Group_1$ is as follows:

$$V(B)_1 = 2\frac{B}{N}(\sum_{g=1}^{G} N_g) = 2B. \tag{5}$$

Substituting the values of $V(B)_g$, ($g \in [1, G]$) into (3) yields the following:

$$\begin{aligned} V(B) &= B(2 + \frac{N_G}{N} + (\frac{N_{G-1}}{N} + \frac{N_G}{N}) + ... + \sum_{i=2}^{G-i+1} \frac{N_i}{N}) \\ &= B\sum_{g=1}^{G} \frac{(g+1)N_g}{N}. \end{aligned} \tag{6}$$

4) The amount of data transferred during gear-shifting: In this section, we calculate and compare the amount of data that is transferred internally when the system changes its configuration using Accordion and other methods such as Rabbit and Sierra. In such methods, as each group contains one replica of the dataset, the entire updated dataset is transferred

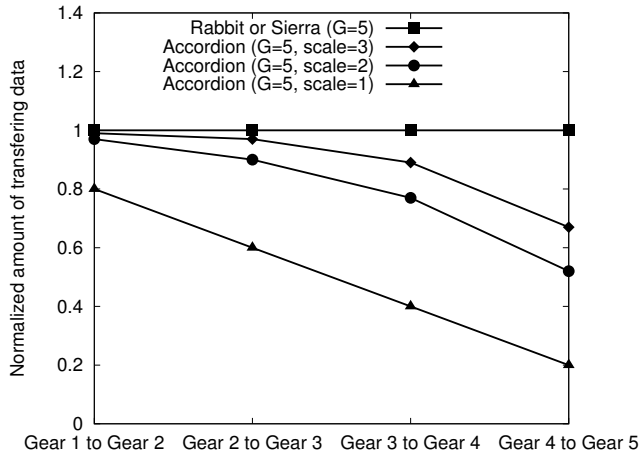

Fig. 3. The normalized amount of data transferred when gear-shifting is performed in each gear

during each gear-shifting. In Accordion, however, shifting between each gear in the system only leads to variation in the subpart of the dataset located on the nodes that are reactivated during transitions.

From the calculations in Section III-A3, the amount of data moved when the system changes from $Gear_g$ to $Gear_{g+1}$ is,

$$V_g(D) = D \sum_{i=g+1}^{G} \frac{N_i}{N}, \tag{7}$$

where G is the number of groups, N_i is the number of active nodes in $Group_i$, N is the total number of nodes, and D is the number of blocks updated in the dataset in $Gear_g$. A constraint on Accordion is that the preferred number of nodes in $Group_g$ is a multiple of the number of nodes in $Group_{g-1}$. To evaluate this further, we consider a simple case where $\forall g \in [1, G], \frac{N_{g+1}}{N_g} = scale$ $(scale \geq 1)$. Substituting into (7), we obtain the following:

$$V_g(D) = D \frac{\sum_{i=g+1}^{G} scale^i}{\sum_{i=0}^{G-1} scale^i}. \tag{8}$$

Figure 3 shows the amount of data moved, which was normalized against the amount of the dataset that was updated during each gear shift with Accordion and other methods (Rabbit and Sierra), where the systems were configured to operate using five gears. The amount moved was always smaller in Accordion than Rabbit or Sierra and it became smaller when the system performed gear-shifting in a higher gear. The degradation of the system performance was reduced so Accordion may be applicable to smooth gear-shifting in multigear power-proportional file systems where the number of gears is high.

5) Skewed dataset distribution in Accordion: The skewed data replication in Accordion leads to an imbalance in the amount of data stored on each node. Like in Rabbit and Sierra, some nodes (in the smaller groups) store considerably more data than others (in the higher groups). However, this imbalance does not lead to considerable problem to the overall I/O performance as current hard disk drives normally make use only part of the capacity other than full of the capacity to provide the I/O requests. Given that the amount of unused storage is increasing [14] and that the energy problem is a greater focus, the provision of smooth gear-shifting is more important when delivering power-proportional systems.

6) Physical data placement on a node using Accordion: In Accordion, each node stores the primary data (the original data allocated) and the backup data (replicas of the data from other nodes) so the physical locations of these two types of data in the disks should be well designed. Normally, the data are stored physically in a number of sectors of the disks. The complexity of writing and replicating the data means that the orderless arrival timing of writing requests on each node promotes the discreteness of the physical locations of the sectors for the primary and backup data on the disks. However, on current disks, the seek time required to allocate the responsible sectors still degrades the I/O performance greatly, especially for discretely located sectors. Therefore, it is preferable to allocate the responsible sectors to sequential locations on the physical disks.

In this paper, we also propose an optimized version of Accordion known as *Accordion-with-Disk-Partition (Accordion-DP)* that uses a partitioning technique on each node, which guarantees that the primary data and backup data will be located in two separate parts of the disks. Thus, the physical locations of the sectors from the primary and backup data in each part are allocated sequentially. For example, in Fig. 2, in $Left_1$, P3, B1–B2 and B4–B6 are stored in separated partitions of the disks. Accordion-DP is expected to improve the I/O throughput performance of file systems because it reduces the seek time on the disks.

B. Load balancer

When a request for a block is received from a client, the file system has to select the node that will serve the request because the block is replicated in multiple nodes in the system. To provide power proportionality when serving a dataset with multiple blocks, it is preferable to balance the load among all of the available nodes. The load balancing mechanism used in Accordion is basically the same as that employed by Rabbit. If n nodes are active when serving a B-block dataset, the goal of the load balancer is to make each node serve $\frac{B}{n}$ blocks. Similar to Rabbit, we define an ideal value for each node where $Node_g^i$ equals $\frac{B}{n \times contain_g^i}$. When the dataset is being read, the current hit of the node $Node_g^i$ is $\frac{served_g^i}{1 \times contain_g^i}$, where $contain_g^i$ is the number of blocks stored locally on $Node_g^i$, and $served_g^i$ is the number of blocks already served by $Node_g^i$. From the possible nodes that may store a requested block, the load balancer greedily selects the node where the distance between the current hit and the ideal value is the largest.

Although this mechanism is similar to Rabbit, the different policy of locating the primary data makes Accordion preferable to Rabbit. In Rabbit, each of the replica set of the whole dataset is stored separately in each group's node while in Accordion, there is always part of the dataset that is stored only in the lowest group's nodes. Consequently, the lowest group's nodes can serve the request from the beginning and hence it leads to

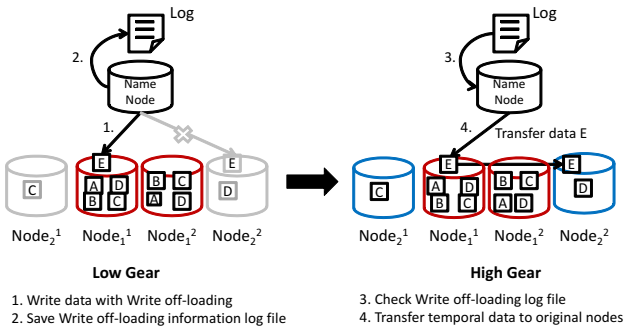

Fig. 4. Updated data reflection using write off-loading.

less skew in distributing the load among nodes in Accordion than in Rabbit. It is explained in more detail and clearer in Sect. IV-C2 through utilizing an empirical experiment.

C. Writing data using write off-loading

When a subset of a group of nodes is deactivated in a low gear, the system can accept write requests from clients using the write off-loading technique. Next, when the system moves to a higher gear by reactivating nodes, it transfers the updated data internally in the background without stopping the processing of read serves from clients.

1) Writing new data using the write off-loading technique: When the system is in a low gear and it has to deal with requests to write new data to a previously stored dataset, the placement of data is performed according to its original policy. Because the system operates in low gear, certain parts of the new data cannot be written to their corresponding deactivated nodes. Hence, the system selects another node randomly from the active nodes to serve this request. Information about the data, the temporary node, and the intended node are saved in a log file. An example of the write off-loading process is shown in Fig. 4. In this example, $Node_2^2$ should serve the write request for data E according to the original data placement policy. However, it is powered off in a low gear so the system decides to use an alternative node, i.e., $Node_1^1$ is selected in this case.

2) Reflecting updated data: When the system changes to a high gear to serve a request for high-performance processing with the newly updated dataset, it needs to perform two functions. The first function is to transfer the data written in temporary nodes to their actual intended nodes. This can be achieved by reading the information in the log files. In Fig. 4, $Node_1^1$ is set to transfer data E to $Node_2^2$. The second function is to serve a request by scanning the new dataset in the storage system. When there is no need to correct the data layout, compared with a normal service in a high gear, the cost of allowing the system to operate in multiple modes to save power will depend greatly on the quantity of data that needs to be transferred.

IV. Experiments

We conducted an empirical experiment using actual machines to verify the methods proposed in this study. We chose Rabbit which also shares the idea of achieving power-proportionality as the comparative method. First, we compared Accordion with Rabbit with respect to power proportionality during the performance of reads. Next, we tested the effectiveness of Accordion for reducing the cost of reflecting updated data when the system shifted to a higher gear. Finally, we evaluated the power proportionality of Accordion using several configurations.

TABLE II. Specification of a node

CPU	TM8600 1.0 GHz
Memory	DRAM 4 GB
NIC	1000 Mb/s
OS	Linux 3.0 64-bit
Java	JDK-1.7.0

TABLE III. Experimental environment

Number of gears	3
Number of active nodes (Accordion, Accordion-DP)	2, 8, 20
Number of active nodes (Rabbit)	2, 7, 21
Number of files	420
File size	64 MB
HDFS version	0.20.2
Block size	32 MB

A. Implementation

In this study, we implemented a prototype of the power-proportional distributed file system Accordion based on a modified HDFS. We changed the current class used to select block locations via an interface where different data-layout policies can be performed. We also added the load balancing mechanism in to HDFS. The write off-loading policy was implemented in the low-power mode, which selects the temporal nodes for writing the new data for the currently turned off nodes. To guarantee the data reliability, block replicas were written to distinct nodes in all of the operation modes.

B. Framework used for the experiments

Our test-bed for the experiments comprised dozens of commodity nodes, which was based on the HDFS architecture with one NameNode, and a cluster of nodes for storing data. We were focused on the energy-aware commodity system so we used low power consumption ASUS Eeebox EB1007 machines, the specifications for which are provided in Tab. II. In the experiments, The power consumption of the cluster of data storage nodes was measured using an AC/DC Power HiTESTER 3334 [15]. The interconnect was a 1000 Mbps switching hub and all of the inactive nodes were hibernating.

C. Power proportionality of Accordion and Rabbit

We evaluated the effectiveness of Accordion by determining the ratio of the throughput when reading a dataset and the power consumption of the cluster of data-storing nodes. In addition to Accordion and Rabbit, we also configured Accordion-DP to test whether it improved the throughput performance.

1) Experimental environments: We compared Accordion and Accordion-DP with Rabbit in terms of their power proportionality when performing reads. These systems were operated using a three-gear configuration based on three groups of

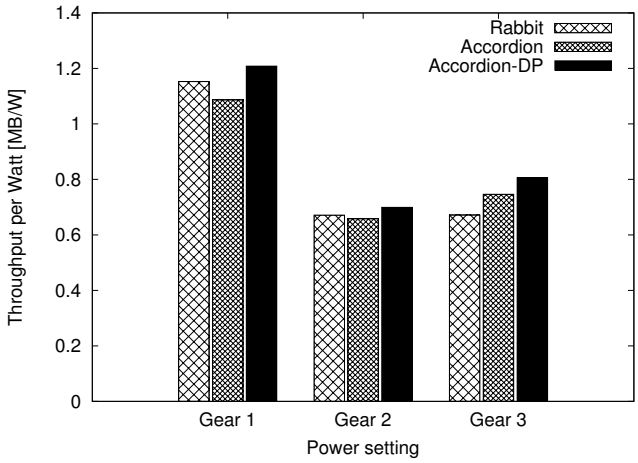

Fig. 5. Throughput per watt using Accordion-DP, Accordion, and Rabbit with various power settings.

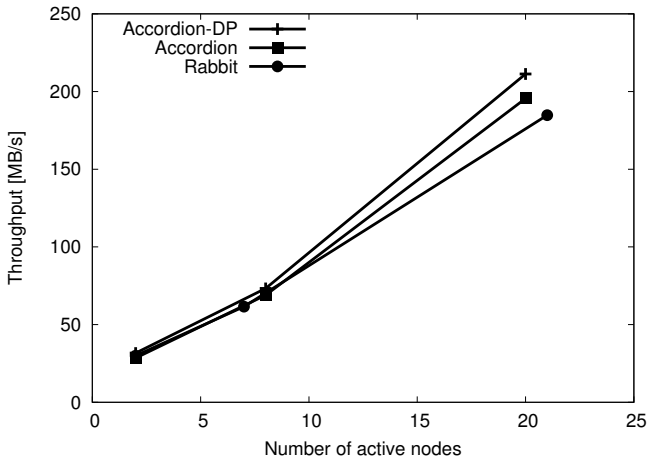

Fig. 6. Read performance using Accordion-DP, Accordion, and Rabbit with various power settings.

nodes. The constraint on the number of nodes in the gear configuration meant that the three gears in Rabbit contained 2, 7, and 21 nodes. To ensure a fair evaluation, the three gears in Accordion and Accordion-DP had 2, 8, and 20 nodes. The scanning workload was distributed uniformly from all of the active nodes. The other information can be referred from Tab. III.

2) Experimental results: Figure 5 shows the throughput per watt to compare the read performance using the aforementioned dataset with three power-setting modes: Gear 1, Gear 2, and Gear 3. The results were the averages of five runs and the Linux buffer cache was cleared between runs. Of the three configurations, Accordion-DP yielded the best results whereas Rabbit had the worst results in Gear 2 and Gear 3. In Gear 3, Accordion-DP delivered 20% better results than Rabbit because it gained better throughput as can be seen in Fig. 6 which shows the average throughputs for the three configurations. The reason was that the better load balancer in Accordion-based increased the throughput while consuming less power (20 nodes vs 21 nodes in Rabbit).

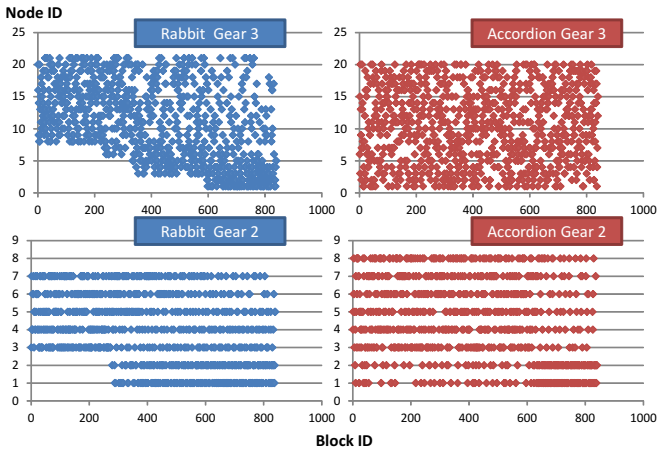

Fig. 7. Distribution of timing on the nodes when serving the blocks in Gear 2 and Gear 3 using Rabbit and Accordion.

The load balancer was the same in all configurations so the differences may be explained by the timing distribution when each node served the requested blocks. Figure 7 shows the results for Rabbit and Accordion in Gear 2 and Gear 3. In each graph, the vertical axis is the NodeID while the horizontal axis shows the BlockID of the dataset. In Rabbit, the nodes in ranges $[1, 2]$, $[3, 7]$ and $[8, 21]$ consequently belong to $Group_1$, $Group_2$ and $Group_3$. In Accordion, the corresponding ranges are $[1, 2]$, $[3, 8]$ and $[9, 20]$. It is easy to recognize that the distribution of the access to blocks was far more balanced with Accordion than Rabbit. In Rabbit and Accordion, the load balancer tended to transfer the workload to the nodes in the outbound group, which could be turned off when the system operated in a lower gear. Rabbit stored the replicas of the dataset in group units, i.e., $Group_1$ stored the primary data, $Group_2$ stored the secondary replicas, etc., so certain groups of nodes were selected centrally by the load balancer when scanning the dataset. By contrast, the skew of the timing distribution was lower when selecting nodes to serve the requests using Accordion. This is because Accordion allocated the primary data among all of the nodes in the system initially so some data were only stored in the low group. Thus, the active nodes in these groups could be selected to serve the requested files from the start of the scanning process.

In this experiment, the benefit of the partitioning technique for separating the primary and backup data areas on nodes was confirmed as Accordion-DP gained better results than Accordion. From the latter experiments, Accordion-DP was used to evaluate the effectiveness of Accordion.

D. Effect of reflecting the updated data on the performance

This experiment evaluated the effect of reflecting the updated data when the system served read requests while performing updated data reflection in the background.

1) Experimental method: First, part of a similar dataset to that used in the above experiment was written to the system while the system operated in Gear 3. The operation mode of the system was then shifted down to Gear 2 by hibernating the nodes of $Group_3$. Next, the remaining parts of the dataset were written to the file system using the data-layout policy

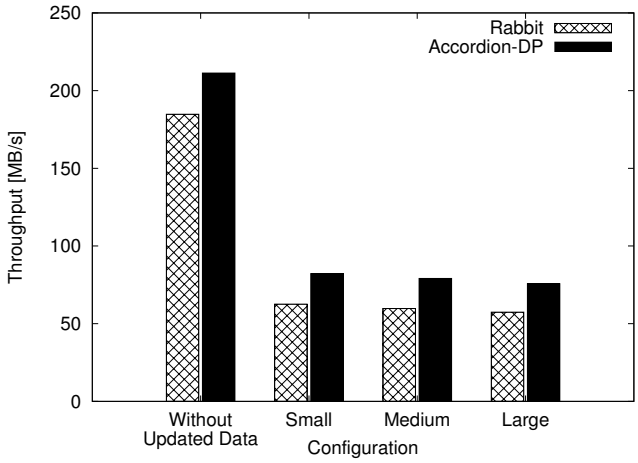

Fig. 8. Average throughput for scanning the dataset.

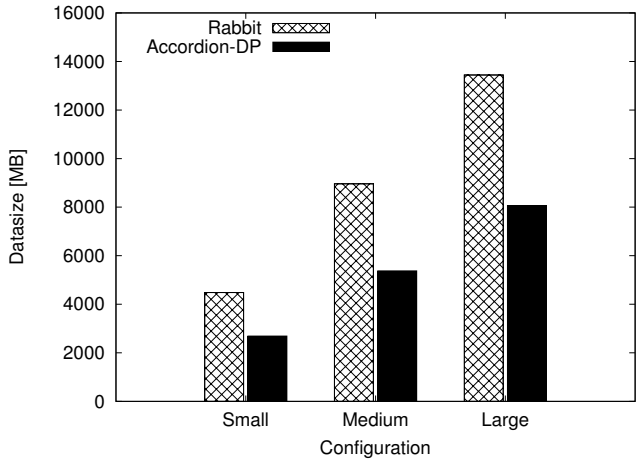

Fig. 9. The amount of data that had to be re-transferred when the system moved from Gear 2 to Gear 3.

with the write off-loading technique. Finally, after finishing the write requests, the system moved to Gear 3 by reactivating the nodes of $Group_3$ at the same time as it served the scanning workload, which was generated in a similar way to that used in the experiment above. The execution time for serving the read requests was measured to verify the effect of updated data reflection on the file system. Here, the total number of files used in the dataset was 420 and each file contained two blocks, which was the same as the experiment above. We also changed the sizes of the parts of the dataset written when the system operated in Gear 2 to 140, 280, and 420 files (small, medium, and large update configurations, respectively). In each case, the number of files written previously when the system operated in Gear 3 was 700, 560, and 420, respectively, so there was no need to re-transfer these files.

2) Experimental results: Figure 8 shows the experimental results as the average throughput while scanning the dataset in three cases: small, medium, and large amounts of transferred data. The results were compared with the results when the system performed the read requests in Gear 3 (without updated data), which were the same as the previous experiment. It is seen that configurations based on Accordion-DP and Rabbit were affected significantly by the updated data reflection. The throughputs were degraded significantly by more than 60% (66% and 61% for Rabbit and Accordion-DP). It became worse with higher amounts of re-transferred data. This confirmed the need to design the data placement method carefully to ensure efficient gear-shifting in power-proportional file systems.

The results also indicated that Accordion-DP improved the performance by 30% compared with Rabbit. The reason is with Accordion-DP, the amount of data re-transfer required was less than that with Rabbit. With Accordion-DP, only part of the dataset written when the system operated in Gear 2 had to be re-transferred when the system moved to Gear 3. By contrast, the whole dataset had to be re-transferred with Rabbit as the placement policy located replicas of the dataset in each group.

Figures 9 and 10 show the amounts of re-transferred data and the execution times required to finish the re-transfer process. Figure 9 shows that with Rabbit, the amounts of data in the small, medium, and large cases were 4480 MB, 8960 MB, and 13440 MB; while with Accordion-DP were 2688 MB, 5376 MB, and 8064 MB, respectively. As shown in Figure 10, Accordion-DP significantly reduced the time required to finish the updated data-re-transfer process by up to 66% (Small case) compared with Rabbit.

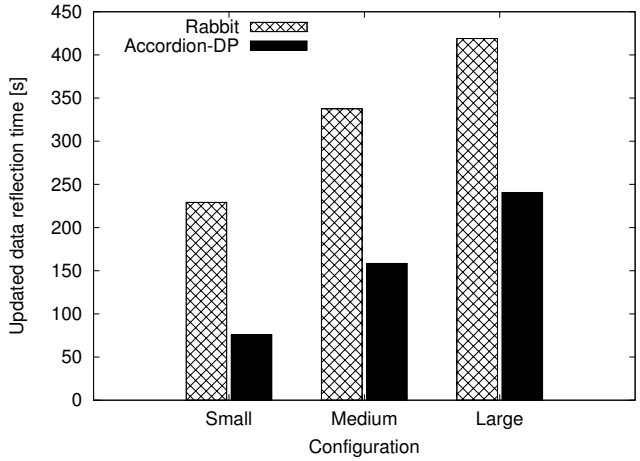

Fig. 10. The execution time for the data-re-transfer process when the system moved from Gear 2 to Gear 3.

E. Effects of different configurations of Accordion

We also performed an experiment to evaluate the effects of different configurations of Accordion on the power proportionality of the system. We used three configurations with different numbers of active nodes. The system was assumed to operate in a three-gear mode: Gear 1, Gear 2, and Gear 3. In the configuration Accordion-DP (x, y, z), x, y, and z indicate the number of active nodes in Gear 1, Gear 2, and Gear 3, respectively. The other environments were the same as those used in the previous experiment in Tab. IV.

Figure 11 shows the performance (throughput per watt) for the three configurations we evaluated. With a larger scale configuration, the power-proportional factor appeared to converge on a high result. In all configurations, the factor increased from Gear 1 to Gear 3 because the number of active nodes increased.

TABLE IV. EXPERIMENTAL ENVIRONMENT

Number of Gears	3
Number of active nodes Accordion-DP (4, 8, 12)	4, 8, 12
Number of active nodes Accordion-DP (4, 12, 20)	4, 12, 20
Number of active nodes Accordion-DP (8, 16, 24)	8, 16, 24
Number of files	420
File size	64 MB

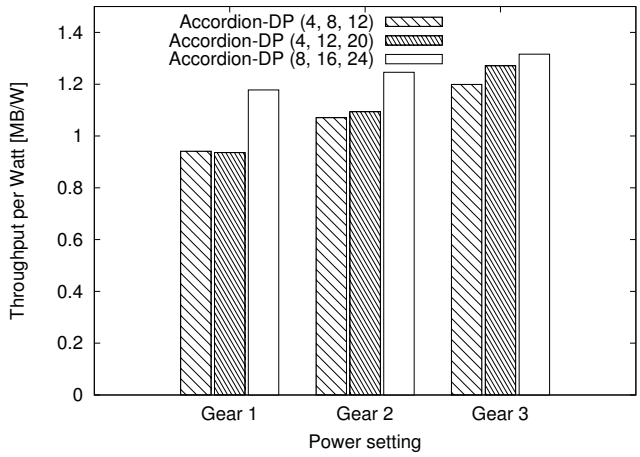

Fig. 11. The throughput per watt with Accordion-DP in several configurations.

Furthermore, from the results at three gears, Accordion-DP (4, 12, 20) experienced the worst standard deviation (0.114 compared with 0.04 and 0.08 in Accordion-DP (8, 16, 24) and Accordion-DP (4, 8, 12), it could be suggested that Accordion is more suitable with the configuration that the number of active nodes in each group is equal.

F. Discussion

The data placement strategy used by Accordion ensures high power-proportionality performance of systems, especially when systems operate in a high gear mode with a larger amount of active nodes. In Accordion, the amount of data requested and the access timing of the active nodes are well distributed among all the active nodes so the I/O throughput performance scales well with the system's size. Furthermore, Accordion was highly effective at reducing the cost of gear-shifting because it improved the performance by 66% when transferring the updated data. The numerical analysis in Sect. III-A4 showed that the efficiency was even greater when gear-shifting was performed in a higher gear.

V. CONCLUSION AND FUTURE WORK

Recently, energy-efficient infrastructures for Big Data processing are gaining much attention from both industrial and academia. In this paper, we identified the issue of ineffective gear-shifting in power-proportional distributed file systems and we proposed the Accordion data placement method to address this problem. Furthermore, the Accordion configuration is highly flexible because the number of nodes in each group can be determined by a weak constrain. In Accordion, the data reliability in the lowest configuration which is omitted in other methods, is ensured through utilizing chained declustering. Extensive experiments using actual machines with the Accordion prototype verified the effectiveness of Accordion. Accordion reduced the execution time required for updated data movement by 66% and improved the power-proportional performance by 20% compared with Rabbit. Furthermore, the numerical analysis and the experimental results also indicated that Accordion is applicable for smooth gear-shifting in multi-gear systems.

In the future, we would like to confirm the effectiveness of Accordion in different experimental environments. Moreover, we want to integrate Accordion with architectures other than HDFS. We will also consider developing a specific algorithm to deal with system failures.

ACKNOWLEDGMENTS

This work was supported partly by Grants-in-Aid for Scientific Research from the Japan Science and Technology Agency (A) ($\#22240005$).

REFERENCES

[1] L. A. Barroso and U. Hölzle, "The Case for Energy-proportional Computing," *Computer*, vol. 40, pp. 33–37, 2007.

[2] W. Charles, O. Mathew, Q. Jin, W. A.-I. Andy, R. Peter, and K. Geoff, "PARAID: A Gear-shifting Power-aware RAID," *ACM Transaction on Storage*, vol. 3, 2007.

[3] E. Thereska, A. Donnelly, and D. Narayanan, "Sierra: Practical Power-proportionality for Data Center Storage," in *Proc. the 6th European Conference on Computer Systems*. ACM, 2011, pp. 169–182.

[4] A. Hrishikesh, C. James, G. Varun, G. Gregory R., K. Michael A., and S. Karsten, "Robust and Flexible Power-proportional Storage," in *Proc. the 1st ACM Symposium on Cloud Computing*. ACM, 2010, pp. 217–228.

[5] Apache Hadoop, "HDFS Hadoop Wiki," http://wiki.apache.org/hadoop/HDFS.

[6] D. Narayanan, A. Donnelly, and A. Rowstron, "Write Off-loading: Practical Power Management for Enterprise Storage," *ACM Transaction on Storage*, vol. 4, no. 3, pp. 10:1–10:23, 2008.

[7] H. H. Le, S. Hikida, and H. Yokota, "An Evaluation of Power-proportional Data Placement for Hadoop Distributed File Systems," in *Proc. Cloud and Green Computing*. IEEE Computer Society, 2011, pp. 752–759.

[8] H. H. Le, S. Hikida, and H. Yokota, "NameNode and DataNode Coupling for Power-proportional Hadoop Distributed File System," in *Proc. the 18th International Conference on Database System for Advanced Applications, Part II*. Springer Verlag, 2013, pp. 99–107.

[9] 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 *Proc. IEEE International Symposium on Modeling, Analysis and Simulation of Computers and Telecommunication Systems*, 2008, pp. 1–8.

[10] J. Leverich and C. Kozyrakis, "On the Energy (In)efficiency of Hadoop Clusters," *SIGOPS Operating System Review*, vol. 44, pp. 61–65, 2010.

[11] R. T. Kaushik and B. Milind, "GreenHDFS: Towards an Energy-conserving, Storage-efficient, Hybrid Hadoop Compute Cluster," in *Proc. the 2010 International Conference on Power Aware Computing and Systems*. USENIX, 2010, pp. 1–9.

[12] J. Kim and D. Rotem, "Energy Proportionality for Disk Storage using Replication," in *Proc. the 14th International Conference on Extending Database Technology*. ACM, 2011, pp. 81–92.

[13] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The Google File System," in *Proc. the 19th ACM Symposium on Operating Systems Principles*. ACM, 2003, pp. 29–43.

[14] J. Gray, "Greetings from a Filesystem User," in *Proc. the fourth USENIX Conference on Files and Storage Techniques*. USENIX, 2005.

[15] "AC/DC Power HiTester 3334," http://www.hioki.com/product/3334/index.html.