International Journal of Networked and Distributed Computing, Vol. 3, No. 3 (August 2015), 167-176

Effective Management of ReRAM-based Hybrid SSD for Multiple Node HDFS

Nayoung Park
College of Information and Communication Engineering Sungkyunkwan University
Suwon, Korea
E-mail : parkny42@skku.edu

Byungjun Lee
College of Information and Communication Engineering Sungkyunkwan University
Suwon, Korea
E-mail : byungjun@skku.edu

Kyung Tae Kim
College of Information and Communication Engineering Sungkyunkwan University
Suwon, Korea
E-mail : kyungtaekim76@gmail.com

Hee Yong Youn
College of Information and Communication Engineering Sungkyunkwan University
Suwon, Korea
E-mail : youn7147@skku.edu

Abstract

Recently, the research of Hybrid ReRAM/MLC NAND SSD is rapidly expanding into the storage areas. Most existing researches of Hybrid SSD are based on a single storage, while the management of multiple nodes like HDFS is still immature. In this paper a new efficient cold data eviction scheme is proposed which is based on node congestion probability. Computer simulation reveals that the proposed scheme significantly reduces replication and recovery time in comparison to the existing replication schemes.

Keywords: HDFS, hybrid SSD, node congestion probability, replication, cold data eviction

1. Introduction

The demand on efficient storage for cloud computing has been growing drastically year after year. Rather than relying on traditional centralized storage arrays, the storage system for cloud computing consolidates a large number of distributed commodity computers into a single storage pool. It provides large capacity and high performance storage service even in unreliable and dynamic networking environment at low cost. In building the cloud storage system, increasing number of industries and research institutions rely on the Hadoop Distributed File System (HDFS) [1]. HDFS provides reliable storage and high throughput access to the data. It is suitable for the applications manipulating large data sets, typically the ones employing a replication management scheme for data-intensive computing. HDFS has been widely used as a common storage appliance for cloud computing.

One of the conspicuous trends in recent storage system is the emergency of SSD. SSD-based storage is becoming a promising technology for next-generation storage due to a number of reasons including low access

Published by Atlantis Press

latency, low power consumption, higher resistance to shocks, light weight, and increasing focus on endurance. Due to the inherent merits of solid state device, SSD can provide better I/O performance compared with the traditional HDD. Recently, a large number of Internet service providers have started to replace HDD in their data centers with SSD [2]. In addition, the falling cost of NAND flash-based SSD is driving its expansion into the market previously reserved for HDD. Fig. 1 shows the architecture of SSD.

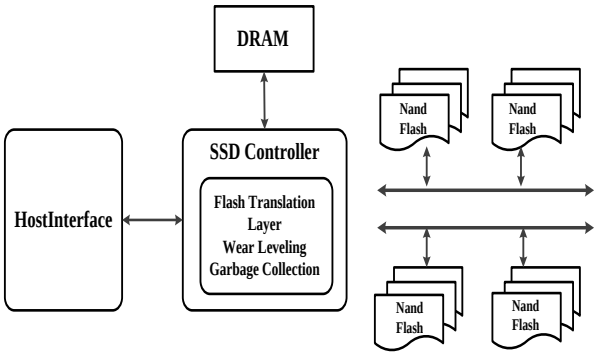

Fig. 1. The block diagram of a traditional SSD architecture.

With the momentum leveraged by both personal computing and enterprise system, SSD has been recognized as a viable choice to build high-performance storage. In order to satisfy the strict requirements on the future storage system of higher speed, reliability and energy-efficiency, SSD is preferred. Various methods contributing to the advancement of the SSD technology have thus been proposed.

Recently, a hybrid ReRAM/MLC NAND SSD employing the cold data eviction (CDE) algorithm was proposed [3]. It dynamically evicts cold pages from ReRAM to MLC NAND to store hot data. When the free space of ReRAM drops below a threshold (for instance, 20% of the ReRAM capacity), eviction is triggered. With this, the page-level migration is dynamically handled which is transparent to the file system. Most existing schemes consider the eviction based on only the state of each node separately. Note, however, that cold and hot data are dispersed in the nodes of Hadoop cluster system. As a result, the performance of the storage system cannot be maximized with this approach. Furthermore, it causes unnecessary waiting time to the users. If several nodes in the HDFS trigger the eviction process simultaneously while many jobs arrive, the users need to wait for the completion of the eviction process.

In this paper, thus, we propose a new scheme of cold data eviction based on the node congestion probability for HDFS. The proposed scheme analyzes the node congestion probability with which the system can decide if the node is available to execute the eviction process or not. In addition, a new page search approach is proposed, which is important for high performance HDFS service. Through comprehensive computer simulation, it is confirmed that the proposed scheme significantly decreases the execution time and recovery time of replication in the HDFS environment in comparison to the random eviction approach, the scheme employed in HDFS, and the CDE algorithm.

The rest of the paper is organized as follows. In Section 2 brief explanation of data replication with HDFS and hybrid SSD are provided. Section 3 presents the proposed cold data eviction scheme, and Section 4 verifies its performance by computer simulation. Finally, Section 5 concludes the paper and suggests the future course of study.

2. Related Work

In this section the previous work relevant to data replication with HDFS and Hybrid SSD are discussed.

2.1. *Data Replication with HDFS*

In order to tolerate failure in cloud computing system, various data replication techniques have been proposed. In the cloud computing environment, data resources are geographically scattered, and thus networking delay has been a major obstacle in rapid data access. Numerous studies have been undertaken to replicate data in several data storages that are physically distributed and as a result reduce the amount of long-distance data transmissions over the network. Fig. 2 describes the structure of data replication with HDFS. The data replication strategies can be categorized by the types, units, and criteria of replication [4]. In terms of replication type, there are two types: static and dynamic.

The former is ineffective for large-scale cloud data service because it statically manages data replication. It is incapable of quickly responding to various network conditions and changes in the data access pattern. For this reason, the studies on dynamic data replication have been actively conducted [5, 6].

[7] suggested replication strategies reducing the network bandwidth and access delay. They also compared the performance of data access patterns

categorized by time and spatial locality. By considering the network capacity and file access pattern, [8] proposed a file replication algorithm improving the performance of basic replication method. Similarly, [6] proposed the Latest Access Largest Weight (LALW) method, which uses data access history in dynamically determining the replication policy by applying greater weight to more recent access. [9] proposed a dynamic optimal replication strategy (DORS) which evaluates the value of a file based on the access history, size, and the network condition in deciding the target files of replication.

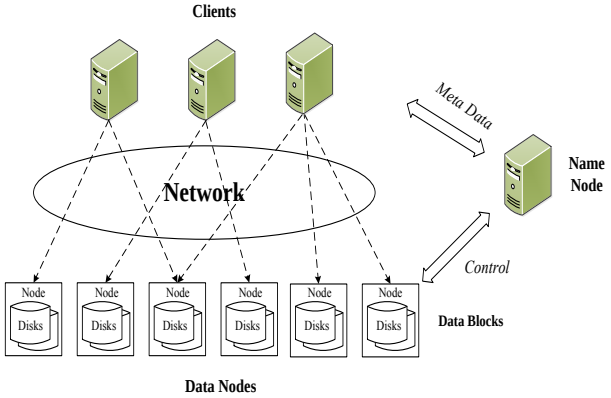

Fig. 2. The structure of data replication with HDFS.

Some researches proposed the strategies for data replication focusing on cost-saving. [10] proposed a service that applies the market economy model to minimize the replication and data access cost on a data grid. [11] suggested a data replication strategy based on a cost-estimate model that considers both the cost of data access and the performance of replication. In order to ensure efficient resource management in an unpredictable data grid environment, [12] developed a model that dynamically selects a resource management strategy that responds to a particular workload type based on the performance history. This model demonstrated that a real-time workload type is an important factor in resource management and the selection of data replication strategy. Similarly, [13] proposed the File Reunion based on Data Replication Strategy for Data Grids (FIRE) scheme, which refers to the file access history of nearby storage to determine data replication, aiming to supply high quality service in the cloud computing environment. A data replication strategy is dynamically selected to provide optimal data service, while the least Locality based Data Replication Strategy (LDRS) scheme turned out to outperform the LRU scheme for sequential access pattern having spatial locality.

[14] suggested a novel cost-effective dynamic data replication strategy named CIR for cloud data centers which applies an incremental replication approach to minimize the number of replicas while meeting the requirement of reliability and cost-effectiveness. Their approach can substantially reduce the data storage cost, especially when the data are stored only for a short duration or have a lower reliability requirement. Nonetheless, their approach is based on only the reliability parameters and pricing model of Amazon S3 which makes it unsuitable for Google cluster of a much higher failure rate than Amazon S3 storage units. Moreover, they did not consider the issue of the trade-offs between cost and performance.

In [15], a dynamic distributed cloud data replication algorithm (CDRM) was proposed to capture the relationship between availability and the number of replica. It maintains the minimum replica for the given availability requirement. The replica placement is based on the capacity and blocking probability of data nodes. Some researches [16] present six different replication strategies for three different access patterns: No Replication or Caching, Best Client, Cascading Replication, Plain Caching, Caching plus Cascading Replication, and Fast Spread. The main aim of these strategies is reduction in access latency and bandwidth consumption.

[17] proposed a centralized data replication algorithm (CDRA) to reduce the total file access time with the consideration of limited storage space of Grid sites. Based on the centralized algorithm, they also designed a distributed caching algorithm wherein the Grid sites react close to the Grid status and make intelligent caching decisions, which can be easily adopted in a distributed environment such as Data Grids. Their approach can reduce the aggregated access delay to data files by at least half of that reduced by the optimal replication solution. The limitation of the algorithm is the consideration of only the access cost.

[18] studied a replication algorithm based on a cost-estimation model, driven by the estimation of the data access gains and the replica's creation and maintenance cost. It allows the grid nodes to automatically replicate data when needed in their Data Grid simulator, GridNet. [19] proposed a dynamic hybrid protocol (DHP) which effectively combines the grid and tree structure so that

the overall topology can be flexibly adjusted using three configuration parameters; tree height, number of descendants and grid depth. It can easily detect read/write conflict and write/write collision for consistency maintenance.

Some of the existing strategies optimize the number of replicas, while others optimize the placement of replicas. Others optimize how often replicas should be updated [20]. The shortcoming of them is that they only consider a restricted set of parameters affecting the replication decision. Furthermore, they only focus on the improvement of the system performance without addressing the energy efficiency issue in data centers.

[21] designed an evolutionary way to decide the optimal replication strategy. In that work, they optimize storage latency and reliability without considering the total energy cost of data center and the issue of load balancing.

2.2. *Hybrid SSD*

The hybrid SCM/MLC NAND flash SSD (hybrid SSD) is a promising solution in boosting the performance of the SSD-based storage while maintaining the cost. Fig. 3 describes the entire structure of the system implemented with SSD. SCM operates as both cache and storage, not simply as a cache/buffer for the NAND flash memory or merely as storage because it is both fast and non-volatile. On the host side, the application layer exists above the operating system, under which the block device layer resides, which in turn is above the interface of the host and storage.

The hybrid SSD includes SSD controller, SCM chip array, and NAND flash memory chip array. Here, MLC NAND flash is preferred to single-level cell (SLC) NAND flash to lower the overall cost of the hybrid SSD. As the brain of the storage system, SSD controller runs complicated algorithms handling the characteristics of erase-before-write and limited endurance of NAND flash. Within the controller, the data management module determines whether the target data are stored in SCM or in NAND flash memory based on the operation on the data and the status of memory. The address translation module manages the logical-to-physical address mapping to provide a logical block interface to the SSD. The wear leveling module guarantees even wear among the storage cells to maximize the system longevity. Additionally, the garbage collection module reclaims free space in the NAND flash when the NAND flash blocks are almost full. Through the controller functions, both the performance and lifetime of SSD can be enhanced. Finally, the error code correction (ECC) module detects and corrects the errors inside the SSD. The role of ECC is becoming more critical as the size scales up, causing degraded reliability. With increased number of memory chips, the overhead on the chip area of SSD grows due to the increased bus area. It is assumed that SCM has much higher reliability than NAND flash memory. Therefore, only a pair of simple BCH ECC encoder/decoder is required for SCM but dozens of low-density parity-check (LDPC) ECC encoder/decoders are required for NAND flash. As a result, the area overhead for SSD controller due to the increasing number of SCM chips is small. In addition, the area for control logic of SCM is negligible. The proposed replication scheme for ReRAM-based storage is proposed next.

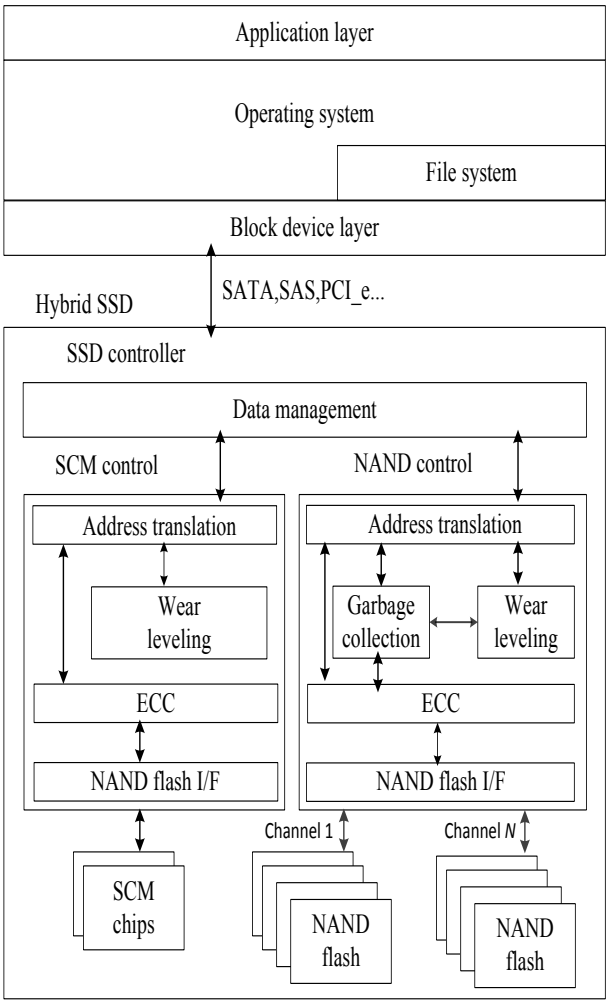

Fig. 3. The overall structure of the whole computer system with SSD.

3. The Proposed Scheme

This section proposes a scheme evicting cold data for the HDFS based on the hybrid SSD. It consists of three parts; eviction of cold data, estimation of node congestion probability used in the cold data eviction, and page search for finding cold/hot pages.

3.1. *System Model*

The proposed scheme targets the development of a replication management approach for HDFS system. The HDFS has numerous similarities with the proprietary distributed file system, Google file system. It consists of a single name node and a set of data nodes. The name node and data nodes are deployed within a number of racks as shown in Fig. 4, each of which has an associated rack number.

Contrary to the existing HDFS, in the proposed scheme the name node mainly manages the namespace of the file system and the location of data pages (the mapping of data pages to data nodes). A file is split into one or more data pages which are dispersed in the data nodes. In Hadoop, the applications are executed in data nodes. When an application needs a data page, it acts as an HDFS client sending a page read (write) request to the name node. The name node finds the requested data node to process the request. Each data node also periodically sends a heartbeat message to the name node to notify its soundness. In HDFS, the number of replicas is set to three, which is also adopted in this paper. By using NRU (Not Recently Used) table, the proposed scheme decides if a data page is hot or cold.

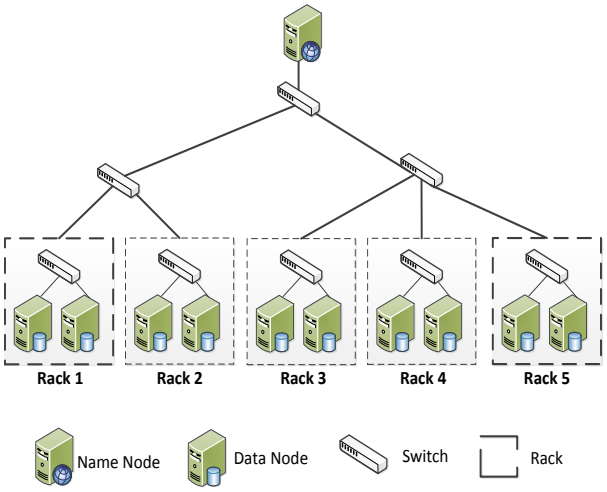

Fig. 4. The structure of HDFS.

The overall process of the proposed scheme is given in Fig. 5. The proposed replication scheme is triggered if the free space of ReRAM of entire nodes drops below a threshold (for instance, 20% of the total ReRAM capacity), the free space of ReRAM of a single node is less than 10%, and the node congestion probability (CP) is lower than the threshold, T_{cp}.

Algorithm 1 : Process of cold data eviction

```
1:  New write data
2:  if Total free space in ReRAM < 20%
3:      Go to Line 6
4:  if Single node free space in ReRAM < 10%
5:      if the node congestion probability > CP_THRESHOLD
6:          if Flag_NRU == 1      (Hit NRU table?)
7:              if R<R_RM
8:                  Add page to cold data eviction list
9:              else
10:                 Go to Line 15
11:             if N_RM == RM_EXECUTION
12:                 Execution of cold data eviction
13:             else
14:                 Go to Line 15
15:         else Search page in ReRAM
```

Fig. 5. The overall process of the proposed scheme.

To judge whether a page is hot or cold, the NRU table is referred. Because the write performance is critical to the NAND-based storage system, the proposed algorithm was designed to optimize the operation of SSD write request rather than read request. Here the page utilization, U, is compared with the utilization threshold, T_u, to choose the least used page for replication. If it is impossible to select a page in ReRAM when cold data eviction is triggered, R_{RM} is reduced to increase the number of candidate pages in ReRAM used in the next eviction process.

Fig. 6 compares the proposed scheme with the existing schemes. In the MLC NAND-only SSD in Fig. 6(a), all pages are stored in SSD, regardless hot/cold, fragmented/not fragmented. In the conventional hybrid SSD (Fig. 6(b)) fragmented pages are sent to ReRAM [22].

Since fragmented cold data decreases free space of ReRAM, fragmented hot data might be stored in MLC NAND. As a result, the SSD performance will be greatly degraded if this situation gets worse or the ReRAM capacity is relatively small. In order to provide high performance under intensive fragmented cold data workloads, a new cold data eviction scheme is proposed. Fig. 6(c) describes that the portions of cold data in ReRAM are evicted to MLC NAND flash. By

dynamically evicting the fragmented cold data pages from ReRAM to MLC NAND, ReRAM can effectively store hot data. Furthermore, fragmentation in MLC NAND is minimized by the eviction of mostly fragmented cold data page.

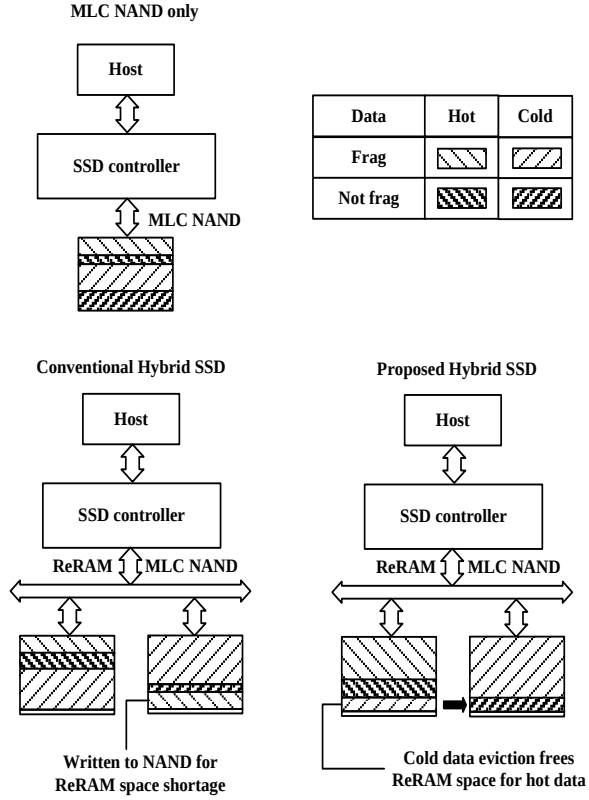

Fig. 6. The comparison of the proposed scheme with the existing ones.

3.2. *Node Congestion*

The key idea of congestion probability based on ECN (Explicit Congestion Notification) is to analyze the distribution of Congestion Experienced (CE) bits from the ECN feedbacks and predict the network state according to the correlation between multiple feedbacks.

One ECN feedback reflects the network state only at one earlier time interval, while a sequence of ECN feedback can indicate the dynamic change of the network state during the past several continuous time intervals. Therefore, we propose to combine the information contained in the CE bits of a sequence of ACK packets together to predict the network state.

Assume that a sender receivers an ECN feedback sequence containing k ECN feedbacks. We calculate CP(t), the congestion probability at the current time t, as

$$CP[i] = \frac{1}{k} * \sum_{i=0}^{k-1} \omega_i * ACK[i].ECN\text{-}Echo \quad (1)$$

where ACK[i].*ECN-Echo* is the value of the CE bit in ACK[i], and w_i are normalized such that $\sum_{i=0}^{k-1} \omega_i = 1$.

The value of CP(t) is calculated using k serial ACKs received until time t, denoted by ACK[i], ACK[i–1],…,ACK[i–k+1], respectively. For the sake of simplicity of presentation, we use ACK[i] instead of ACK[i].*ECN-Echo*.

The weights of the CE bits in different ACK packets are assigned using the *exponentially weighted moving average* method. The ACKs are divided into several segments, and all ACKs in the same segment are assigned a same weight. This is because a large number of ACKs increase the feedback delay, while a single ACK cannot correctly reflect the congestion state. To achieve the tradeoff between sensitivity and stabilization, the k ACKs are divided into n segments, and the ACKs of segment_x are assigned with a weight, w_x. The value of CP(t) can be calculated as

$$CP[i] = \sum_{x=1}^{n} \omega_x * \frac{n}{k} * \sum_{i=\frac{k}{n}(x-1)}^{\frac{k}{n}(x-1)} ACK[i] \quad (2)$$

We calculate CP(t) with moving weighted average value of n samples. Note that the congestion probability based on ECN tracks the change of the CP(t) value to estimate the congestion level. Generally, the newer the ACK packet, the more accurate it is in predicting the current network state. Therefore, a larger weight needs to be assigned to newer ACK. The weights of ACKs in each segment are assigned as

$$\omega_x = \frac{1}{\alpha} * \omega_{x-1}, \quad \alpha \in (0,1) \quad (3)$$

While all the weights satisfy

$$\sum_{x=1}^{n} \omega_x = 1 \quad (4)$$

It is clear that the weight of x^{th} segment, ω_x, is larger than the weight of previous segment, ω_{x-1}, since the value of α is less than 1.

In highly dynamic network, it is very difficult to get accurate traffic state with only a smaller number of

packets. However, using large number of packets will increase the feedback latency. Like the way that TCP uses three duplicate ACKs to reflect packet loss, we choose four ACKs for one segment to track the congestion state. After receiving three duplicate ACK packets, the sender enters the congestion avoidance mode. Similar to this, we choose a period of four ACK packets to compose one segment reflecting the congestion state. Thus, we have

$$k = 4n \tag{5}$$

With Eq. (3), Eq. (4) and Eq. (5), we can rewrite Eq. (2) as

$$CP[i] = \sum_{x=1}^{n} \omega_t * \frac{1}{4} * \sum_{i=4(x-1)}^{4x-1} ACK[i],\ i \in [1,\ k] \tag{6}$$

Eq. (6) can be expanded to

$$\begin{aligned} CP[i] = & \left(\omega_1 * \frac{1}{4} * \sum_{i=0}^{3} ACK[i] \right) \\ & + \left(\frac{\omega_1}{\alpha} * \frac{1}{4} * \sum_{i=4}^{7} ACK[i] \right) \\ & + \cdots + \left(\frac{\omega_1}{\alpha_{n-1}} * \frac{1}{4} * \sum_{i=4(n-1)}^{4n-1} ACK[i] \right) \end{aligned} \tag{7}$$

If CP(t+1) > CP(t), the congestion probability increases; otherwise, the congestion probability decreases. Therefore, the change of CP can properly reflect the change of network state. Defining ΔCP = CP(t+1) − CP(t) − CP(t), we have

From Eq. (8), it is clear that ΔCP is related to only ACK[4i], that is to say, the last ACK of each segment. We can also see that the value of ΔCP lies in a discrete set containing 2^{n+1} elements. If the CE bit of the last ACK in the last segment is marked with 0, the value of ΔCP will be in [−1, 0]. In this case, it can be concluded that the congestion probability is decreasing, which has the same effect of a single ECN feedback. On the contrary, if the CE bit of the latest arriving ACK is 1, the value of ΔCP will be in [−1, 1]. In this case, it cannot be determined whether the network is going to be congested or not. However, ΔCP ≤ 0 always indicates that the network congestion may be relieved regardless of the value of the CE bit of the newly arriving ACK. The value of CP is calculated by the procedure shown in Fig. 7

$$\begin{aligned} \Delta CP = & \frac{\omega_i}{4} * \left(\begin{array}{l} (ACK[4] - ACK[0]) \\ + \frac{1}{\alpha} * (ACK[8] - ACK[4]) \\ + \ldots + \frac{1}{\alpha^{n-1}} * (ACK[4n] - ACK[4n-4]) \end{array} \right) \\ = & \frac{\omega_i}{4} * \left(\begin{array}{l} -ACK[0] + \left(1 - \frac{1}{\alpha}\right) ACK[4] \\ + \left(\frac{1}{\alpha} - \frac{1}{\alpha^2}\right) ACK[8] \\ + \ldots + \left(\frac{1}{\alpha^{n-1}} - \frac{1}{\alpha^n}\right) * ACK[4n] \end{array} \right) \end{aligned} \tag{8}$$

Algorithm 2 : Procedure of calculating CP

```
1:  CP(t+1)=CP(t)
2:  int ECN-Echo=hdr_flags::access(pkt)
    → ECN-Echo()
3:  float ω1, ω2
4:   for i=1 to m−1 do
5:      ACK[i−1]=ACK[i]
6:  end for
7:   ACK[m−1]= ECN-Echo
8:   u1 = ACK[m−1]+ ⋯+ACK[m/2]
9:   u2 = ACK[m/2−1]+ ⋯+ACK[0]
10:  CP(t)= ω1*u1 + ω2*u2
11: return CP(t)
```

Fig. 7. The procedure for calculating the CP.

3.3. *Page Search*

In searching cold pages, three approaches are available as shown in Fig. 8, (a) From the first page, (b) From random position, and (c) From the unsearched page. The numbers in the figure denote the search sequence of ReRAM pages, and the arrow shows the start point of the search.

For the approach of Fig.8 (c), the start point is the cold page in ReRAM found in the previous cold data eviction operation. The page number is stored in the memory, and updated when the cold data eviction process is triggered.

In the proposed scheme the approach of Fig.8 (c) is adopted due to lower search overhead compared with the other approaches. With the adopted search approach, the unsearched region is scanned first. Statistically, the chance of the searched region to contain cold page after the execution of cold data eviction will be relatively low.

By checking the unsearched region first, the overhead of searching can be minimized. We next evaluate the performance of the proposed scheme.

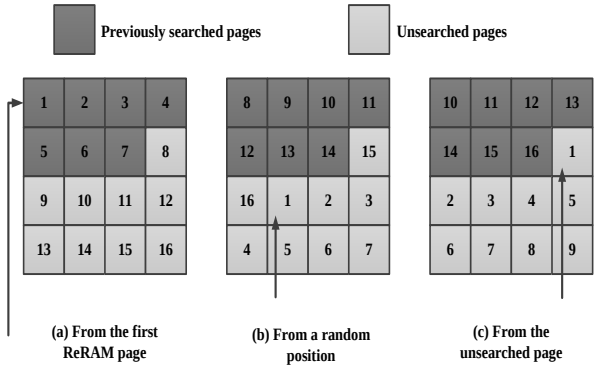

Fig. 8. The approaches for the search of cold page.

4. Performance Evaluation

In the simulation tree structure is assumed as the network topology of the HDFS. To simulate the tree network topology, the nodes are distributed in the racks as follows: there are 100 racks and each rack is equipped with one switch. The 100 racks are randomly distributed over a 100 × 100 unit square plane, and a rack occupies a 10 × 10 square plane. For any two racks, there is no intersection area between them. Among the 100 racks, one is designated as the root rack of all other racks in binary tree topology of the height of 7. After forming the 100 racks in the tree topology, 3,500 nodes are randomly deployed within the 100 racks. For any two nodes in the same rack, their locations are within the square plane occupied by the rack.

Based on the generated network topology, the simulation is performed with the parameter setting summarized in Table I. In each node the available replication space is represented as the maximum number of data block replicas allowed to be stored. It is set by randomly selecting a number between 0 and 50

In a simulation run, some nodes are randomly selected which frequently need to write data to their disks. Therefore, many nodes can concurrently issue the replication requests. The number of requesting nodes are changed from 500 to 2,500 to evaluate the effectiveness of the proposed scheme with different load conditions. For the settings of the read, write and erase latency, the NRU table and eviction list are used in addition to the parameters shown in Table I.

Table 1. MLC NAND/ReRAM Specification and Used Table

	MLC NAND	ReRAM
Read latency (max.)	85μs/page	100ns/sector
Write latency(typical)	Lower page 400μs Upper page 2800μs	(Set/Reset) 100ns/sector
Erase latency (typical)	8500μs/block	-
Access unit	Page(16KiB)	Sector(512B)
NRU	2000 entries	
Eviction list	1500 entries	

In addition, dynamic workloads are also considered for the disk queue and switch link queue. Whenever one or more nodes concurrently issue the replication requests, block read-write operations are needed between multiple node pairs. The number of concurrent block read-write operations is randomly set between 0 and 50. Due to concurrent block read-write operations, a number of operations need to be handled with the disk queue and switch link queue before issuing the replication requests.

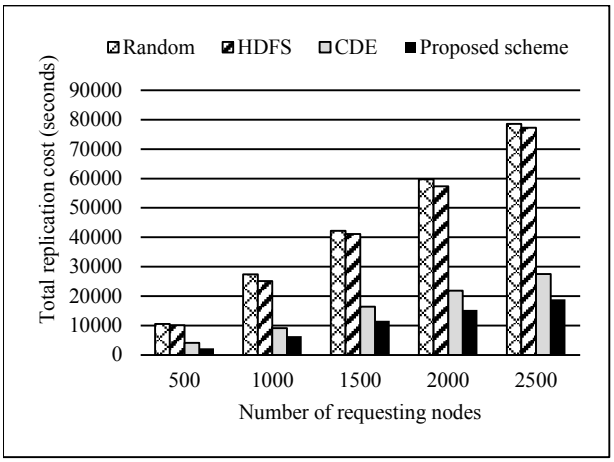

Fig. 9. The comparison of total replication costs.

Fig. 9 shows the total replication cost for different numbers of requesting nodes. As seen from Fig. 9, the total replication cost of all the schemes increase with the number of requesting nodes. Basically, the HDFS replication scheme adopts the random scheme to place the replicas of a data block, but with additional consideration on the possible rack failure. Therefore, the total replication cost of the HDFS replication scheme is

similar to that of the random replication scheme. Notice that the proposed scheme consistently reduces the replication cost compared to the existing schemes. In case of the requesting nodes of 500, the improvement with the proposed scheme is not significant. However, as the number of requesting nodes increases, the reduction becomes quite substantial.

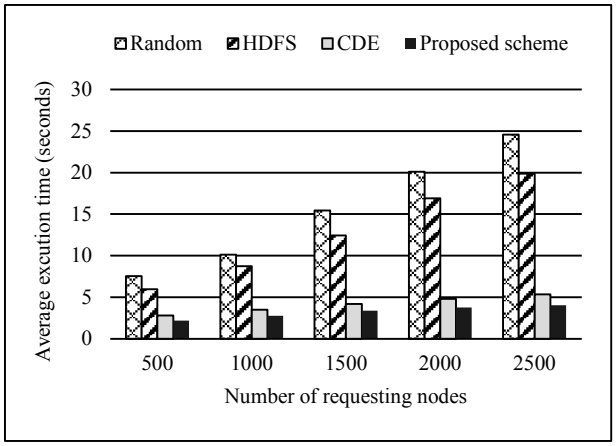

Fig. 10. The comparison of average execution times.

Fig. 10 shows that the average execution time follows almost the same trend as the replication cost of Fig. 9. The proposed scheme outperforms the other schemes, especially with relatively large number of requesting nodes.

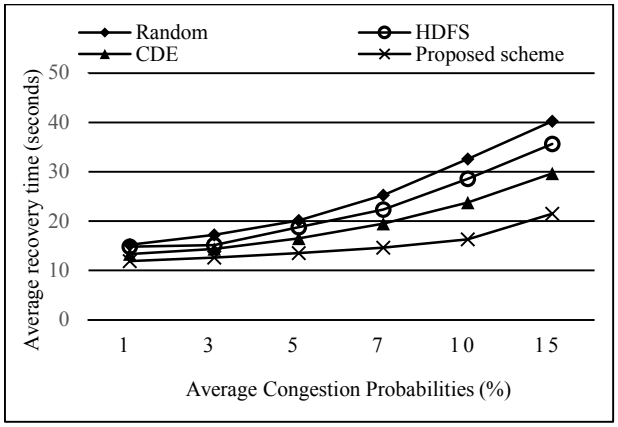

Fig. 11. Average recovery time with different congestion probabilities.

The average recovery time is shown in Fig. 11 for different congestion probabilities and 1,000 requesting nodes. As can be seen from the figure that the proposed scheme requires the smallest average recovery time, which is about one-fifth of that of the HDFS. And, as the congestion probability grows, the improvement gets larger.

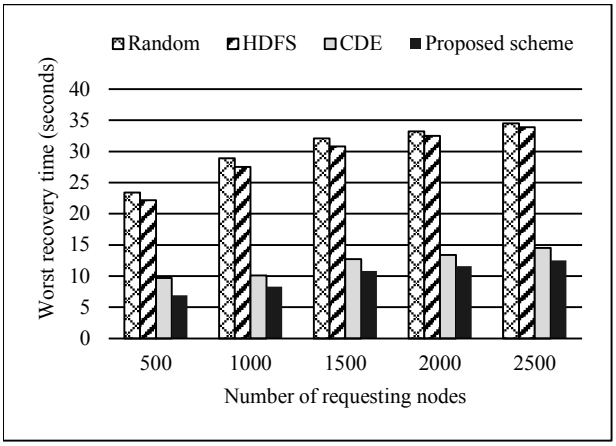

Fig. 12. The comparison of worst recovery times.

Fig. 12 shows the recovery times in the worst case. The worst recovery time denotes the largest access time to retrieve a failure-free data replica, which is measured by accumulating the access time of all replicas of a data block. As shown in the figure, the proposed scheme decreases about 70% of the worst recovery time of the HDFS. Compared to the CDE algorithm, it is about 20%.

5. Conclusion

In this paper we have proposed a novel cold data eviction scheme which conspicuously considers the congestion status of the nodes. The proposed scheme analyzes the node congestion probability by which the system can decide if the nodes are available to trigger the cold data eviction process or not. Also, the proposed scheme classifies the data as hot or cold, and decides the storage location of them using NRU algorithm. Computer simulation reveals that the proposed scheme significantly decreases the recovery and execution time in the HDFS environment in comparison to the previous schemes. In particular, with relatively high congestion probability, the proposed scheme substantia excels the existing schemes.

As the future course of study, we plan to extend the proposed scheme to effectively decide the number and location of the replications of SSD blocks in the HDFS environment. Also, the design parameters will be determined through comprehensive modeling of the proposed scheme on the target performance measures. There exist many storage nodes in a cloud computing system, and energy efficiency is an important issue. The

proposed scheme will be investigated further to maximize the energy efficiency.

Acknowledgements

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2012R1A1A2040257 and 2013R1A1A2060398), the second Brain Korea 21 PLUS project, ICT R&D program of MSIP/IITP (1391105003), and Samsung Electronics (S-2014-0700-000). Corresponding author: Hee Yong Youn.

References

1. K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The Hadoop Distributed File System," *in Proceedings of the IEEE 26th Symposium on Mass Storage Systems and Technologies* (*MSST* 10), pp.1-10, May 2010.
2. C. Metz. Flash drives replace disks at Amazon, Facebook, Dropbox. http://www.wired.com/wiredenterprise/2012/06/flash-data-centers/all/, June 2012
3. C. Sun, K. Miyaji, K. Johguchi, K. Takeuchi, "A high performance and energy-efficient cold data eviction algorithm for 3D-TSV hybrid ReRAM/MLC NAND SSD," *IEEE Transactions on Circuits and Systems*–I:*Regular papers*, vol. 61, no. 2, Feb. 2014.
4. S. Venugopal, R. Buyya, R. Kotagiri, "A Taxonomy of Data Grids for Distributed Data Sharing, Management and Processing," ACM Computing Surveys, Vol. 38, pp. 1-53, 2006.
5. K. Sashi, A.S. Thanamani, "Dynamic Replication in a Data Grid Using a Modified BHR Region Based Algorithm", Future Generation Computer Systems, Vol. 27, No. 2, pp. 202-210, 2011
6. R.S. Chang, H.P. Chang, "A Dynamic Data Replication Strategy Using Access-Weights in Data Grids," Supercomputing, Vol. 45, No. 3, pp. 277-295, 2008.
7. K. Ranganathan, I. Foster, "Design and Evaluation of Dynamic Replication Strategies for a High-Performance Data Grid," International Conference on Computing in High Energy and Nuclear Physics, 2001.
8. H. Sato, et al., "Access-Pattern and Bandwidth Aware File Replication Algorithm in a Grid Environment," International Conference on Grid Computing, pp. 250-257, 2008.
9. W. Zhao, et al., "A Dynamic Optimal Replication Strategy in Data Grid Environment", International Conference on Internet Technology and Applications, pp. 1-4, 2010.
10. M. Carman, K. Stockinger, "Toward an Economy-based Optimization of File Access and Replication on a Data Grid," International Symposium on Cluster Computing and the Grid, pp. 340-345, 2002.
11. H. Lamehamedi, et al., "Data Replication Strategies in Grid Environments", International Conference on Algorithms and Architectures for Parallel Processing, pp. 378-383, 2002.
12. B.D. Lee, J.B. Weissman, Y.K. Nam, "Adaptive Middleware Supporting Scalable Performance for High-End Network Service," Network and Computer Applications, Vol. 32, pp. 510-524, 2009.
13. A.R. Abdurrab, T. Xie, "FIRE: A File Reunion Based Data Replication Strategy for Data Grids," International Conference on Clustering Computing and the Grid, pp. 215-223, 2010.
14. W.H. Li, Y. Yang, D. Yuan, A novel cost-effective dynamic data replication strategy for reliability in cloud data centres, in: IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing, 2011.
15. Q. Wei, B. Veeravalli, B. Gong, L. Zeng, D. Feng, CDRM: a cost-effective dynamic replication management scheme for cloud storage cluster, in: Proc. 2010 IEEE International Conference on Cluster Computing, Heraklion, Crete, Greece,September 20–24, 2010, pp. 188–196.
16. K. Ranganathan, I.T. Foster, Identifying dynamic replication strategies for a high-performance data grid, in: Proc. Second Int'l Workshop Grid Computing (GRID), 2001.
17. D.T. Nukarapu, B. Tang, L.Q. Wang, S.Y. Lu, Data replication in data intensive scientific applications with performance guarantee, IEEE Trans. Parallel Distrib. Syst. 22 (8) (2011) 1299–1306.
18. H. Lamehamedi, Z. Shentu, B. Szymanski, Simulation of dynamic data replication strategies in data grids, in: Proc. 12th Heterogeneous Computing Workshop (HCW2003) Nice, France, April 2003, IEEE Computer Science Press, Los Alamitos, CA, 2003.
19. S.C. Choi, H.Y. Youn, Dynamic hybrid replication effectively combining tree and grid topology, J. Supercomput. 59 (2012) 289–1311.
20. M. Tu, T. Tadayon, Z. Xia, E. Lu, A secure and scalable update protocol for P2P data grids, in: 10th IEEE High Assurance Systems Engineering Symposium, Texas, 2007, pp. 423–424.
21. O.A.-H. Hassan, L. Ramaswamy, J. Miller, K. Rasheed, E.R. Canfield, Replication in overlay networks: a multi-objective optimization approach, in: Collaborative Computing: Networking, Applications and Worksharing, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol. 10, 2009, pp. 512–528.
22. H. Fujii, K. Miyaji, K. Johguchi, K. Higuchi, C. Sun, and K. Takeuchi, "11 performance increase, 6.9 endurance enhancement, 93% energy reduction of 3D TSV-integrated hybrid ReRAM/MLC NAND SSDs by data fragmentation suppression," in *Symp. VLSI Circuits Dig.Tech. Papers*, Jun. 2012, pp. 134–135.