INFORMATION
Volume 18, Number 9, pp.3961-3969

ISSN 1343-4500

An Efficient Backup System using Hot-Cold Data Classification in a Distributed Memory System

Woochur Kim*, Donghee Min*, Joonhyouk Jang** and Jiman Hong***

* *Department of Computer Science and Engineering, Soongsil University, Seoul, Korea*
E-mail: {manwoo425, dhmin118}@gmail.com

** *Department of Computer Science and Engineering, Seoul National University, Seoul, Korea*
E-mail: jhjang0@gmail.com

*** *School of Computer Science and Engineering, Soongsil University, Seoul, Korea*
E-mail: jiman@ssu.ac.kr

Abstract

As the IT technology advances, data processing system is required to handle and process large amounts of data. For that reason, the In-Memory system is being used which saves and manages all the data on the main memory. That system uses fault tolerance technique for data persistence. But those fault tolerance techniques lead to performance degradation of In-Memory system. So it is necessary to develop backup technique without impact on the performance. In this paper, we propose an algorithm to classify the data into Hot and Cold data in consideration of the data usage characteristics in the In-Memory system and compound backup technique to ensure data persistence. Experimental results show that the proposed technique increases data persistence and reduce impact on the processing performance.
Key Words: Distributed Memory System , Backup Technique, Hot-Cold data.

1. Introduction

Many previous data process systems usually store and maintain most of the data on a hard disk. Storing and maintaining data in a hard disk is a major cause of low computing performance because a hard disk is relatively slower than the processing speed of memory. Due to the continuous advancement of IT technology, the unit capacity of hard disks and memory have decreased. This is why data in typically stored in memory rather than on a hard disk, which has a slow processing speed [2][4]. Storing and maintaining data in the memory enables high speed processing in real time for massive data due to the big data era with high speed. However, if the memory has no power supply, the stored data might be lost due to its volatile characteristic. Due to its volatility, there is a risk of losing all the data when an unexpected fault occurs in the in-memory system. Therefore, a fault tolerance method is necessary to ensure continuous service and to prevent the loss of data due to unexpected failure of the in-memory system [3]. A number of fault tolerance methods can be used such as the logging and check-pointing method that saves identical data in a system, and the overlapped and duplexing method that stores data in another system, etc. The logging and check-pointing method places a high load on the system; it therefore reduces the data process

performance of the system. On the other hand, the duplexing method incurs a high construction cost, since an extra system is needed. In this paper, we classify hot data and cold data by utilizing the characteristic of in-memory system data using a hot-cold classification method. The suggested method will be applied to a real dispersion memory system and the data process performance and data permanency of the in-memory system will be compared to those of the compound backup method through experiments.

This paper consists of the following sections. Chapter 2 explains the in-memory system, backup methods, and the problem of the existing backup method as reported in the existing research. Chapter 3 discusses the implementation of the hot-cold classification method in an in-memory system and the implementation of the compound backup method, in which the use characteristics of each data is considered. Chapter 4 evaluates performance using data permanency analysis and compares the process performance of the existing methods through many experiments to determine the suitability of the proposed method as suggested in chapter 3. Finally, the paper ends with a conclusion.

2. Related Work

2.1 In-Memory System and Backup Technique

An in-memory system is a system that uses the main memory for executing applications and as the major data storage. In the in-memory system, delays of disk operation do not occur when processing data, so it is faster than an on-disk system. Especially, it is effective in a variety of fields such as real time finance trading, online gaming, mobile applications, session management from communication companies, social media, etc., whereby a massive amounts of data are used. To save such massive amounts of data effectively in the limited storage of the in-memory system, a simple atypical database system is more suitable than the existing compound relational database system. The name of the atypical data model, 'NoSQL' refers to 'Not Only SQL' or 'Not Relational'. It has better horizontal expandability than the relational database system, and is thus suitable for massive server construction. The NoSQL model can be classified into 4 types by data model [1]. This paper only discusses the Key-Value Store, which manages all data in the memory and is suitable for massive data processing. Therefore, a memory system should be provided with a backup method due to the volatility of data. To minimize data loss and prevent performance drop caused by the disk I/O, many studies are currently being conducted. The most popular backup methods are the logging and check-pointing methods.

The logging method writes changes of the stored or be-stored data onto a hard disk. It is therefore possible to restore data in an in-memory system through recording logged data on a

hard disk when an unexpected failure of the system occurs. The logging method has the advantage of being able to minimize data loss by recording changes in data on a hard disk. However, its disadvantage is a lower data process speed due to the system load caused by hard disk I/O operation when data changes. Also, it involves unnecessary writing execution, because all changes of data are recorded onto a hard disk. This incurs unnecessary system load and waste of hard disk space. To prevent the unnecessary waste of disc space due to logging a backup file, many systems execute the re-organization of the logging backup file that is separated from the main process. During the re-organization of the logging data file, the system load is increased.

The checkpoint cycle strongly influences the process performance and data loss of an in-memory system. The checkpoint cycle records all the stored data of the in-memory system onto a hard disk. However, data can be lost during the checkpoint cycle. While the loss can be reduced by shortening the cycle, this causes significant system load due to the hard disk I/O process. The check-pointing method is not suitable for a system that has frequent writing processes due to its cycle. Even when data does not change, the check-pointing cycle processes hard disk writing. A reading delay time therefore occurs because of the system load. Basically, some of the hard disk writing process will be unnecessary because of the recoding of the same data as the existing data.

The logging method records data changes, so it is suitable for a system that has frequent data change. However, it incurs unnecessary waste of space on the backup data file if it records changes of identical data. To prevent unnecessary waste of space, the logging method performs re-organization of the logging backup file. The re-organization process of the logging backup file also causes system load, similarly to the check-pointing method. In other words, the system data process performance will be reduced due to the additional data writing process.

To analyze the problems that arise when using the logging and check-pointing methods that do not consider data characteristics, we conducted to check the data process performance of the in-memory system due to the reading and writing ratio. Experiments were conducted on different cases, including the check-pointing method, the logging method, using both methods, and using neither method. To compare data process performance, the ratios of the reading and writing process were divided into 2:8, 5:5, and 8:2 for 1,000,000 transactions and the changes of process performance were then analyzed. The check-pointing method showed better performance than the logging method, but there was a risk of losing data. Conversely, the

logging method had a low risk of losing data, but the performance was poor. Therefore, this paper applies a data backup method while considering data use characteristics rather than applying each method separately. The method incorporates the characteristics of low system load from the check-pointing method and low risk of losing data from the logging method.

3. Compound Backup Technique

3.1 Hot-cold data classification

This chapter will discuss the proposed compound backup method that resolves the low performance and data loss problem from the existing backup method. The hot-cold data classification method that considers data using the characteristic will be explained and the compound backup method with the hot-cold data classification method will also be explained. Fig. 1 shows the compound backup of the in-memory system to which the hot-cold data classification method is applied. Hot data and cold data are managed by their own bitmap. A different backup method is applied to each data to decrease data loss and system load.

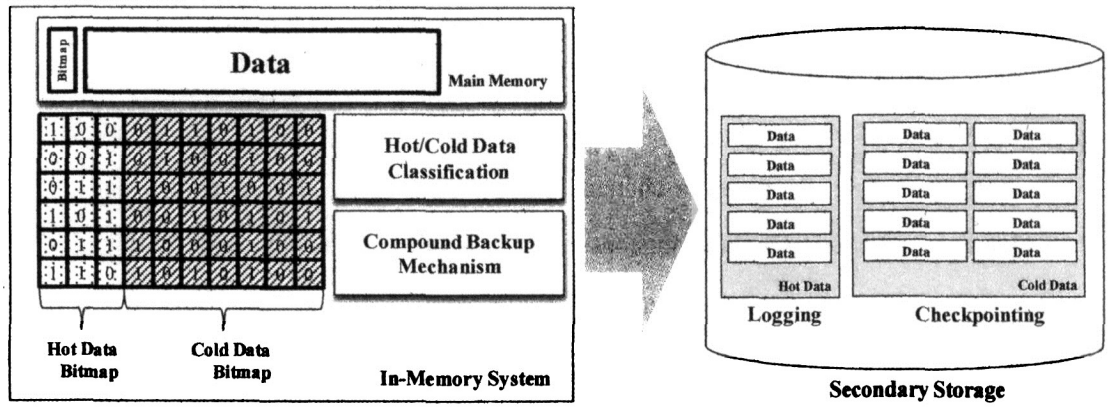

Fig. 1. Overall system components of compound backup mechanism

The hot-cold data classification method classifies hot data and cold data according to the data use characteristic [7]. The data use characteristic is defined, whether the data is usually used for writing or the reading process. Hot data is data mainly used for the writing process, while cold data is usually used for the reading process. If data cannot be classified due to its frequent use for writing and reading, it is classified as warm data. Warm data is classified into hot or cold data, depending on the standard of classification. The hot-cold data classification method in this paper uses a hot-cold classification algorithm using a certain cycle.

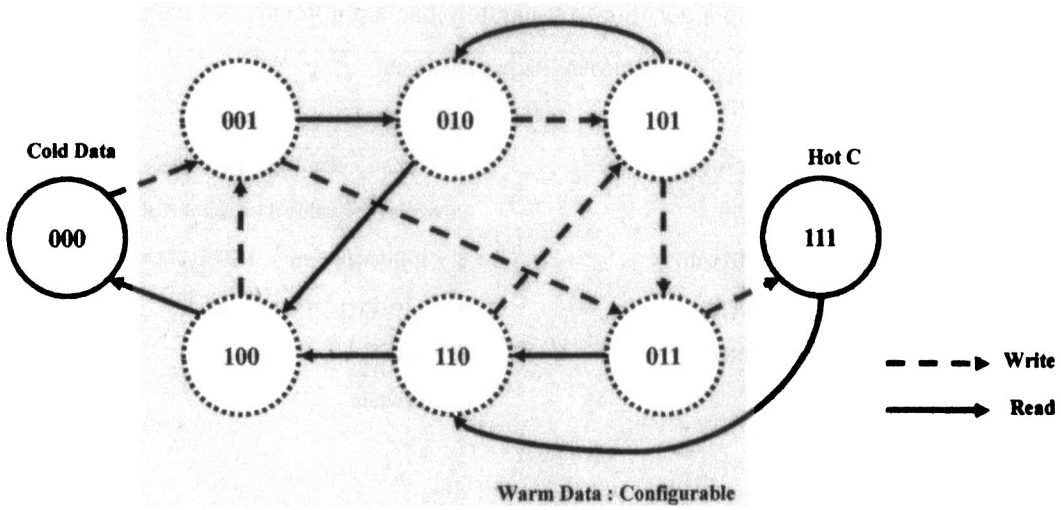

Fig. 2. Bit change as data processing characteristics.

Each data can be classified into hot data or cold data using the hot-cold data classification algorithm by recording characteristic information. Each bit contains a weight according to time flow, so it is possible to classify hot or cold data. Fig. 2 shows the bit change with the reading and writing process. It is possible to classify hot or cold data by using the recorded bit change information.

3.2 Compound Backup Technique

The existing backup method has a trade-off relationship between data loss and performance according to the selection of the backup method. The proposed compound backup method decreases data loss and focuses on the improvement of performance of the in-memory system.

Through the hot-cold data classification method, classified hot data and cold data have their own characteristics. Hot data is used frequently, so the current stored value has a high possibility of changing with time. If there is data change, the backup file will no longer be valid. Therefore, applying the logging method can minimize data loss by renewing the backup file due to data change. Cold data is also frequently read, so it has a low possibility of changing its stored values with time. Since data changes occur frequently, the valid time of the backup file will be increased. Therefore, the check-pointing method should be applied to minimize the unnecessary renewal process of the backup file. Hot-cold data from the hot-cold data classification method is applied with the backup method that considers each characteristic. The compound backup algorithm is shown in Table [1]. Every checkpoint cycle and logging rewrite, it check bitmap of all the data and apply proper backup method.

Table 1. Compound backup algorithm to separately backup hot and cold data using bitmap

Compound Backup Algorighm	
Checkpointing	Logging
/*this function operates every checkpointing cycle*/ 1: Function backupCheckPointing(){ 2: **while** key-value data is not NULL **do** 3: if Bitmap of key is COLD_DATA 4: backup data as Check pointing 5: **end if** 6: **end while** 7: }	1: Function backupLoggingRewrite() { 2: **while** key-value data is not NULL **do** 3: if Bitmap of key is HOT_DATA 4: backup data as Logging 5: **end if** 6: **end while** 7: }

3. Results and Discussion

3.1 Properties of the sequence selection

To evaluate the proposed compound backup method, an experiment environment is constructed and the tests are executed. The YCSB benchmark tool from Yahoo was used for measuring the amount of data processing in the in-memory system [8]. YCSB can change the process frequency of reading and writing, and can analyze the average amount of data to be processed and the delayed section, etc. Data consists of a total of 1,000,000 transactions. YCSB also differentiates the ratios of reading and, which are the same as those obtained from the evaluation of the existing method. A comparison of the amount of data processing is then made between the proposed compound backup method and the existing backup method. Also, an evaluation is performed of the size of the backup file from the compound backup method and data restoration. The in-memory system is built with three distributed server and key distribution layer and the open source Redis is used for the in-memory[9].

3.2 Result

The result from the comparison of the compound backup method proposed in this paper and the existing backup method is as follows. The performance comparison is divided into frequency of reading and writing as well as the amount of data processing. Fig.3 shows a comparison of the data process performance for each backup method.

The compound backup method has a high amount of processing compared to the logging method. On the other hand, it has a low amount of processing compared to the check-pointing

method. Fig.4 shows the size of the backup files for the compound backup method and the existing backup method. The existing method operates for each data independently, so it contains duplication of data. Hence, the compound backup method applies an extra backup method for the same data, so the size of the file will be around 50% of the previous file.

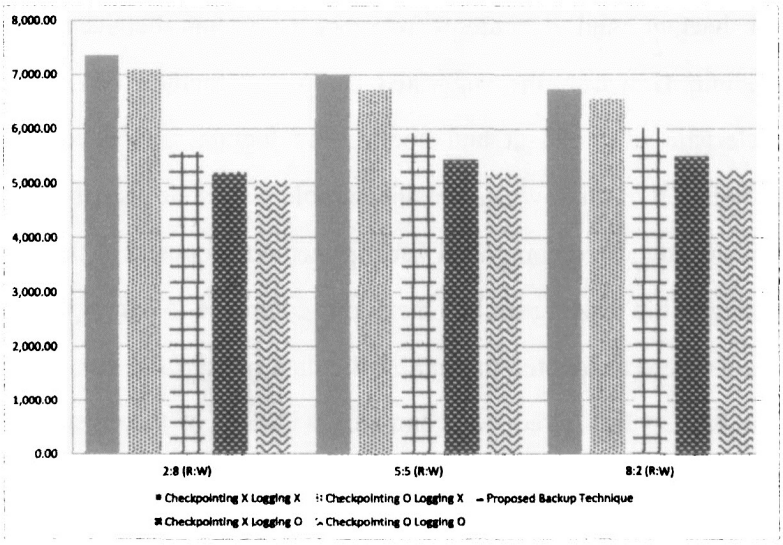

Fig. 3. Performance of proposed technique and previous backup methods

Executing the existing backup method and compound backup method on the same data will decrease the I/O time of the entire disk. The reading process usually uses cold data, so the check-pointing period will be long and the valid period of the check-pointing file will be extended. Also, the check-pointing file recording time will be decreased since only the cold data is recoded; the re-organization of the logging file therefore only focuses on hot data. The disk I/O time will then be reduced.

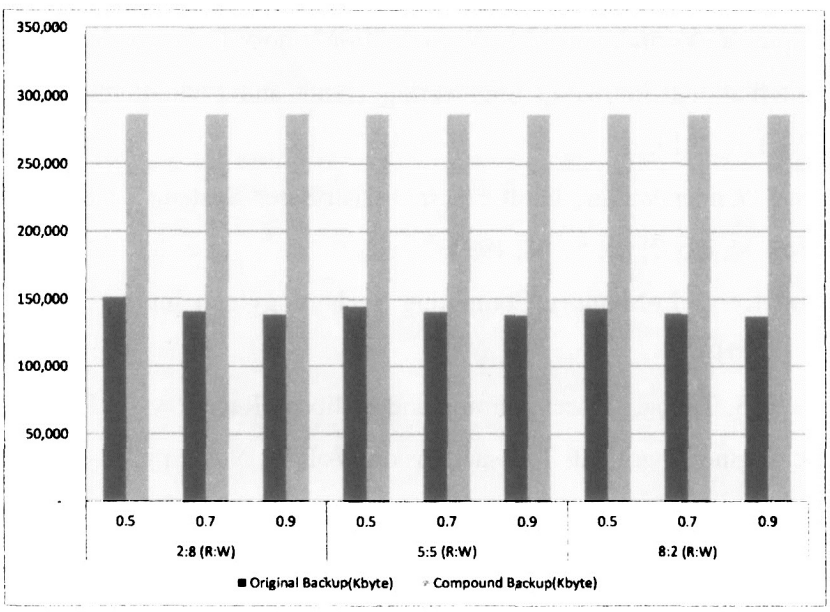

Fig. 4. Backup file size of compound backup technique

This paper suggests a compound backup method to prevent data loss of the in-memory system. The compound backup method uses the hot-cold data classification method, which considers the data use characteristics and applies a different backup method to each data. The existing backup method does not consider the data use characteristics, so it involves a duplication of backup, and a tradeoff relationship occurs between the data loss and processing amount. However, the suggested compound method considers the reading or writing characteristic of the data and applies the logging and check-pointing methods, enabling the resolution of the low performance problem without loss of data. To evaluate the performance of the suggested compound backup method proposed in this paper, the amount of data processing of the existing backup method is compared to that of the proposed method. Also, experiments were conducted to analyze the data backup file size and data restoration. The size of the backup file is reduced compared with that of the existing methods and the data loss issue can be resolved.

4. Acknowledgments

This work was supported by the ICT R&D program of MSIP/IITP. [10043896, Development of virtual memory system on multiserver and application software to provide realtime processing of exponential transaction and high availability service].

References

[1] R. Cattell, "Scalable SQL and NoSQL Data Stores", ACM SIGMOD Record, Vol. 39, No. 4, pp. 12-27, 2010.

[2] M.K. Gupta, V. Verma, and M.S. Verma, "In-Memory Database Systems – A Paradigm Shift", International Journal of Engineering Trends and Technology (IJETT), Vol. 6, No. 6, pp. 333-336, 2013.

[3] F. Cristian, "Understanding Fault-Tolerant Distributed Systems", Communications of the ACM, Vol. 34, No. 2, pp. 56-78, 1991.

[4] Youngsoo Koo, "In-Memory Computing Technology and future, Technology Inside G CNS R&D, 2013.

[5] R. Koo and S. Toueg, "Checkpointing and Rollback-Recovery for Distributed Systems", Software Engineering, IEEE Transactions on, Vol. 13, No. 1, pp. 23-31, 1987.

[6] J. Zhao, S. Li, D.H. Yoon, Y. Xie, and N.P. Jouppi, "Kiln: Closing the Performance gap between Systems with and without persistence support", Proceedings of the 46th Annual IEEE/ACM International Symposium on Microarchitecture, pp. 421-432, 2013.

[7] J.J. Levandoski, P. Larson, and R. Stoica, "Identifying Hot and Cold Data in Main-Memory Databases", Data Engineering (ICDE), IEEE 29th International Conference on, Brisbane QLD, pp. 26-37, 2013.

[8] B.F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears, "Benchmarking cloud serving systems with YCSB", Proceedings of the 1st ACM Symposium on Cloud Computing, pp. 143-154, USA, 2010.

[9] Han, J., Haihong, E., Le, G., and Du, J., "Survey on NoSQL database", Pervasive computing and applications (ICPCA), 6th international conference on, IEEE, pp. 363-366, 2011.

[10] Schindler, Jiri. "Profiling and analyzing the I/O performance of NoSQL DBs.", ACM SIGMETRICS Performance Evaluation Review. Vol. 41, No. 1, ACM, 2013.

[11] Holt, Brandon, et al. "Claret: Using Data Types for Highly Concurrent Distributed Transactions." Proceedings of the Workshop on Principles and Practice of Consistency for Distributed Data (PaPoC), 2014.

*Corresponding author: Jiman Hong, Ph.D.
School of Computer Science and Engineering,
Soongsil University,
369 Sangdo-Ro, Dongjak-Gu, Seoul, 156-743, Korea
E-mail: jiman@ssu.ac.kr