INTERNATIONAL JOURNAL OF CIRCUITS, SYSTEMS AND SIGNAL PROCESSING
DOI: 10.46300/9106.2021.15.183 Volume 15, 2021

Reused Shadow Recovery Scheme for Flash-based Edge Gateway Servers

Siwoo Byun,
Dept. of Software, Anyang University
22, Samduk-ro, 37th street, Anyang-shi, 137-060
South Korea

Received: May 20, 2021. Revised: October 27, 2021. Accepted: November 10, 2021.
Published: December 6, 2021.

Abstract—Edge computing refers to decentralized computing technology to reduce cloud computing's overload or security problems that redirect local data to a central data center. Edge computing is emerging as a technology that complements cloud computing in an IoT environment where huge amounts of data are generated in real time. Recently, solid state drives using flash memory have recently been recognized as a suitable storage for massive IoT data services. In this study, we propose a new data recovery scheme based on shadow paging using flash memory for effective and safe data services in IoT edge gateways. The proposed scheme recycles invalidated old data blocks that are discarded when new data is stored. Thus, The proposed scheme minimizes the burden of additional storage space required to traditional shadow paging schemes, and reduces I/O performance degradation. Simulation results show that the space gain of the proposed scheme reaches even to 29%.

Keywords—IoT Edge computing, Flash-based System, Data Recovery, Flash Memory, Reused Shadow

I. Introduction

Recently, IoT(Internet of Things)[1,2,3] sensor network has received significant attention in smart system areas. Small IoT devices can be designed with on-board calculations, wireless communications and sensor detection abilities (Figure. 1). Recent work has also begun exploring the potential applications for measuring various IoT environments.

Figure 1. Examples of IoT Sensor Devices

The amount of data generated by the IoT or mobile devices is increasing by a large margin. Such IoT devices include sensors, smart phones and wearables, and have limited computing resources and battery energy. Although the cloud is a server with high scalability and superior resources, it is very far away from most end-users, and the movement of IoT data is putting a heavy burden on network links. To cope with these important IoT issues, the model of edge computing was proposed[4,5,6].

In the edge computing environment, useful computing resources are located on the edge of the network and very close to end-devices such as IoT sensors. Therefore, IoT computing resources should be placed close to end-devices to reduce data traffic and latency. That is, a lot of smart services can be provided in IoT edge gateway to process and store data close to end-devices that generate IoT big data. The edge gateway serve as relay agent to the cloud and can also be used for very low-cost hardware (table 1).

Table 1: Example of IoT Gateway Device Specification

Items	Specification
System Platform	Raspberry Pi3
System OS	Raspbian
Processor/Core	1.2GHz-64bit/4
Main Mem.	1GB
Sensors	Humidity(%).,Temperature(‘c), Noise(dB)
Software	JAVA, Android-Studio, bmx

Flash memory has become a critical storage component in building embedded systems such as smart IoT gateway because of its non-volatile, shock-resistant, and power-economic nature. Its density and I/O (Input Output) performance have been improved to a level at which it can be used not only as a main storage for portable computers but also as a mass storage for general computing systems. Although flash memory is not as fast as RAM (Random Access Memory), it is a hundred times faster than a hard disk in read operations.

II. Motivation

2.1 Components of IoT network environment

In general, an IoT network consists of three components: a sensor device, an IoT gateway, and a cloud network, each meaning a data source, a data communication network, and data processor[4,7].

1) Sensor device: Many sensors are placed in wide areas of IoT environment. These sensors produce huge volume of measured data which is a core part of IoT services. The device serves as a human-computer interface that delivers users' requirements to IoT network. These sensors and devices are all be interconnected so that they send sensor data and provide various IoT application services.

2) IoT gateway: IoT gateway collects measured data from sensor devices and forwards it to cloud servers. The sensor devices need to preprocess measured data before they send the data to cloud servers. For example, IoT gateway performs preprocessing of measured data to reduce data redundancy and unnecessary communication overhead.

3) Cloud network: In general, cloud servers have enough resources such as CPU, memory and storages to support IoT applications. The cloud server receives sensor data and user requirements and sends the service results back to the end user after required data processing.

2.2 Reliable Data Management for IoT Edge Gateways

The data recovery means the ability to restore a database to its pre-failure state in the event of a disaster failure, such as physical breakdown, or a non-disaster failure in which a consistency is destroyed by wrong operation. First, disaster-oriented failures are usually recovered using historical copies copied to storage devices such as hard disks. Second, user-oriented failures are usually recovered by redo/undoing some operations to the nearest stable data state from the time the transaction fails. In a database system, data managers(DM) are linked to the bottom of the transaction manager(TM), which accesses actual storage device through an internal recovery manager(RM), so the recovery functionality should always work smoothly. Thus, the database management system must maintain information about data changes during the transaction processing [8].

The techniques for recovering data from non-disaster failures, such as failed transactions, include immediate update-in-place approaches[8,9] and shadow paging approaches[10,11]. In update-in-place approaches, a modified data item is written directly to the same storage location as hard disk. On the other hand, the shadow paging techniques store newly stored data items in different locations on disk storage, allowing multiple copies to exist on the same data item. The value before the data item is modified is referred to as "before image" and the new value after the modification is referred to as "after image". The shadow paging techniques store both previous and subsequent values on disk, so unlike the immediate renewal techniques, logs do not need to be maintained for recovery.

In general, instant renewal techniques have the advantage of consuming less storage space compared to shadow paging techniques, while overhead resulting from redo/undo log storage commonly results in system performance degradation. The shadow paging techniques store revisions in different areas until the transaction is successfully completed, and at completion point, the revisions are reflected to the database. Therefore, there is no dirty page, no redo/undo computation of transactions, so there is no burden on the logs involved, and recovery failure algorithm is very simple.

On the other hand, a lot of storage space is required to store the shadow pages, which contribute to increase system overhead for efficient space management. In addition, the pages of the updated data are frequently repositioned on disk space, resulting in disk I/O performance degradation as the associated pages are distributed without concentration [12].

Due to the rapid development of flash memory technology, it has been in the spotlight as a good storage memory. In addition, there is enough space for shadow pages due to the large capacity of recent flash memory technology. However, unlike traditional storages such as hard disks and random access memories, specialized data processing techniques should be developed by considering the characteristics of flash memory which is impossible to write in place and takes much more time for write and erase operations. In this paper, considering the characteristics of these flash memories, we propose a new data recovery technique using flash memory and shadow pages for effective and safe recovery in IoT edge gateways.

III. Proposed Work

3.1 Flash Memory Database Model

Proposed flash memory data management model is s shown in Figure 2. Flash memory database manager handles flash data operations from start to commitment and flash memory page manager handles mapping table. Flash segment manager comprises four distinct processes: a collector, a cleaner, a cycle leveler, and an allocator.

Allocator handles the set of free block from which it can satisfy new requests. The cycle leveler is a process that allows data to be written evenly across the device, which is essential for maximizing the capacity and longevity of the memory. Finally, the collector is responsible to clean the outdated data on flash memory so as to reduce the overhead of the cleaner.

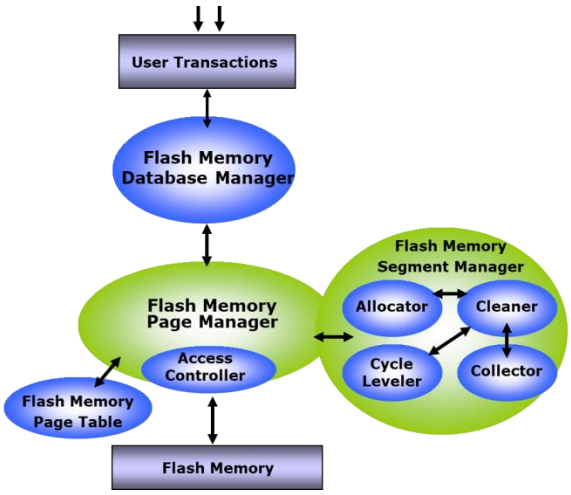

Figure 2. Flash Memory Data Management

To improve data processing performance, Index techniques such as hash tables and b-tree search index were proposed. However, since b-tree index has a very high management cost due to severe overwrite caused by frequent update in flash memory, b-tree flash translation layer was proposed [13]. Since this scheme requires additional memory area and additional latency, FlashDB scheme was developed through dynamic automatic tuning. In addition, MicroHash and MicroGF structures were proposed for small storage spaces such as sensors which use a small amount of energy, but they are difficult to apply to large data [14].

3.2. File System for Flash Memory Devices

NAND-type flash memory as a main storage for sensor devices has different characteristics from conventional magnetic disks. In flash memory, stored data cannot be overwritten when data is written. In order to change the previously stored content, a write operation must be performed on a new empty block after performing an erase operation on the block in which the content is stored. In addition, the erasing each block is limited to about 100,000 times, and if it exceeds this limit, the block will no longer be writable [13].

A log-structured file system(LFS) is a file system in which data and meta data are written sequentially to a circular buffer, called a log. Conventional file systems tend to lay out files with great care for spatial locality and make in-place changes to their data structures in order to perform well on optical and magnetic disks, which tend to seek relatively slowly (Figure 3). On the other hand, the log structure is naturally suited to media with append-only zones or pages such as flash storages. LFS buffers relevant data in main memory and writes it sequentially with metadata such as inode to the flash memory on segment basis. Therefore, it does not require search time to update blocks and inodes.

LFS storage should reclaim free space from the tail of the log to prevent the file system from becoming full when the head of the log wraps around to meet it. To reduce the overhead incurred by this garbage collection, most implementations avoid purely circular logs and divide up their storage into segments[15].

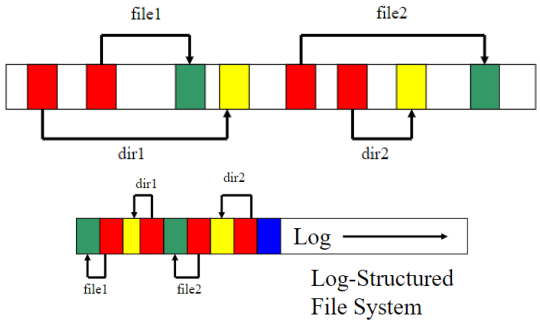

Figure 3. Conventional file system vs. Flash file system

The LFS relays the redirection between the inode number and the inode through the inode map. The input of the map is the inode number, and the output is the latest address of the inode. Each segment has a summary block which is linked to a long chain structure connected by the next summary pointer, so LFS becomes a long linear log. Due to its random access nature, flash memory does not require the segments to be adjacent.

The most well-known file system for embedded systems using flash memory is JFFS2 which is the enhanced version of Journaling Flash File System (JFFS)[16]. JFFS is an LFS-based file system developed by Axis Communications in Sweden that takes into account the nature of flash memory by improving the general file system. Traditional file systems fix meta-information to specific blocks and update them each time the file system is modified. However, JFFS uses the entire blocks of flash memory evenly without using specific blocks. JFFS also stores meta-information of the file system every certain interval to prevent data loss caused by abrupt power failures.

In this study, proposed scheme recycles invalidated old data blocks that are discarded when new data is stored. In other words, proposed scheme exploits recycled shadow pages for efficient transaction recovery with performance and stability.

3.3 Shadow-based Data Recovery for flash-based Edge Gateway

Conventional shadow paging techniques copy the current page table to the shadow page table when the update (write) transaction begins to execute. The list of current page tables points to the most recent data pages and is overwritten as the transaction is performed, while the shadow page tables remain unmodified until the transaction ends. This means that a copy of a new page is created when a update transaction is executed, but the previous copy of the page is not overwritten. Thus, two versions are maintained for the update transaction. The two versions are the previous version referenced by the shadow page table and the new version by the current page table.

If a transaction failure occurs and it needs to be returned to the previous version, shadow schemes discard the current page table and replace the previous shadow page table with the

current page table. Otherwise, if the transaction is completed successfully, the shadow page table is no longer needed and can be discarded. In this respect, shadow paging schemes are classified as no-redo/no-undo techniques because there is no redo/undo operation [8].

Traditional shadow paging schemes require large space to maintain shadow pages and destroy the original layout of data pages. To reduce the significant space overhead of the traditional schemes, Reused Shadow Recovery (RSR) scheme is proposed. RSR scheme is suitable for IoT edge gateways with flash memory storage.

However, the characteristics of the flash memory file system described in the previous section must be considered for efficient RSR. In the traditional LFS-based approaches, data blocks are stored once and expired once due to the physical properties of write-once flash memory.

On the other hand, proposed RSR scheme reuses the expired blocks of discarded version which are supposed to be disposed. To reuse these invalidated pages efficiently, we devised a new shadow paging scheme with deferred cleaning technique. Figure 4 shows the flash memory-based shadow recovery mechanism and an operation interface for JFFS-like file systems, as follows.

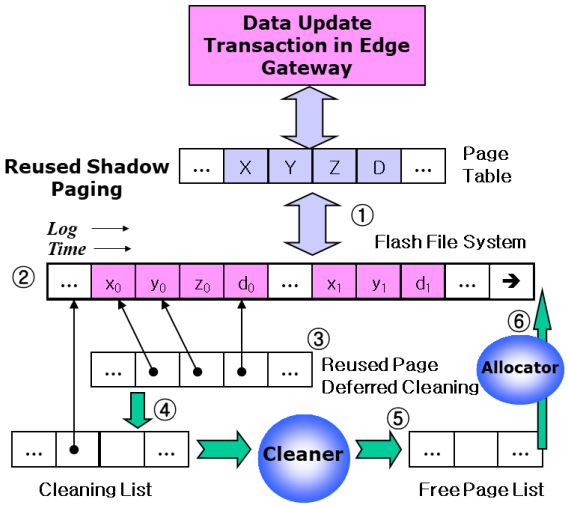

Figure 4. Reused Shadow Page Management for Edge Gateway

1) The user's update application connects to the RSR data management system by API or system call and performs write transaction for data updates processing. For this update transaction, data items stored in the associated flash file system are accessed through the corresponding page table.

2) RSR does not overwrite old shadow pages for the data items, but creates new pages and stores the modified values on the new pages. The new pages are assigned in a sequential direction by the flash block allocator.

3) The most recent data pages for the updated data item is retained in the reused shadow area and used as a backup page. In RSR, the erase operations for invalidated pages are pending until the update transaction is completed. The invalidated pages are recycled as shadow pages for fast recovery process if the update transaction is aborted.

4) If the update transaction has reached to commitment state and the shadow pages are unnecessary, the shadow pages are registered into the segment erasing area for actual page cleaning.

5) Periodically, the shadow pages registered in the segment erasing area are inserted into the free page area after the segment erasing process.

6) The cleaned pages are inserted into the free page area to supply new pages by flash allocator. Note that the shadow pages can be reused when the flash file system is not completely occupied and there is some free space. For example, if there is not enough block space for shadow paging, the advantages of shadow paging are reduced and system overhead increase because the flash file system uses new block spaces immediately through the cleaning process. However, IoT data transactions are basically read-intensive without roll back for large-scale data analysis. In addition, general flash storages typically retain at least 5% free space, so no overflow occurs.

The proposed RSR scheme allows the update operation to be processed without logging overhead. Since logging and check pointing overheads are eliminated, RSR scheme can achieve high transactions performance with low response time.

The shortcomings of the traditional shadow paging schemes can be resolved in RSR as follows. First, traditional techniques have the disadvantage of consuming a lot of storage space for shadow pages. However, in RSR scheme, the additional space is not wasted because old data pages that are discarded from the flash file system are recycled as new shadow pages. Thus, the space burden for shadow pages can be reduced.

Second, in disk-based file system environments, updated data pages frequently change their locations on disk storage as shadow pages increase. This results in poor I/O performance and consequently poor system performance because the associated pages are distributed widely rather than concentrated. However, RSR scheme uses flash memory as data storage device rather than hard disk device, enabling fast random access as like main memory. Therefore, this disadvantage is also overcome because I/O performance degradation is not caused by the data page distribution.

As a result, the proposed RSR scheme reduces the logging and rollback overhead caused by update transactions and increases the transaction performance for IoT gateway systems. In addition, RSR scheme minimizes the waste of storage space for shadow paging by recycling old data pages in flash file systems.

IV. Performance Evaluation

4.1 Experimental System Model

The researcher compared the performance of RSR(Reused Shadow Recovery) scheme, UR(Update-in-Place Recovery) scheme, and SR(Shadow Recovery) scheme. Simulation system was programmed using C++ and CSIM library [17] for workload generation and performance analysis. The simulation model for the flash memory-based data environment is based on the closed queuing model provided by the CSIM.

The CSIM simulation module generates data transaction workloads, but actual I/O delay is measured from the built-in flash memory storage (256GB) to increase its practicality. The experimental system consists of a user transaction generator, a transaction manager for the user transaction, a data manager, and a page manager.

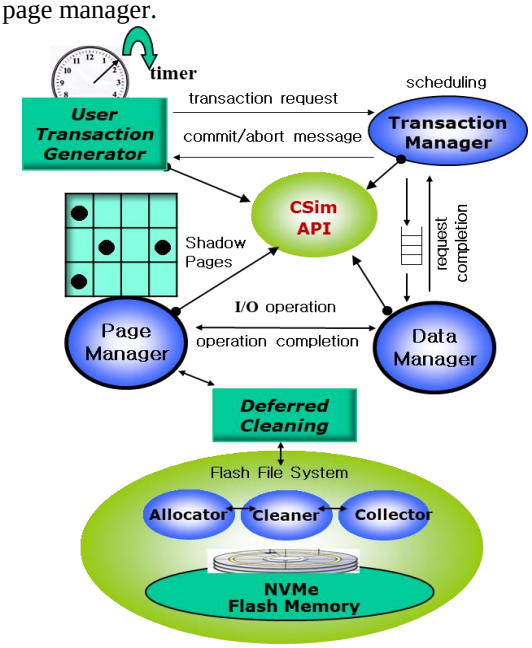

Figure 5. Simulation Model for Performance Evaluation

The user transaction generator generates read or update(write) transactions at predefined intervals to make the user transaction workload required for the simulation. The transaction manager manages and analyzes the user transaction, and then sends it to the data access request queue of the data manager. After logical data access for the transaction, the data manager sends a physical access request to the flash-based shadow page manager. The shadow page manager manages recycled shadow pages and perform actual I/O requests in flash memory for the user transactions.

The performance evaluation metrics for the simulations are flash storage space consumption, transaction throughput, and response time of the transaction. The throughput rate is defined as the number of transactions that are successfully completed per second, and the response time is defined as the time that elapses between the submission and the completion of the transaction. Flash memory space consumption is the amount of data required to manage shadow pages and related information accompanied by the transaction execution.

4.2 Test results and their interpretation

In this experiment, the performance of UR, SR, and RSR was analyzed. We looked at the changes in the workload of each scheme by varying the number of user transactions in the experiment. The main result of the experiment are shown in Figure 6 (transaction throughput), Figure 7 (transaction response time), and Figure 8 (space consumption).

In Figure 6, the number of transactions entering the CSIM queuing system gradually increases as the volume of transactions increases, resulting in a gradual increase in workload. This increase is mainly due to the increment of paging congestion that follows the increment of the workload level. In this experiment, the highest throughput is exhibited by SR, followed by RSR and UR. Experimental results show that the graph curves of SR and RSR schemes are located at the top of the UR scheme in the entire interval, so the transaction processing results of the two shadow paging schemes are better than UR scheme.

Figure 7 also shows that the average response time gradually increases, as the amount of transactions generated per second increases. The graphs of the two shadow paging schemes shows that their responsiveness is better than that of the UR scheme under middle and high workload condition.

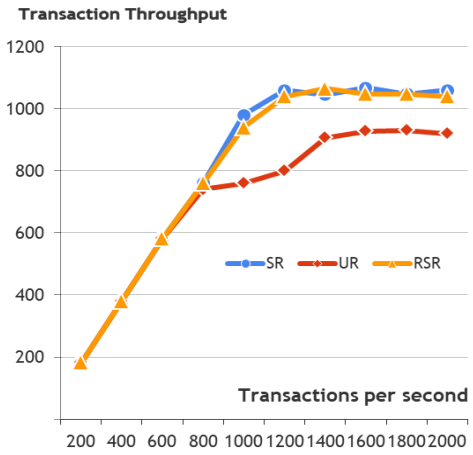

Figure 6. Transaction Throughput

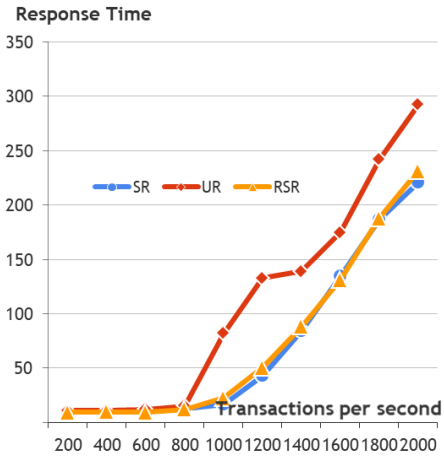

Figure 7. Transaction Response Time

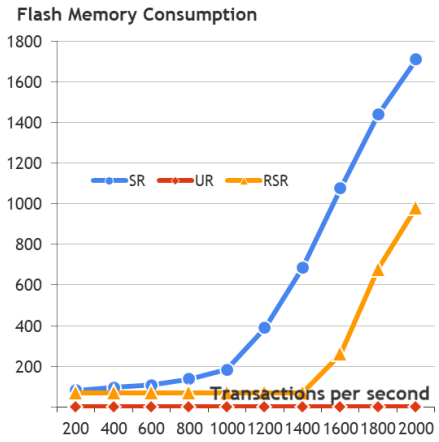

Figure 8. Space Consumption of Flash Storage Device

As the amount of transaction generated per second exceeds 1200 to 1400, the transaction throughput is no longer enhanced and system performance is gradually degraded. This means that even though the transaction processing requests beyond that range, the system suffers from performance degradation caused by data contention, meaninglessly increasing the system workload.

As a result of the analysis of the graph above, RSR showed similar transaction processing and response performance to SR. SR shows slightly higher performance than RSR in high workload environment. This small deference is because RSR also reduces the degree of data contention caused by shadow paging. On the other hand, RSR effectively reduces the burden of storage space by recycling data pages that are invalidated and discarded inside the flash file system. Note that UR should write new image after erasing the previous image, so it could suffer from flash overwrite congestion caused by the slow erase operations, especially in high workload conditions.

Figure 8 shows the results of storage space usage for each scheme. As previously mentioned, UR does not store previous images, so it uses the least storage space. Note that RSR scheme uses much less storage space than SR over the entire interval. In terms of storage space consumption, RSR scheme uses 29% lower space than SR scheme. This means less workload is placed on shadow page storage management. However, if the recycling shadow area is insufficient, RSR could suffer from recycling overflows and this could lead to transaction performance degradation.

V. Conclusion

This paper described characteristics of IoT edge computing and edge gateway device with flash memory storage, as well as the structure of flash file systems considering the characteristics of flash memory. This paper proposed a new transaction recovery scheme for flash memory-based edge gateways.

The proposed shadow recovery scheme reduced the logging and rollback workload compared to the immediate renewal technique of conventional recovery schemes. In addition, the proposed scheme recycled the discarded data pages through the deferred shadow cleaning to reduce the additional space overhead of conventional paging schemes, thus increasing the storage space efficiency and transaction processing performance. CSIM simulation results show that the space gain of the proposed scheme reaches even to 29% with respect to storage consumption.

Future study includes light-weight shadow management for small-size gateways and extending the proposed scheme to hybrid storage environment where the types of data storage and I/O bandwidth are different, and deep learning algorithms to optimize space consumption .

References

[1] P. Bonnet, J. Gehrke, and P. Seshadri, Towards Sensor Database Systems, Proceedings of the Second International Conference on Mobile Data Management, pp.3-14, Hong Kong , Jan. 2011

[2] S.V.R.K.Rao, M.Saritha Devi, A.R.Kishore, Praveen Kumar, Wireless sensor Network based Industrial Automation using Internet of Things (IoT), International Journal of Advanced Trends in Computer Science and Engineering, vol.7, no.6, pp.82-86, 2018.

[3] Y. Son, Y. Lee, "A Study on the Interpreter for the Light-Weighted Virtual Machine on IoT Environments," International Journal of Web Science and Engineering for Smart Devices, Vol.3.2, pp.19-24, 2016

[4] Wei Yu, Fan Liang, Xiaofei He, William Grant Hatcher, Chao Lu, Jie Lin, Xinyu Yang, A Survey on the Edge Computing for the Internet of Things, IEEE Access, vol.6, pp.6900-6919, Nov. 2017.

[5] Gopika Premsankar, Mario Di Francesco, and Tarik Taleb, Edge Computing for the Internet of Things: A Case Study, IEEE Internet Of Things Journal, vol.5, No.2, 2018

[6] Amir M. Rahmani, T. Nguyen Gia, B. Negash, A. Anzanpour, I. Azimi, M. Jiang, P. Liljeberg, Exploiting smart e-Health gateways at the edge of healthcare Internet-of-Things: A fog computing approach, Future Generation Computer Systems, vol. 78, no.2, pp. 641-658, 2018

[7] A. ElSharif Karrar, M. Fadl Idris Fadl, Security Protocol for Data Transmission in Cloud Computing, International Journal of Advanced Trends in Computer Science and Engineering, vol.7, no.1, pp.1-5, 2018.

[8] Tamer Ozsu, and Patrick Valduriez, Principles of Distributed Database Systems, Springer New York, 2011.

[9] Vijay Kummar, Albert Burger, "Performance Measurement of Main Memory Database Recovery Algorithms Based on Update-in-Place and Shadow Approaches", IEEE Transactions on Knowledge and Data Engineering, 4(6), 1992, pp. 567-571.

[10] Jack Kent, Hector Garcia-Molina, "Optimizing Shadow Recovery Algorithms", IEEE Transactions on Software Engineering, 14(2), Feb. 1988, pp. 155-168.

[11] J. Kim, S.Joo, H. Kang, An Efficient Recovery System for Spatial Main Memory DBMS, Journal of the Korea Spatial Information Society, Vol 8 No. 03, pp. 1-14, 2006.12

[12] E. M. Song, Y. K., Kim and C. H., Ryu "No-Log Recovery Mechanism Using Stable Memory for Real-Time Main Memory Database Systems", RTCSA'99, IEEE CS, Dec 1999, pp. 428-431.
[13] S.W. Byun, M. H. Hur, "An index management using CHC-cluster for flash memory databases", Journal of Systems and Software, 82(5), 2009, pp.825-835.
[14] E. J. Lee, M. S. Jeong, H. K. Bahn. "NVM-based Write Amplification Reduction to Avoid Performance Fluctuation of Flash Storage", The Journal of The Institute of Internet, Broadcasting and Communication, 16(4), 2016, pp.15-20.
[15] Log-structured file system, https://en.wikipedia.org/wiki/Log-structured_file_system
[16] Journaling Flash File System version 2, https://en.wikipedia.org/wiki/JFFS2
[17] CSIM,https://www.csim.com/overview.html,IntroductiontoCSIMModelingEnvironment.