Proceedings of

The International Conference on Intelligent Information Technology (2INTECH 2015)

and

International Workshop on Advanced Computing and Multimedia Technology (ACMT 2015)

October 21 - 23, 2015

Guam, USA

2INTECH 2015 and ACMT 2015

Table of Contents

A Study on Fast Partition Page Table Management for the DIMM Tree Architecture

Young-Kyu Kim[1], Yong-Hwan Lee[2], Byungin Moon[1]

[1.]School of Electronics Engineering, Kyungpook National University, Daegu, Korea
[2.]School of Electronic Engineering, Kumoh National Institute of Technology, Gumi, Korea

bihmoon@knu.ac.kr

Abstract

Recently, as a method to service high-quality multimedia contents, in-memory computing systems have been attracting public attention.The in-memory computing was developed to keep the entire data in main memory to avoid expensive mechanical harddisc access and to increase the ability to process large volumes of data or complex data. Adualinlinememorymodule (DIMM) treearchitecture (DTA) wasproposedasa newmemorysystemtoimplementthein-memorycomputing system. In addition, the DTA uses a partitioned DIMM tree policy for efficient memory management. However, the partitioned DIMM tree has several drawbacks, such as unnecessary hardware overhead and conflict between the system area and the user area. So, this paper proposesan advanced method to overcome the unnecessary hardware overhead, andsuggests two methods to avoid thesystem area conflict problem. We modeled and simulated the proposed methods, and presentedthe most efficient method in terms of the system performance by analyzing the simulation results.

Keywords:DIMM tree architecture; in-memory computing; big memory server; main memory; DRAM

1. Introduction

Recently, the server systems for servicing multimedia contents have become a center of attraction; particularly storage performance of server systems is regarded as of major importance, because multimedia files have recently changed to highcapacity and high definition format [1-3]. However, traditional disk-based server system for multimedia server is inefficient because of poor disk performance, and thus related studies which use use memory-based storage such as DRAM-based storage have been proposed [4,5]. Moreover, the multimedia server system was suggested to employ in-memory computing technologies because of Big-data issue in recent years [6]. The in-memory computing was developed to keep the entire data in main memory to avoid expensive mechanical hard-drive I/O access and to increase the ability to process large volumes of data or complex data[6].

Advanced memory technology, drastic decline in price of memory, and evolution of 64-bit multi-core processors led development of in-memory computing technology [5]. However, hardware platforms for the in-memory computing still have several significant problems. Among the problems, traditional memory interfaces, such as themulti-drop bus and thepoint-to-point links are not suitable for the in-memory computing platform because of signal integrity and access latency [7]. Dual inline memory module (DIMM) tree architecture (DTA)

was proposed to solve the drawbacks of the traditional memory interface methods, and to implement the large-capacity memory system. The DTA is able to grow the number of DIMMs exponentially with each level of latency in the tree (Figure 1) [7].

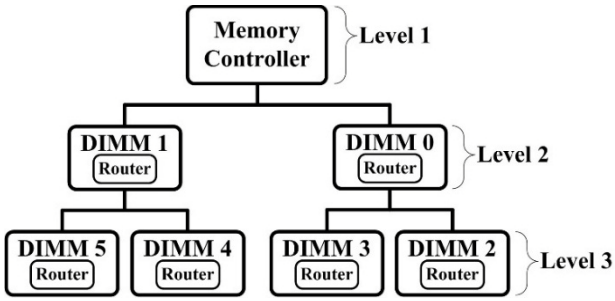


Figure 1. DIMM tree architecture.

Studies about DTA have been progressed to improve system performance and to implement a actual DTA system. Particularly [8] suggested the efficient policiy for management large-capacity main memory and themethod for direct data transaction between DIMMs in two levels. Such studies are excellent for improving performance of the DTA system, but this paper found two critical problems those must be overcame to implement the DTA. Therefore, this paper studies the problems of existing DTA, and proposes solutions for overcoming the studied problems.

2. Background

DIMM tree architecture uses a special type of DIMMs called tree DIMMs (T-DIMMs). A T-DIMM has two external channels and one internal channel. The external channels are respectively used for connecting an upper level and a lower level. The internal channel is used to drive DRAM ranks in T-DIMMs. A T-DIMM must be able to choose between aborting the received command, executing that command, or forwarding it to lower level T-DIMMs in order to process a received command. Such operations are ran by a DIMM interface router (DIR) in the T-DIMMs(Figure 2) [7].

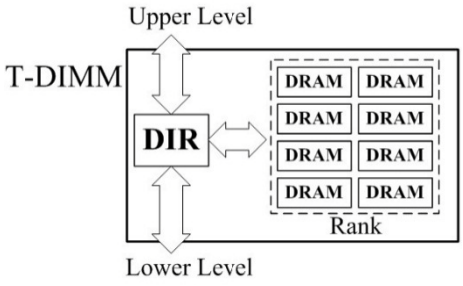


Figure 2. T-DIMM block diagram.

The DTA using a partitioned DIMM tree policy was proposed for efficient memory management.The partitioned DIMM tree policy partitions the DIMM tree into a fast partition and slow partition. The fast partition is composed of the faster levels of the DIMM tree and acts as a cache for the pages in the rest of the DIMMs in the slow partition which is used as the main memory. In addition, the DTA proposed direct DIMM-to-DIMM

transfer in order to reduce contention of the memory channel caused by transferring pages between the fast and slow partitions. The partitioned DIMM tree needs to use fast partition page table (FPPT) to keep track of the pages in the fast partition, and the FPPT management is similar to a traditional page table(Figure 3) [8]. The Figure 3 is an example of 4-way FPPT in the DTA using 39-bit physical address.

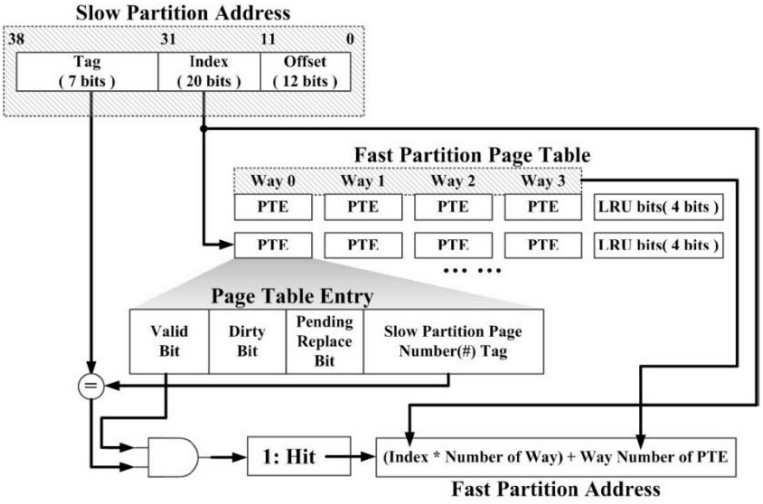


Figure 3. Fast partition page table entry and fast partition address.

In order to access the main memory, first it must access the FPPT to check the fast partition hit or miss byusing a tag and index of slow partition address (physical address) (Figure 3). If the result is hit, a fast partition address is index * number of way + way number, as in Fast Partition Address of Figure 3. If the result is miss, the slow partition page has to be uploaded to the fast partition. By the way, if the fast partition entry is already full, the page must be replaced by the page replacement policy, which is similar to the thatof the operating system. The fast partition is a setassociative structure, and some part of the fast partition is assigned to the FPPT [8].

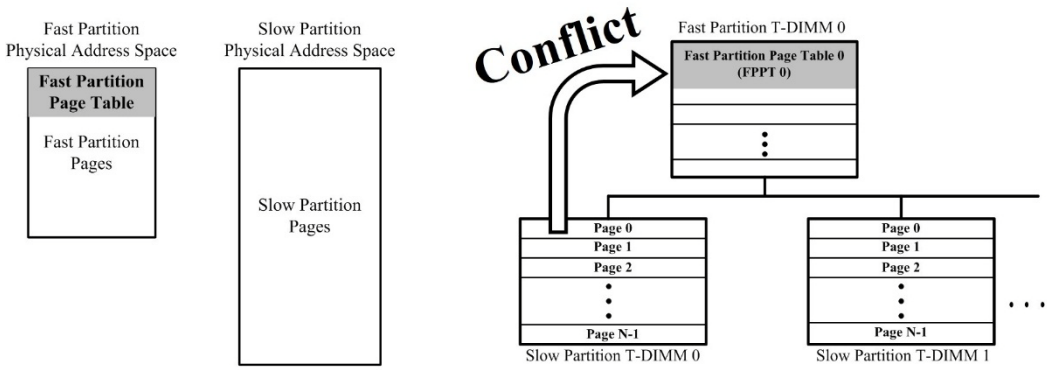
3. Motivation

Partitioned DIMM tree policy is an efficient method to improve system performance. However, this paper found following problems about using the FPPT.

A. **Page table hardware overhead:**In order to access the main memory in the DTA, the FPPT must be alwaysaccessed to check the fast partition hit or miss. [8] proposed modified TLB and page table for solving the problem. Each TLB and page table entry add a bit to specify whether the current page exists in the fast partition or not, and a field for the fast partition page number [8]. However, modified page table does increase the hardware overhead. Therefore, this paper proposes DIMM tree TLB (DT-TLB) that is an advanced TLB management method for the partitioned DIMM tree without to modify the page table.

B. **FPPT Squatting:**Since the FPPT is located in a part of the fast partition, the FPPT of the fast partition needs to be protectedfrom user accesses. However, as the fast partition address is a subset

of the slow partition address, the some pages of slow partition,which has corresponding address with FPPT area,must notbe updated to the fast partition (Figure 4). To overcome this problem, this paper proposes fast partition management policies, such as Fast-FPPT and Slow-FPPT.



(a) FPPT in the fast partition (b) Conflict with the FPPT area and slow partition pages

Figure 4. Conflict with the FPPT area and slow partition pages corresponding its FPPT area.

4. Proposed Methods

This paper proposes the DT-TLB and describes its concrete managements. The DT-TLB is similar with a traditional TLB, but it addsa L-flag bit to specify whether the current page exists in the fast partition or slow partition (Figure 5). If the L-flag is one, it shows that the current page exists in the fast partition, and the Frame number in the DT-TLB entry has a frame number of the fast partition. On the other hand, if the L-flag is zero, the current page exists in the slow partition, and the Frame number in the DT-TLB entry has a slow partition frame number.

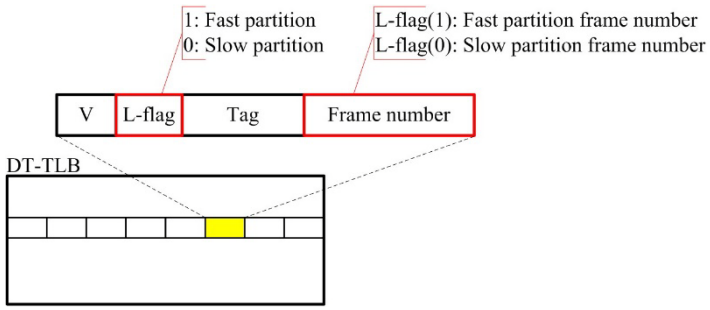


Figure 5. L-flag bit in the DT-TLB entry.

When a new frame is updated into the DT-TLB entry caused by the TLB miss,the L-flag of the DT-TLB is selected(Figure 6). If current upload frame has been already referenced by the FPPT, the initial value of the L-flag becomes one (Figure 6(a)), while if not, the initial value is set to zero (Figure 6(b)). Also, the frame number of the current DT-TLB entry is updated either the fast partition frame number or the slow partition frame number according to the L-flag.

When the last-level cache miss occurs, a cache line must be uploaded from the fast partition or slow partition. According to the result of the DT-TLB,it is determined which partition will be accessed. If the DT-TLB hit

occurs and L-flag is one, it accesses the fast partition without checking the FPPT (Figure 7(a)). If the DT-TLB hit occurs but L-flag is zero, it must check the FPPT, and then it need to upload or replacement the page from the slow partition to fast partition with updating the FPPT and the DT-TLB (Figure 7(b)).

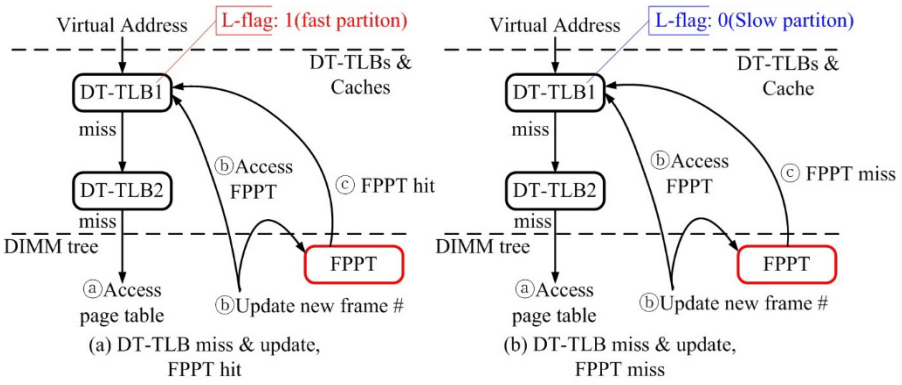


Figure 6. DT-TLB miss and update.

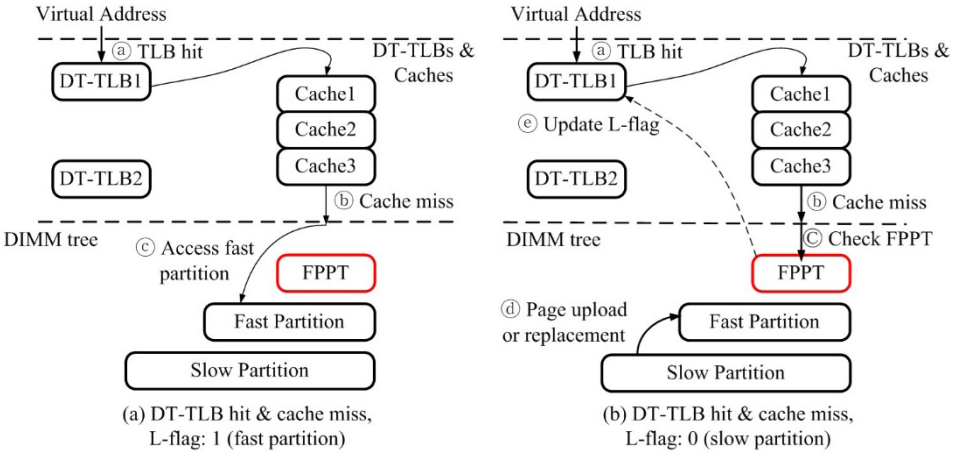


Figure 7. DT-TLB hit and cache miss.

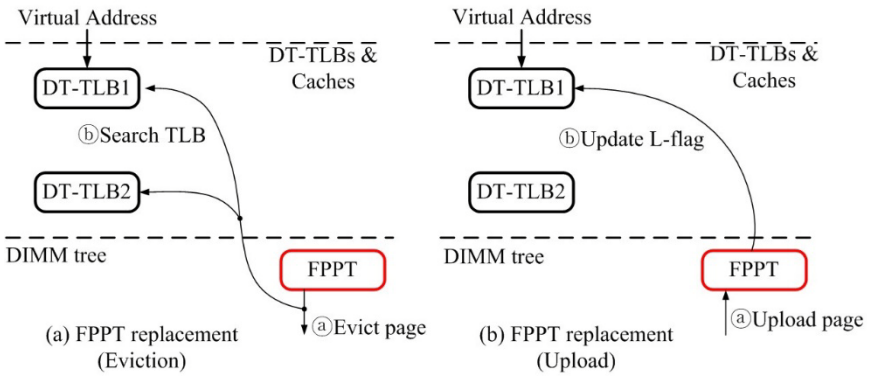


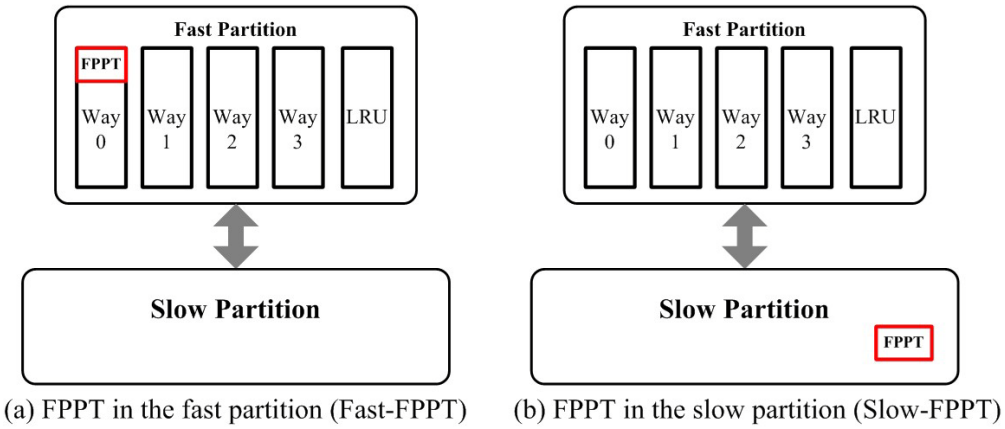
Figure 8. FPPT replacement.

Ifthe fast partition is full, one page is evited and a new page will upload by the page replacement policy. If a fast partition page is evicted, it must find its corresponding DT-TLB entry because the L-flag and frame have to

be updated to new information. However, because the FPPT hasno data for tracking the DT-TLB entry corresponding with the evicted page from the fast partition, the DT-TLB must be searched to find the DT-TLB entry by using the frame number and tag of the evicted page (Figure 8(a)). After page eviction procedure, the page upload procedure from the slow partition to the fast partition is followed. The the page upload procedurecan be overlapped with DT-TLB search procedure in order to offset the performance overhead as searching the DT-TLB. Then, the L-flag in the DT-TLB entry is updated (Figure 8(b)).

In order to solving the FPPT Squatting problem, this paper proposes two methods, Fast-FPPT and Slow-FPPT. First, the Fast-FPPT allocates only a part of one way in the fast partition for the FPPT area, (Figure 9(a)). Since the Fast-FPPT use just one way for allocating the FPPT, rest of ways can be used to upload the slow partition pages. Therefore, it can avoid the FPPT Squatting problem. However, the area allocated the FPPT has a penalty that the replacement of the area would frequently occur more than the others do.

Second, Slow-FPPT is a method that allocates a part of the slow partition for the FPPT area, (Figure 9(b)). Actually, this method may become an inefficient method in terms of the performance, but the FPPT in the slow partition would be rarely accessed because of high hit rate of the DT-TLB. Therefore, it can be considered that there is almost no influence of performance because most access is processed by the L-flag of the DT-TLB.



(a) FPPT in the fast partition (Fast-FPPT) (b) FPPT in the slow partition (Slow-FPPT)

Figure 9. Fast-FPPT and Slow-FPPT.

Using the Fast-FPPT enables quick access to the FPPT, but a part of the way used for the FPPT area could occur frequent page replacement, and using the Slow-FPPT has to access the FPPT in the slow partition. In this paper simulates the Fast-FPPT and the Slow-FPPT to verifythe efficient method in terms of the system performance.

5. Experiments

In order to verify the proposed methods, it needs workloads for verifying a large-capacity main memory system. Therefore, in this paper, we prepared 14 workloads, which are extracted from SPEC OMP2012 benchmarks by the Pin tool [9,10]. Each workload has 10 billion virtual addresses. In addition, according to the locality of the extracted addresses, it classified into high memory intensive workloads, normal memory intensive workloads andlow memory intensive workloads(Table 1). The workload_Set_1 consists of the five high

memory intensive workloads, and the workload_Set_2 has five normal memory intensive workloads. The four low memory intensive workloads do not be used for the simulation because the loads of the main memory are nearly not occurred. This paper simulated to process the virtual addresses in the workloads by TLBs, caches, and main memory, and counted clock cycles for processing the all addresses in the workload set.

Table 1. Workload description

Workload set	SPEC OMP2006 Benchmarks in workload	Number of addresses	Description
Workload_Set_1	351.bwaves,363.swim, 360.ilbdc 370.mgrid331, 372.smithwa,	Each benchmark has 10 billion addresses. (A total of 50 billion)	High memory intensiveworkloads
Workload_Set_2	357.bt331,359.botsspar, 362.fma3d,367.imagick, 376.kdtree	Each benchmark has 10 billion addresses. (Total of 50 billion)	Normalmemory intensive workloads
Do not use	350.md, 352.nab, 358.botsalgn, 371.applu331	Each benchmark has 10 billion addresses. (A total of 50 billion)	Low memory intensive workloads

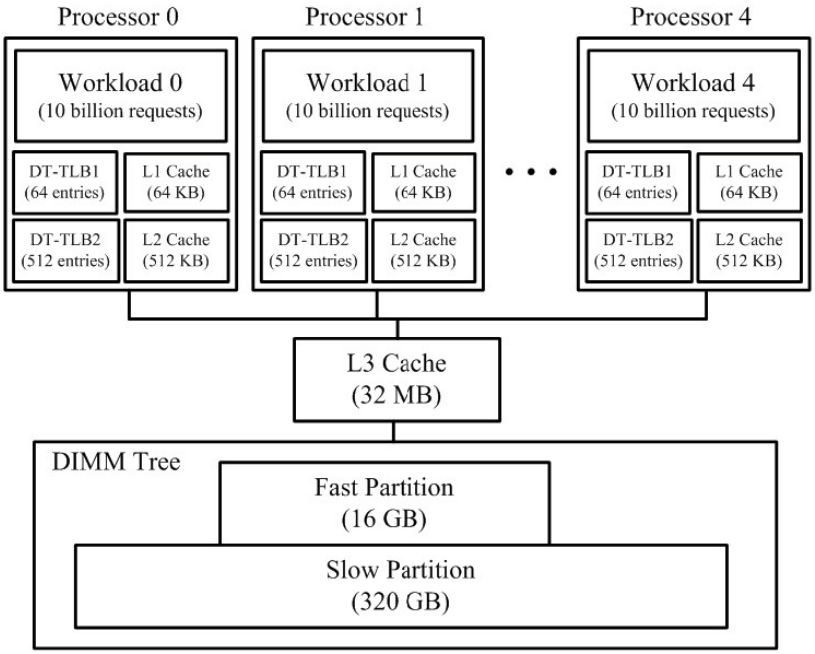


Figure 10. DTA system hardware architecture.

In order to verity the proposed DT-TLB operations and to predict performances for Fast-FPPT and Slow-FPPT, this papermodeled the DT-TLB with Fast-FFPT system and DT-TLB with Slow-FPPT system, respectively. The DTA system for modeling is aimed at a high-end server computer system for supporting the high-quality multimedia contents. Figure 10 shows the system architecture for modeling. The DTA system consists of five processors, each processor has 2-level DT-TLBs with caches. The Level 3 cache is external to the processor, and is used to a shared cache. The size of a T-DIMM is 4 GB, DIMM tree with branch factor and depth of four. Accordingly, the capacity of the fast partition is 16 GB, and slow partition is 320 GB. The fast partition is 4-way associative.

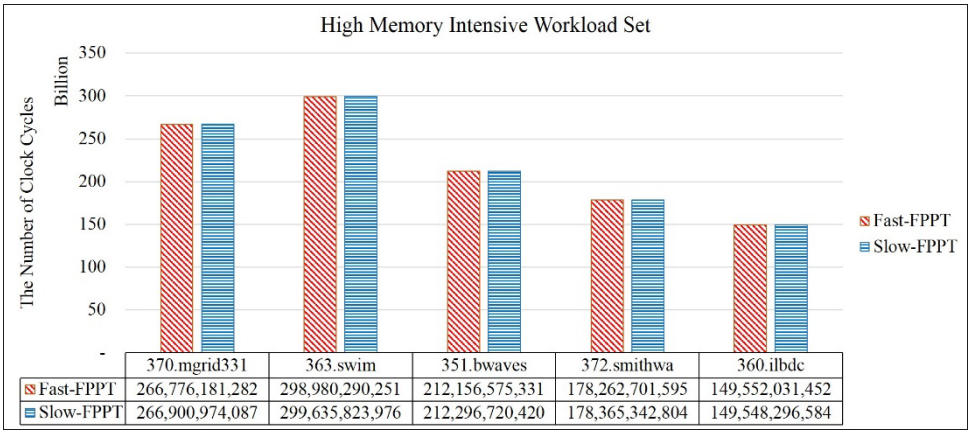
As five processors in the modeled system were implemented by using the multi-thread code, the modeled system can run five workloads in the workload set simultaneously. The two models respectively executed the workload_Set_1 and workload_Set_2 in the Table 1, and we compared the simulation results. We obtain the simulation condition like the real systems, such as natural contention of shared resource and large main memory load because of using such simulation method. Parameters of the simulation are configured by previous studies(Table 2) [11-13].

Table 2. Simulation Configuration.

Parameters	Configuration
Level 1 DT-TLB	8 ways, 64 entries, 1 clock latency
Level 2 DT-TLB	32 ways, 512 entries, 5 clock latency
Level 1 Cache	8 ways, 1,024 entries, 5 clock latency
Level 2 Cache	16 ways, 8,192 entries, 12 clock latency
Level 3 Cache	32 ways, 524,288 entries, 30 clock latency
Main memory	320 GB, 200 clock latency
DIMM to DIMM switch time	1 clock latency

6. Experimental Results

Figures 11 and 12 show the the simulation results of the DT-TLB model with Fast-FPPT versus the DT-TLB with Slow-FPPT model. high memory intensive Workload_Set_1 and normal memory intensive Workload_Set_2 are used as workloads for evaluating the two models. Figures 13 and 14 show the difference in the performance between the Fast-FPPT and the Slow-FPPT. As each of workloads has 10 billion requests, the simulation resultsshow thenumber of clock cycles that are needed to process total 50 billion requests. In simulation results, the performance of the Fast-FPPT is more excellentthan that of the Slow-FPPT at most of workloads, butthe Fast-FPPT is worse than the Slow-FPPT at the 376.kdtree and the 367.imagick. Overall analysis resultis that the performanceisinfluenced by the fast partition hit rate of the L-flag in the DT-TLB.



	370.mgrid331	363.swim	351.bwaves	372.smithwa	360.ilbdc
Fast-FPPT	266,776,181,282	298,980,290,251	212,156,575,331	178,262,701,595	149,552,031,452
Slow-FPPT	266,900,974,087	299,635,823,976	212,296,720,420	178,365,342,804	149,548,296,584

Figure 11. Comparison of the system performances between Fast-FPP and Slow-FPPT with a high memory intensive workload set.

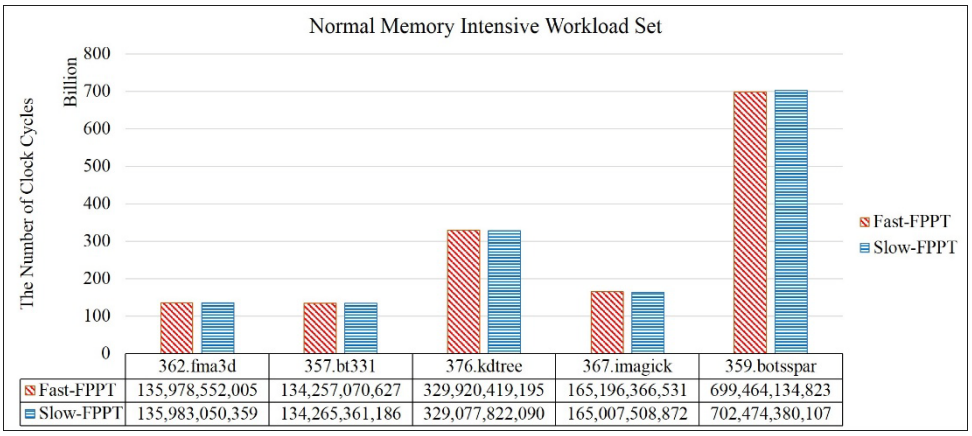


Figure 12. Comparison of the system performances between Fast-FPP and Slow-FPPT with a normal memory intensive workload set

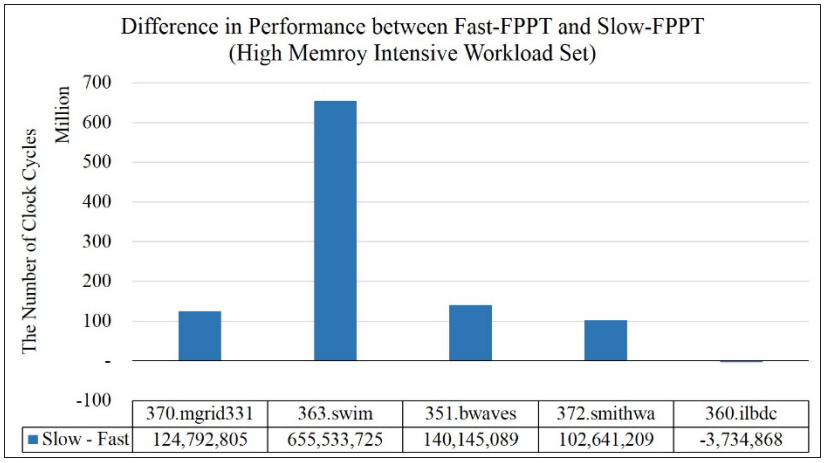


Figure 13. Difference in the performance between the Fast-FPPT and the Slow-FPPT with a high memory intensive workload set

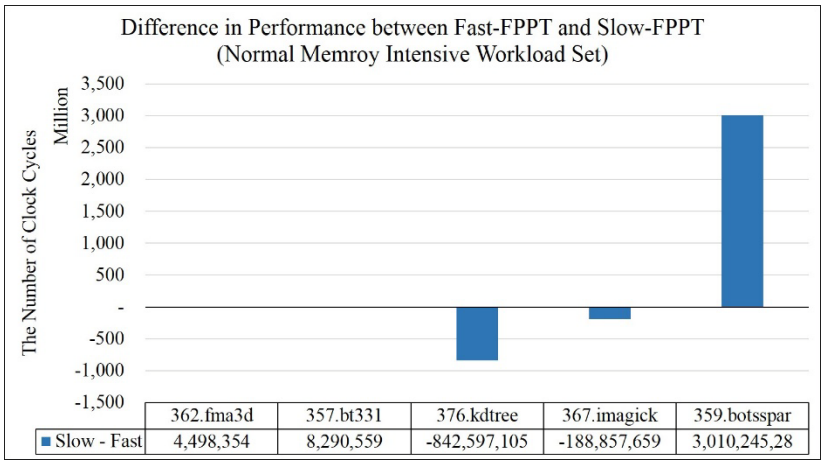


Figure 14. Difference in the performance between the Fast-FPPT and the Slow-FPPT with anormal memory intensive workload set

7. Conclusion

This paper,as a study about a high-performance server system for multimedia contents, proposed DT-TLB and detailed management methods for implementing the DTA. In addition, in order to overcome the FPPT squatting problem, it also proposed the Fast-FPPT and the Slow FPPT. The proposed methods are verified to solve the problems of existing DTA by the modeling and simulation.The performance of the Fast-FPPT is better than that of the Slow-FPPT at most simulations. Though, actually, the performance of the Fast-FPPT is similar to that of the Slow-FPPT, but the performance difference would become large in the actual multimedia server system, because server systemsusually operate without rest for a long time. This studyis expected to contribute to implement the high performance multimedia server for supporting high-quality multimedia contents. Also, it could be applied to implement a datacenter or a big data database. Finally, our future work is to study on hardware architecture, organization, and power opimization for the DIMM tree and T-DIMM to implement actually the DIMM tree architecture.

Acknowledgements:
This investigation was financially supported by Semiconductor Industry Collaborative Project between Kyungpook National University and Samsung Electronics Co. Ltd.

Corresponding Author:
Associate professor Byungin Moon
School of Electronics Engineering
Kyungpook National University
Buk-gu, Daegu 41566, Korea
E-mail: bihmoon@knu.ac.kr

References

1. Shirale, S., Patmas, M., Gunjal, P., & Rane, D. (2015). A Online Multimedia Data Processing on Cloud and Hadoop Platform. International Journal of Computer Technology and Electronics Engineering 2015;5(2):21:24.
2. Nieh J, Yang SJ. Measuring the Multimedia Performance of Server-Based Computing. Proceedings of Proceedings of the 10th International Workshop on Network and Operating System Support for Digital Audio and Video, Chapel Hill, North Carolina, USA Jun. 2008;55-64.
3. Ramesh B, Savitha N, ManjunathAE. Mobile applications in multimedia cloud computing. International Journal of Computer Technology and Applications 2013;4(1):97.
4. Lechtenborger J, Vossen G, Zeier A, Mruger J, Muller J, Lehner W, Kossmann D, Fabian B, Gunther O, Winter R. In-memory databases in business information systems.Business & Information Systems Engineering2011;3(6):389-395.
5. Brocke JV, Debortoli S, Muller O, Szekely N. How in-memory technology can create business value: insights from the Hilti case. Communications of the Association for Information Systems 2014;34(1):151-167.
6. Ousterhout J, Agrawal P, Erickson D, Kozyrakis C, Leverich J, Mazieres D, Mitra S, Narayanan A, Parulkar G, Rosenblum M, Rumle SM, Stratmann E, Stutsman R. The case for RAMCloud. Communications of the ACM 2011;54(7):121-130.
7. Therdsteerasukdi K, Byun GS, Ir J, Reinman G, Cong J, Chang MF. The DIMM tree architecture: A high bandwidth and scalable memory system. Proceedings ofInternational Conference on Computer Design (ICCD), 2011 IEEE 29th International Conference, Massachusetts, USAOct. 2011;388-395.
8. Therdsteerasukdi K, Byun GS, Ir J, Reinman G, Cong J, Chang MF.Utilizing Radio-Frequency Interconnect for a Many-DIMM DRAM System. Emerging and Selected Topics in Circuits and Systems, IEEE Journal 2012;2(2):210-227.

9. Muller MS, Baron J, Brantley WC, Feng H, Hackenberg D, Henschel R, Jost G, Molka D, Parrott C, Robichaux J, Shelepugin P, Waveren MV, Whitney B, Kumaran K. SPEC OMP2012—an application benchmark suite for parallel systems using OpenMP. Proceedings ofInternational Workshop on OpenMP in a Heterogeneous World. Springer Berlin Heidelberg, Rome, Italy Jun. 2012;223-236.
10. Luk CK, Cohn R, Muth R, Patil H, Klauser A, Lowney G, Wallace S, Reddi VJ, Hazelwood K. Pin: building customized program analysis tools with dynamic instrumentation. ACM Sigplan Notices 2005;40(6):190-200.
11. Muralimanohar N, Balasubramonian B, Jouppi NP. CACTI 6.0: A tool to understand large caches. University of Utah and Hewlett Packard Laboratories, Tech. Rep, 2009.
12. Shafaei M, Fei Y. HiTS: A High Throughput Memory Scheduling Scheme to Mitigate Denial-of-Service Attacks in Multi-core Systems. In:Computer Architecture and High Performance Computing (SBAC-PAD), 2014 IEEE 26th International Symposium on. IEEE Paris, France Oct. 2014;206-213.
13. Metha S, Fang Z, Zhai A, Yew PC. Multi-stage coordinated prefetching for present-day processors. Proceedings of the 28th ACM international conference on Supercomputing. ACM, Muenchen, Germany Jun. 2014;73-82.