Data Placement Based on Data Semantics for NVDIMM/DRAM Hybrid Memory Architecture

Gaku Nakagawa, Shuichi Oikawa
Department of Computer Science
University of Tsukuba
Tsukuba, Ibaraki, Japan
e-mail: {gnakagaw, shui}@cs.tsukuba.jp

***Abstract*— Non-Volatile Dual Inline Memory Module (NVDIMM) makes it possible to expand the main memory with non-volatile memory. However, constructing the main memory only with NVDIMM is unrealistic because NAND Flash, the most promising candidate as NVDIMM device, has several shortcomings about write access. The hybrid memory architectures with NVDIMM and Dynamic Random Access Memory (DRAM) is a method to hide the shortcoming of NAND Flash. In the architecture, we can offload write-hot data to DRAM. In this paper, we utilize data semantics to determine data placements on NVDIMM/DRAM hybrid memory architecture. The architecture requires distributing data between NVDIMM and DRAM. Data semantics (i.e., meaning of data) is useful for the decision for the data placements. As a proof-of-concept, we executed a simulation experiment to determine data allocation between NVDIMM and DRAM based on the data semantics. As a result, we could suppress write access to the NVDIMM area under only 0.2% of the DRAM area.**

Keywords-memory management; nvdimm; non-volatile memory.

I. INTRODUCTION

Non-Volatile Dual Inline Memory Module (NVDIMM) [1] makes it possible to expand the main memory with non-volatile memory. NVDIMM is an interface standard to connect between solid state drive and Dual Inline Memory Module (DIMM) slots [1]. Now, we can access SSDs (Solid State Drives) only via an input/output bus, such as Serial ATA (Serial Advanced Technology Attachment) and PCI-Express (Peripheral Component Interconnect) [2]. With the NVDIMM standard, we can access SSD via a memory bus. It reduces the latency between Central Processing Unit (CPU) and SSDs. Thus, NVDIMM makes it possible to use SSD as part of the main memory.

Constructing the main memory only with NVDIMM is unrealistic. It is required to combine NVDIMM and the existing DRAM. NAND Flash, the most promising candidate as NVDIMM device, has two shortcomings related to write access. One is that the write access latency is much larger than that of DRAM. The other is limited write endurance. Thus, if we place data with many write access (write-hot data) on NVDIMM, the system will lose its performance and durability. The hybrid memory architectures with NVDIMM and DRAM are methods to hide the shortcomings of NAND Flash [3] – [5]. In the architectures, we can offload write-hot data to DRAM.

NVDIMM/DRAM hybrid memory architecture requires distributing data between NVDIMM and DRAM. It is ideal that there are write-hot data on the DRAM area and write-cold data on the NVDIMM area. A simple way is data migration based on the number of write access. In the way, all new data is placed on NVDIMM area. If the memory manager detects write-hot data on NVDIMM area, it moves the detected data to the Non-Volatile Memory (NVM) area. However, the simple method has a problem. With this method, the memory manager cannot detect write-hot data before actual write access concentrations.

In this paper, we utilize data semantics to resolve this problem. Data has its meaning in each program context (data semantics). The semantics have their characteristics about write access (i.e., write-hot or write-cold). With the characteristics, we can determine the appropriate placement area for each data.

As a proof-of-concept, we executed a simulation experiment to determine data allocation between NVDIMM and DRAM based on the data semantics. As a result, we could suppress write access to the NVDIMM area under only 0.2% of the DRAM area.

The paper is organized as follows. Section II shows the usefulness of data semantics for data placement on NVDIMM/DRAM hybrid memory architecture. Section III shows an evaluation experiment for a proof-of-concept. Section IV shows the summary of this paper.

II. DATA PLACEMENT BASED ON DATA SEMANTICS

NVDIMM/DRAM hybrid memory architecture requires determining data placement between NVDIMM and DRAM. Data semantics is useful information for the decision. Each program places its data in its memory area. The data have semantics in each program context, such as numeric data, string data, some data structures, and so on. Each data semantic has its write access characteristics. For example, a pointer that indicates the head of a linked list has the possibility to be updated. In contrast, the string data that contains command line arguments does not have the possibility to be updated. We can predict whether data is write-hot or write-cold based on the write access

 ISBN: 978-1-61208-529-6

characteristic of the data. With that prediction, we can determine the appropriate placement area for each data.

Current operating systems do not know the data semantics in user processes because they do not take care of the data meaning. In this research project, we proposed a method to determine the data placement at programming language runtime level. In the method, a programming language runtime manages data placement between the allocated NVDIMM area and DRAM area based on the data semantics. The proposed method focuses on class types in the target program as the data semantics.

III. SIMURALATION EXPERIMENT

For a proof-of-concept, we executed a simulation experiment for data placement based on data semantics. We modify an existing Java language runtime to distribute data between 2 separated areas: pseudo-NVDIMM area and DIMM area. In the experiment, we did not use any NVDIMM device. The NVDIMM area was standard DRAM. Thus, the results did not take into account the characteristics of NVDIMM. The simulation software is implemented based on Jikes Research Virtual Machine (Jikes RVM). The base version of Jikes was 10709.

In the simulation, the language runtime corrects the characteristics of write access to each class type. The runtime determines the write-hot classes based on the corrected information. It determines data placement based on the list of write-hot classes, i.e., it places the write-hot class instance on DRAM area. The runtime often makes a miss decision. They may place the write-hot data to NVDIMM. The runtime detects the write-hot objects, as well as the write-hot class. If it found a write-hot object on the pseudo-NVDIMM area, it moves the object to the DRAM area. Also, it detects the write-cold objects on DRAM area. When the runtime finds them, it moves the detected object to the pseudo-NVDIMM area. We executed a benchmark software on the simulation software. We adopted the Jython benchmark from the DaCapo benchmark suite [6], version 2006-10-MR2.

Fig. 1 describes the number of write accesses to each memory area in chronological order. The blue and red lines represent the number of write accesses to NVDIMM and DRAM respectively. The green dotted line describes the sum of the number of write accesses to the two areas. The data shows that the number of write accesses to the pseudo-NVDIMM was much lower than that of DRAM area. The number of write accesses to the pseudo-NVDIMM area was only 0.2.

Fig. 2 describes the data distribution between the DRAM area and the pseudo-NVDIMM area. The data shows that there was much data on the pseudo- NVDIMM area. The maximum size of the DRAM area was 8.2 MiB. The average size of the DRAM area was 6.1 MiB. The maximum size of the pseudo-NVDIMM area was 29.9 MiB. The average size of the pseudo- NVDIMM was 27.3 MiB.

The results show that we can reduce the number of write access to the pseudo-NVDIMM area with data semantics. The number of write accesses to the pseudo- NVM area was much less than that to the DRAM area while the runtime placed much data on the pseudo- NVDIMM area than on the DRAM area.

IV. SUMMARY

In this paper, we propose a new approach for the data placement decision on the hybrid memory architecture. The proposed approach utilizes the data semantics to resolve the problem. Data has data semantics in the program context. The semantics have their characteristics about write access. With the characteristics, we can determine the appropriate placement area for each data. The results of a proof-of-concept evaluation show that the proposed method has significant merits for the data placement problem related to the NVDIMM/DRAM hybrid memory architecture.

There are several future works. The important one is an experiment on cycle-accurate computer architecture simulators. In this paper, the accuracy of the experiment is not sufficient because we did not use any real NVDIMM devices. For a detailed discussion, we need a more accurate evaluation. An experiment on the simulator that reproduces memory access behavior (i.e., cycle-accurate simulator) would be useful for that.

REFERENCES

[1] JEDEC Solid State Technology Association, “JEDEC Annou nces Support for NVDIMM Hybrid Memory Modules”. [Onli ne]. Available from: https://www.jedec.org/news/pressrelease s/jedec-announces-support-nvdimm-hybrid-memory-modules, 2016.11.1.

[2] David A. Patterson and John L. Hennessy, “Computer Organization and Design”, Fourth Edition. Morgan Kaufmann Publishers Inc., 2008.

[3] G. Dhiman, R. Ayoub, R. Tajana, “PDRAM: A Hybrid PRAM and DRAM Main Memory System,” in Proc. of the 46th Annual Design Automation Conference (DAC’ 09), pp. 664–669, 2009.

[4] P. Zhou, B. Zhao, J. Yang, Y. Zhang, “A durable and energy efficient main memory using phase change memory technology,” in 6th Annual International Symposium on Computer Architecture (ISCA ’09), ACM, pp. 14–23, 2009.

[5] W. Zhang and T. Li, “Exploring Phase Change Memory and 3D Die-Stacking for Power/Thermal Friendly, Fast and Durable Memory Architectures,” in Proc. of PACT’ 09, IEEE, pp. 101–112, 2009.

[6] S. M. Blackburn, R. Garner, C. Hoffmann, A. Khang, K. Mckinley, R. Bentzur, et al., “The DaCapo Benchmarks: Java Benchmarking Development and Analysis,” in Proc. of the 21st annual ACM SIGPLAN conference on Object-Oriented Programing, Systems, Languages, and Applications, ACM, pp. 169–190, 2006.

 ISBN: 978-1-61208-529-6

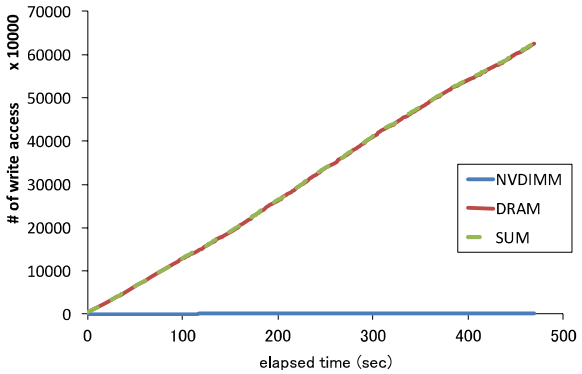

Figure 1. Change of write access

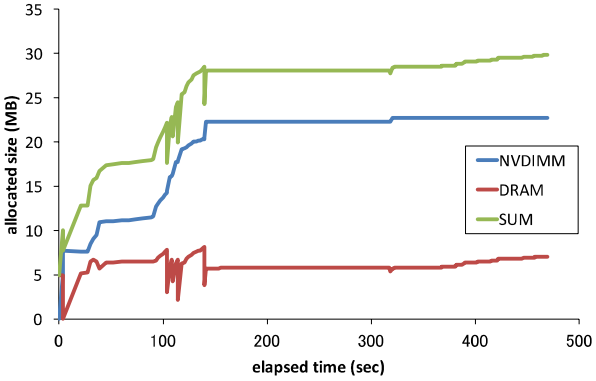

Figure 2. Change of memory allocation size

 ISBN: 978-1-61208-529-6