하이브리드 메모리 시스템에서의 오브젝트 배치를 통한 성능-에너지 조율

김태욱°, Safdar Jamil, 김영재
서강대학교 컴퓨터공학과
{taeugi323, safdar, youkim}@sogang.ac.kr

Performance-Energy Mediating Object Placement on Hybrid Main Memory System

Taeuk Kim, Safdar Jamil, Youngjae Kim
Department of Computer Science and Engineering, Sogang University, Seoul, South Korea

요 약

In large-scale HPC data centers, main memory holds a significant share in power and energy consumption within data centers. Recent developments in main memory such as Hybrid Main Memory System (HMMS) comprising of DRAM and NVM have shown the potential to reduce energy consumption. However, performance degrades in such HMMS due to higher access latency of NVM. In this paper, we propose a novel methodology to optimize the latency in HMMS while satisfying the energy envelope by allocating memory objects on DRAM and NVM respectively. Our methodology considers the object access pattern and the nature of the NVM module to decide the object placement. We model the proposed object placement approach using Integer-Linear Programming (ILP). We show that our methodology reduces 254% of execution time compared to MOCA, the recent study about performance-power efficient object placement, with satisfying strict energy constraint.

1 Introduction

High-performance computing (HPC) applications exploit large amount of resources of data centers, which contributes to severe power and energy consumption. It is known that about 30% of this consumption appears in memory level (DRAM). In reducing power/energy consumption, non-volatile memory (NVM) technologies, such as Spin-Transfer Torque RAM (STT-RAM) and Phase Change Memory (PCM), can be adopted [1]. These NVMs are byte-addressable, have fast read/write accesses, and consume less power/energy than DRAM. Still, access latencies of NVMs are too high to replace DRAM, so various architectures are proposed as Hybrid Main Memory System (HMMS), which consists of DRAM and NVM. In HMMS, placement of memory object becomes a challenging task as it should maintain performance with reducing the energy consumption. Several works [2, 1] have been done to achieve optimal performance with reducing memory level power/energy consumption by placing memory object in HMMS. But, none of the existing studies have considered the characteristics of NVM devices and object access patterns.

In this paper, we propose a methodology to optimize the performance while meeting the energy requirements. Our target applications are HPC applications as they consume more energy. Our approach is based on (i) profiling heap objects access patterns of applications which are based on a two-pass memory profiler [3], (ii) estimating expected energy consumption of allocating object on DRAM and NVM respectively, and finally, (iii) a placement planner based on Integer Linear Programming (ILP) which decides the placement of objects on basis of information extracted from previous steps. With several memory system configurations, our idea shows not only 254% speedup on average, but also satisfying strict energy constraint.

2 Background

2.1 STT-RAM

STT-RAM is an emerging NVM technology as it has lowest access latencies than other NVMs. It is a resistance-based memory device that changes its magnetic direction of resistance layer with forcing high voltage when it stores data to the memory cell. So, the energies of STT-RAM read and write are not equivalent to DRAM. Writing data to STT-RAM is more expensive than DRAM whereas reading data from STT-RAM consumes similar energy as of DRAM. We adopt STT-RAM as NVM candidate because other NVM devices energy consumption is not yet determined properly. STT-RAM per-bit energy consumption is proposed by Kultursay et al. [4] and we adopted that energy model.

A prior research on STT-RAM provides a solution of expensive write problem by exploiting write mechanism [4]. Unlike DRAM, STT-RAM row buffer and sense amplifiers are independent of each other as shown in Fig 1. So updating row buffer cannot change the corresponding memory array row directly. Memory array update occurs when write operation incurs row buffer conflict. Once row buffer conflict occurs, row buffer is written back to corresponding row of array and then target address row is fetched to row buffer, like the write-back cache. Kultursay et al. [4] designs STT-RAM driver to write row buffer to memory array partially; it finds dirty blocks in row buffer and updates only those blocks which significantly reduces the writes on memory array.

2.2 Memory Access Patterns of HPC Applications

HPC applications run in the granularity of days and access memory frequently for computation. The major occupied memory space of HPC applications is heap space that is dynamically allocated to variables. Majorly accessed variables of HPC applications change their size with varying workload.

A recent work analyzes the patterns of memory accesses in object level across various spectrum of applications [3]. It states that most of HPC application variables have regular scaling pattern where 126 out of 127 variables scale or have fixed size with growing workload size. Also, it profiles total accessed volumes, lifetimes, localities in page level and cache line level of variables.

2.3 Related Work

To optimize the performance and minimize energy consumption in HMMS, various works have been done. Unimem [2] discusses

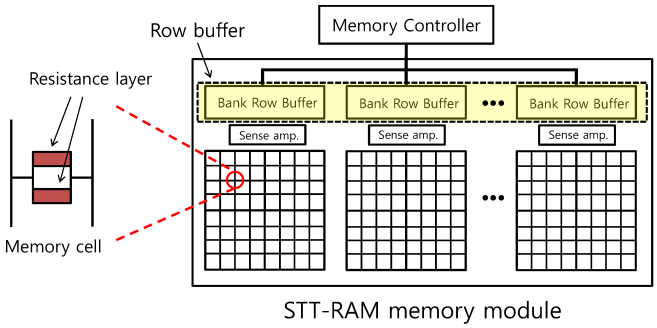


그림 1: Architecture of STT-RAM

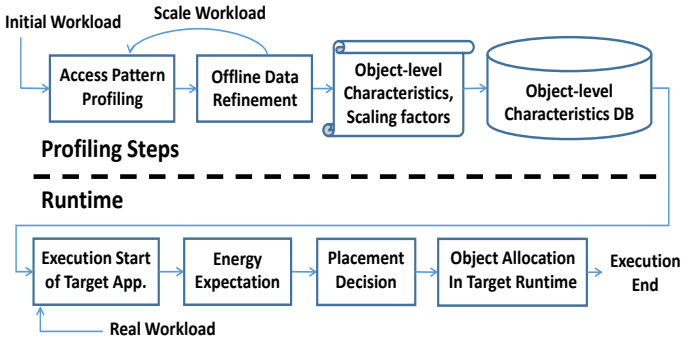


그림 2: System Flow of ePlan Framework

how heap objects should be classified following by their access patterns and then allocates them to appropriate memory device to optimize the performance. But it only considers performance, not energy consumption of HMMS. Also, it does not concentrate on the nature of NVM device. MOCA [1] discusses performance and energy consumption. It classifies memory objects to bandwidth-, latency-, and power-sensitive and allocates the objects to best-fit memory module in ternary HMMS. MOCA considers object level access patterns, but it does not take into account workloads of application and thorough access patterns. Also, it does not consider the characteristics of NVM.

3 Design and Implementation

Motivated by the needs of data center and prior studies on HMMS, we propose a heap object placement policy, ePlan that accomplishes the optimal performance with meeting the required energy limit. We adopt STT-RAM to consider the characteristics of NVM device. The overall system overview of ePlan is shown in Fig 2. It consists of two parts; Profiling steps and runtime. Through profiling steps, it extracts access patterns of target application to get memory object characteristics. In runtime, it estimates the energy consumption with profiled information and decides the allocation of objects.

3.1 Access Pattern Profiling

This module extracts object access patterns using two-pass memory profiler [3] which consists of online and offline profiling of the application. When application runs for the first time, profiler instrument on instruction level and record object characteristics such as lifetime, read/write sequentiality, access volume, and spatial/temporal localities. But profiling application is expensive and it comes with its own overhead of instrumentation. Thus, we suggest using database to store profiling results of the application. In the database, scaling factors of various workload will be stored and that scaling factors will be used to calculate the expected energy consumption.

3.2 Energy Consumption Prediction Modeling

After extracting object access metadata for application, this module calculates the expected energy consumption of object. We adopt a per bit energy model for DRAM and NVM on the basis of prior work [4]. Below equations shows the energy consumption of i-th object in DRAM (DE_i) and STT-RAM (SE_i) and Table 1 describes notations of equations.

$$DE_i = dE_{AP} \cdot AV_i + dE_{RW} \cdot AV_i + dE_{REF} \cdot SV_i \cdot L_i$$

$$SE_i = sE_{AP} \cdot AV_i + sE_{RBA} \cdot AV_i + sE_{WB} \cdot N_{DC} \cdot V_{CL}$$

3.3 ePlan: Performance-Energy Mediating Placement

This section explains our object allocation approach, ePlan, to mediate the performance and energy which decides optimal placement of objects in HMMS while meeting energy requirement. We propose an Integer-Linear Programming (ILP) algorithm which considers three constraints for placement.

표 1: Notations used in Modeling

Notations	Description
dE_{AP}	DRAM activate/precharge energy
AV_i	*i*th object access volume
dE_{RW}	DRAM read/write energy
dE_{REF}	DRAM refresh energy
SV_i	*i*th object allocated volume
L_i	*i*th object lifetime
sE_{AP}	STT-RAM activate/precharge energy
sE_{RBA}	STT-RAM row buffer access energy
sE_{WB}	STT-RAM writeback energy
N_{dc}	Number of dirty cachelines
V_{CL}	Cache line Size (64 byte)
C_D	DRAM capacity
C_S	STT-RAM capacity

I *Decision Constraint:* We set the variable X_i of ILP as an adequate memory module to allocate i-th object. In our binary HMMS, all X_i must be one or zero, which represent DRAM and STT-RAM respectively.

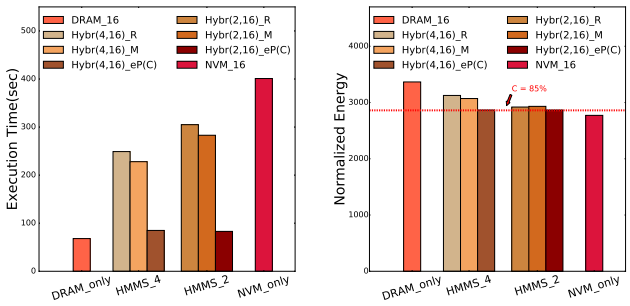
$$0 \leq X_i \leq 1 \quad \text{for } i = 1, 2, ..., N$$

II *Capacity Constraint:* Object placement is also affected by the capacities of memory devices. Objects allocated in HMMS, should not exceed the capacities of memory modules.

$$\sum_{i}^{N} X_i \cdot S_i \leq C_D$$

$$\sum_{i}^{N} (1 - X_i) \cdot S_i \leq C_S$$

Above equations guarantee that the total objects placed on each memory module does not exceed the capacity constraint.



(a) Execution Time (b) Expected Energy

그림 3: Performance and energy consumption on various configurations. In X-axis, DRAM_only and NVM_only are single device memories. HMMS_*w* indicates binary HMMS whose DRAM size is *w* GB. In detail, Hybr(*x*,*y*)_*p* means the placement *p*={R, M, eS(C)} on HMMS which consists of *x* GB DRAM and *y* GB NVM.

X_i are the decision constraint and S_i are the size of *i*-th object.

III *Energy Constraint:* Objects placed in HMMS should not exceed the required energy consumption. The formalization for energy constraint is given below where DE_i and SE_i are energy consumptions of i-th object in DRAM and NVM respectively, derived in section 3.2.

$$\sum_{i}^{N} \{X_i \cdot DE_i + (1 - X_i) \cdot SE_i\} \leq \sum_{i}^{N} DE_i \cdot R$$

With these constraints, our ILP algorithm pursues to minimize the latency in accessing objects.

$$\sum_{i}^{N} \{L_{DRAM} \cdot X_i \cdot AV_i + L_{STTRAM} \cdot (1 - X_i) \cdot AV_i\}$$

4 Evaluation

We evaluate the efficiency of ePlan with various placement methods in terms of execution time and energy consumption. We emulate the binary HMMS using Quartz [5] in 32GB memory system and our target application is the BFS in PBBS benchmark [6] with 6.7GB workload. Due to the lack of energy-measuring equipment, we use estimated energy in evaluation. To consider the practical energy consumption, we set the major object as heap variable whose size is larger than 4KB.

In Fig 3, the *DRAM_16* and *NVM_16* indicate that all major objects are allocated to 16GB of DRAM and NVM respectively, which is the upper and lower bounds of time and energy. To consider the capacity constraint, we use two configurations on HMMS; one is (DRAM, NVM) = (4, 16) GB and other is (DRAM, NVM) = (1, 16) GB. Our evaluating placements are Random (*R*), MOCA (*M*) and ePlan(C) (*eP(C)*). *R* allocates object randomly by only considering capacity constraint. *M* is based on our counterpart MOCA [1], which allocates object with considering capacity, performance and energy. *eP(C)* is our proposed approach where *C* is the percentage of energy constraint for *DRAM_16*. For these experiments, we select *C* as 85 empirically.

Fig 3.(a) shows that for execution time *eP(C)* performs 293%, 268% faster than *R*, *M* in *HMMS_4* and 367%, 341% faster in *HMMS_2*, which is nearly similar to *DRAM_only*. As DRAM portion decreases in HMMS, more objects are allocated to NVM and performance degrade in *R* and *M*. But, ePlan smartly selects less effective objects in latency to place in NVM and avoid the performance degradation. Besides, in Fig 3.(b), only ePlan fulfills given energy constraint *C* in all cases. *R* does not consider energy consumption in placement. *M* takes energy into account in allocation, but still it does not satisfy the energy requirement.

5 Conclusion & Future Work

In this paper, we present ILP based object placement algorithm on HMMS which optimizes performance while meeting the required energy consumption. Through systematic experiments, it is shown that our algorithm satisfies energy requirements while optimizing performance than other placement candidates. The current placement is static and with workload changes, we have to recompute the placement decision. Dynamic placement of objects in HMMS is our future work.

Acknowledgment

This research was supported by Next-Generation Information Computing Development Program through National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT (2017M3C4A7080243).

참고 문헌

[1] A. Narayan, T. Zhang, S. Aga, S. Narayanasamy, and A. K. Coskun, "MOCA: memory object classification and allocation in heterogeneous memory systems," in *IPDPS*, pp. 326–335, IEEE Computer Society, 2018.

[2] K. Wu, Y. Huang, and D. Li, "Unimem: Runtime data managementon non-volatile memory-based heterogeneous main memory," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '17, pp. 58:1–58:14, ACM, 2017.

[3] X. Ji, C. Wang, N. El-Sayed, X. Ma, Y. Kim, S. S. Vazhkudai, W. Xue, and D. Sanchez, "Understanding object-level memory access patterns across the spectrum," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '17, pp. 25:1–25:12, ACM, 2017.

[4] E. Kultursay, M. Kandemir, A. Sivasubramaniam, and O. Mutlu, "Evaluating stt-ram as an energy-efficient main memory alternative," in *2013 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, vol. 00, pp. 256–267, 04 2013.

[5] H. Volos, G. Magalhaes, L. Cherkasova, and J. Li, "Quartz: A lightweight performance emulator for persistent memory software," in *Proceedings of the 16th Annual Middleware Conference*, Middleware '15, pp. 37–49, ACM, 2015.

[6] J. Shun, G. E. Blelloch, J. T. Fineman, P. B. Gibbons, A. Kyrola, H. V. Simhadri, and K. Tangwongsan, "Brief announcement: The problem based benchmark suite," in *Proceedings of the Twenty-fourth Annual ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '12, pp. 68–70, ACM, 2012.