Energy-Efficient Hybrid DRAM/NVM Main Memory

Ahmad Hassan
SAP Labs
Belfast, United Kingdom
ahmad.hassan@sap.com

Hans Vandierendonck
Queen's University of Belfast
Belfast, United Kingdom
h.vandierendonck@qub.ac.uk

Dimitrios S. Nikolopoulos
Queen's University of Belfast
Belfast, United Kingdom
d.nikolopoulos@qub.ac.uk

1. INTRODUCTION

Energy consumption has become a major concern in high-end computing systems. One key energy consumer is main memory, which is increasingly important due to growing data set sizes. Increasingly large amounts of data are stored in main memory of servers for applications like in-memory analytics, system modeling and simulations. DRAM, the de facto technology for main memory, consumes continuous static power (leakage and refresh) and is unlikely to scale beyond 22 nm in the future. Emerging non-volatile memory (NVM) technology holds promise to augment or replace DRAM in the memory hierarchy. NVM offers near-zero static power (negligible leakage and zero refresh) and higher density. However, NVM technology has several weaknesses which open new challenges for adopting NVM in existing systems. Due to increased latency and energy per memory access, NVM cannot be used as a drop-in replacement of DRAM.

This work proposes a hybrid memory system, consisting of DRAM and NVM, to reduce overall energy of the memory system. We demonstrate the feasibility of application-level data management on hybrid memory system. We demonstrate that hybrid NVM / DRAM main memory is energy-efficient for memory sizes commonly employed in high-end compute systems. We develop first-order performance and energy models of memory systems and apply them to a wide variety of applications ranging from enterprise-level in-memory data stores to embedded applications. However, our proposed models are generic and equally applicable to high performance computing applications (HPC). We apply models at the fine granularity of application objects and demonstrate that application-level object placement admits more energy-efficient solution than OS-level page placement.

2. CONTRIBUTIONS

We make the following research contributions in this work:

- Data placement policies for energy-efficient hybrid DRAM / NVM memory.
- Performance and energy models to derive data placement.
- Programming interface and API to allocate memory on hybrid main memory from an application.
- Tools for quickly analyzing any source code and deriving placement policies.

3. RESEARCH OUTCOME

This research has produced the following publications.

1. Energy-Efficient Hybrid DRAM / NVM Main Memory (Extended Abstract) [1].
2. Energy-Efficient In-Memory Data Stores on Hybrid Memory Hierarchies [2].
3. Software-managed Energy-efficient Hybrid DRAM / NVM Main Memory [3].
4. On The Energy-Efficiency of Byte-Addressable Non-Volatile Memory [4].

Two patents have been filed under US patent office:

1. Analytical models and techniques for Software-Managed Energy-Efficient Hybrid DRAM/NVM Main Memory. *US Patent ID 83059367.*
2. A novel technique to allocate memory on multiple types of main memory technologies from software application layer. *US Patent ID 83093314.*

4. CHALLENGES

DRAM technology has hit scaling and power barriers [5, 9]. The ability of DRAM technology to scale below 22 nm feature sizes is yet to be confirmed [5]. With significant leakage power and high refresh power, DRAM-based main memory consumes 30-40 % of the total server power [7, 6]. The DRAM size directly influences the power consumption

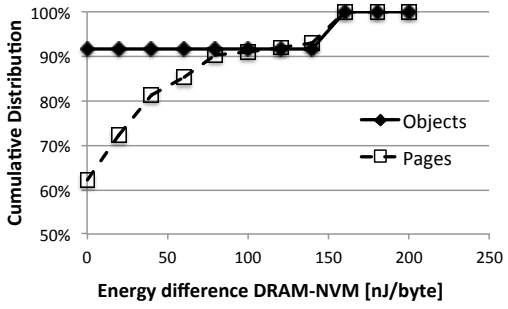

Figure 1: Histogram of DRAM energy minus NVM energy for objects and pages of the jpeg benchmark.

of the servers.

Non-Volatile Memory (NVM) is emerging as a compelling main memory technology due to high density and low leakage power. Various novel NVM technologies have been proposed over recent years, each with different strengths and weaknesses in energy, performance, durability, density and scalability, and each with different likelihoods of making it to mass production. The main contenders are Phase Change Memory (PCM), Spin Transfer Torque memory (STT-RAM) and Memristors [8]. These technologies are byte addressable and exhibit low leakage power and zero refresh power [9]. As such, NVM is promising to augment DRAM in main memory. However, NVM has several weaknesses. Reading and writing often takes longer than for DRAM and consumes more energy, with writing suffering more than reading [17, 9]. There is an asymmetry in read and write cost (e.g., PCM read and write latency is approximately 4.4× and 12× times DRAM latency). Similarly, dynamic energy of PCM read and write is approximately 2× and 43× of DRAM respectively [17, 9].

The challenge is how to manage data on a hybrid DRAM / NVM memory system. Such system shall rarely expose the high latency and dynamic energy of NVM and use NVM to increase the capacity of main memory. This work addresses this challenge by proposing data management policies at the fine granularity of application objects. We demonstrate that our approach is more energy-efficient than current state-of-the-art approaches.

5. MOTIVATION

The state-of-the-art proposes various hybrid memory solutions that are operated by the hardware or the OS. These solutions try to second-guess the properties of the workloads and migrate large chunks of data, typically at the page granularity of the virtual memory system, between DRAM and NVM. This introduces runtime overhead and energy consumption due to the monitoring and the migration.

The success of hybrid memory depends on the presence of policies to place data on the appropriate type of memory. Prior research has proposed distinct solutions to manage data in hybrid memories. This work proposes software management policies for placement of data in a hybrid main memory at the fine granularity of application-level objects, which are individual program variables or memory allocations. Figure 1 shows why operating on objects leads to higher energy savings. The dashed curve shows the cumulative distribution function of the difference in energy consumed for a 4 KB page when it is stored on DRAM in comparison to storing the page on RRAM. 61 % of pages incur no extra energy, while around 8 % of pages are clearly hot. This leaves 31 % of pages in the gray zone. Likewise, the solid curve shows that 91 % of objects are clearly cold and incur no added energy, while 9 % of objects are clearly hot. This analysis shows that objects are a better granularity to decide placement in a hybrid memory hierarchy, as they are strongly biased towards one particular memory technology. Moreover, this strong bias reduces the need for migration of objects between memories. This work is motivated by the superiority of application-level object placement over OS-level page placement for energy-efficient hybrid memory system.

6. CURRENT APPROACHES

OS-level memory management identifies main memory access properties of the application at the granularity of virtual memory pages. The placement of virtual memory pages on hybrid memory is sub-optimal when objects with different access frequencies are placed on the same page. However, the main benefit of OS-level page placement is that it is transparent to the application. Shin *et al* [11] show 36 % energy savings by collectively migrating pages with similar access characteristics between DRAM and PCM. Ramos *et al* [10] propose a Rank-based Page Placement (RaPP) policy which uses multi-level queues to rank pages within the memory controller. That information is fed into OS which triggers page migration.

At the ***hardware level***, Qureshi *et al* [9] propose a write cancellation policy. By canceling writes and re-doing them after reads complete, performance increases by 3× and PCM lifespan increases from 3 to 9.7 years. Lee *et al* [17] use DRAM as a buffer to hide the latency of PCM. The use of DRAM buffer reduces the execution time from 1.6× to 1.2× and dynamic energy from 2.2× to 1.1× for an application in comparison to PCM-only system. Cho *et al* [12] replace a write operation with read-modify-write operation (*Flip-N-Write*) to improve write endurance, write energy and write bandwidth of PCM.

Application-level data placement requires programmer intervention. Li *et al* [18] propose an OS and compiler-assisted approach to identify write intensive parts of the application. Compiler assisted data placement reduced power consumption by 9.8 %. Prior research also proposes user level tools and techniques for exploiting NVM persistence. Coburn *et al* [15] proposed NV-heap, a user-level persistent object abstraction that provides the building block for implementing any persistent data structure such as trees and hash tables. Closest to our work is the work of Li *et al* [13] who explore the use of NVM for scientific applications by estimating what application objects to store in NVM. They identified the connection between the context of data in data structures with the read-only access behavior. They demonstrate power savings up to 27 %, compared to a DRAM-only system. The heuristic used by Li *et al* identifies almost exclusively read-only data suitable for NVM. Our work presented here disputes this conclusion. We evaluate the suitability of storing both read and write-dominated objects on NVM using analytical models of performance and energy.

7. PROPOSED SOLUTION

We propose a methodology for deciding the placement of objects in a hybrid memory system, i.e., placing object either on DRAM or NVM. We develop an application analysis method and tool to evaluate the opportunities for application-

Table 1: Comparison of an NVM-only system (NVM), SWP-CV, SWP-SV and RaPP.

	CPI Increase (%)				Energy Savings (%)			
Benchmark	**NVM**	**SWP-CV**	**SWP-SV**	**RaPP**	**NVM**	**SWP-CV**	**SWP-SV**	**RaPP**
CJPEG (CJ)	18.44	2.90	3.54	5.37	-779.3	81.20	86.60	77.19
Consumer Lame (LM)	91.53	1.80	1.16	14.10	-1375.80	69.13	74.30	73.70
Ghostscript (GS)	47.83	2.36	2.69	12.30	-398.70	80.15	86.03	79.80
Astar (AS)	35.86	1.40	0.67	17.93	79.70	77.84	82.20	77.14
Hmmer (HM)	50.79	2.66	2.04	15.54	68.30	71.47	72.30	63.00
bzip2 (BZ)	29.32	2.10	1.29	6.61	81.20	81.51	85.00	82.60

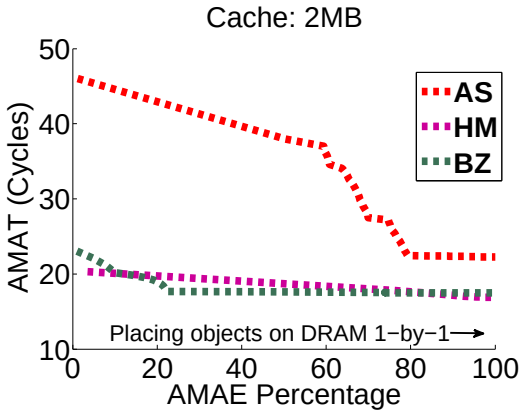

Figure 2: Placing data incrementally in the memory hierarchy and its impact on AMAT and AMAE. The left-most (right-most) point on each curve corresponds to NVM-only (DRAM-only) memory.

level management of hybrid memory systems. Our method uses the output of the profiling tool and evaluates first-order performance and energy models on that data. At the same time, the methodology allows us to calculate appropriate sizes for DRAM and NVM for these workloads. Moreover, we present a programming API in the same section and argue that such an API is feasible to implement in modern OSes. Finally, by using cycle-accurate validation, we demonstrate that application-level, profile-guided object placement in hybrid memory can achieve higher energy savings than state-of-the-art OS-level approaches.

7.1 Tool

We devise a method to evaluate the application-level data management on hybrid main memory system. We implemented a tool [3] to instrument application-defined objects, OS pages and cache blocks. The tool can be easily applied to a wide range of application as it does not require modifications to the source code. The tool measures total memory access, off-chip accesses and lifetime of each object. An object maps one-to-one to global variables, stack-allocated variables and dynamically allocated memory (*malloc* and *mmap*) of an application.

7.2 Evaluation Setup

We validate the software data placement methodology using GEM5 [16] cycle-accurate simulation. We simulate ARM Cortex-A7 CPU for embedded benchmarks and X86 CPU for SPEC and in-memory data stores. We evaluate a representative set of applications from MiBench benchmark suite, SPEC CPU2006 benchmarks. We also validate against an in-memory database and in-memory key/value store. We simulate a hybrid memory with two memory controllers. One corresponding to DRAM and other for RRAM. The timing parameters for RRAM are derived from the prior literature [17]. Specifically, we derived the tRCD, tRP, tRRDpre and tRRDact parameters for RRAM following Lee *et al* [17]. We configure GEM5 with 1 GB RRAM and 128MB of DRAM. We modified GEM5 memory port in order to route packets to appropriate memory using the virtual addresses from within the application.

7.3 Data Placement

We identify the opportunities of data partition into frequently and rarely-accessed objects. Using the memory access frequencies for each object we derive the placement of each object on the hybrid memory. Data placement in a hybrid memory allows to minimize energy by placing frequently accessed (hot) objects in DRAM and rarely accessed (cold) objects (especially large ones) in NVM. By frequent accesses, we mean the accesses to main memory (off-chip). In our analytic models, we calculate average memory access time (AMAT) and average memory access energy (AMAE) of each object when it is placed on DRAM and NVM. We generate a sorted list of all application objects where application with most off-chip accesses are on the top of the list.

Figure 2 plots for each object o_i in position i in the sorted list the cumulative values $\Sigma_{j=1}^{i}\Delta AMAE(o_j)$ (X-axis) and $\Sigma_{j=1}^{i}\Delta AMAT(o_j)$ (Y-axis). From left to right, the curves show the transition in AMAT and AMAE when an object or a page is moved one-by-one from NVM to DRAM. The left-most point on the curve represents an NVM-only memory system as all objects are placed on NVM. The right-most point on the curve represents a DRAM-only memory system. Figure 2 shows that moving more objects onto DRAM results in an energy increase as the active amount of DRAM grows, but memory access latency shrinks as DRAM has lower latency. Figure 2 shows that only a small number of objects are hot while the remaining objects have least influence on AMAT. Tables 1 provides the comparison of DRAM-only system with NVM-only, SoftWare Placement Self Validation (SWP-SV), SoftWare Placement Cross Validation (SWP-CV) and RaPP. Both SWP-CV and SWP-SV out-performs RaPP on both performance (CPI) and energy.

Comparison of cross and self validation confirms that our methodology works well for identifying the most important objects of the application regardless of the input size. The access characteristics of each object remains constant for different inputs. We also evaluated the impact of cache size on object placement decisions. We use 5 different cache configurations to find out the variation in placement decision. Our results confirm that the cache size, used during the profiling phase, has no impact on the placement of objects on hybrid memory.

7.4 Programming interface and API

We split the virtual address range in two portions, one each corresponding to DRAM and NVM. This is a feasible solution as 64-bit virtual address spaces are large enough so as not to overflow either partition of virtual memory. We extend the physical memory map in BIOS with a flag to mark distinct entries for the DRAM and NVM memories. We propose a lightweight extension to the mmap() system call where a flag is added to allocate memory on either DRAM or NVM. This choice is extended to the user-level allocation library, where the *malloc* calls are extended.

8. SCOPE OF PROPOSED SOLUTION

Our solution is not restricted to any particular class of application. Our methodology, tools and models are generic and equally applicable to all classes of applications ranging from embedded to high performance computing applications. We applied our methodology to four types of applications. We validated against two classes of application that are widely used in today's data centers: an in-memory database and an in-memory key-value store [2]. We further validated our methodology on MiBench and SPEC benchmark suites [3].

9. CONCLUSION

Non-volatile memory is crucial in order to sustain the power demands of high-end exascale systems. NVM technology provides new ways of overcoming the power challenges of DRAM. We have investigated the feasibility of orchestrating data placement and migration for hybrid memory at the application level. We compared the data management of application objects against pages (a granularity of data that may be used by the OS or hardware) and cache blocks (a granularity that may be used by hardware). We found that placing application objects on hybrid memory yields more energy savings than state-of-the-art OS-level page migration or hardware approaches.

We propose tool for deriving placement, performance model, energy model and a programming API that allows programmers to control data placement in hybrid memory. We conclude that NVM is an attractive technology to augment DRAM in the main memory system.

10. REFERENCES

[1] A. Hassan, H. Vandierendonck and D. Nikolopoulos. Energy-Efficient Hybrid DRAM/NVM Main Memory (Extended Abstract). In *PACT*, 2015.

[2] A. Hassan, H. Vandierendonck and D. Nikolopoulos. Energy-Efficient In-Memory Data Stores on Hybrid Memory Hierarchies. In *DaMoN*, pp. 1:1–1:8, 2015.

[3] A. Hassan, H. Vandierendonck and D. Nikolopoulos. Software-managed Energy-efficient Hybrid DRAM/NVM Main Memory. In *CF*, pp. 23:1–23:8, 2015.

[4] H. Vandierendonck, A. Hassan, and D. Nikolopoulos. On the energy-efficiency of byte-addressable non-volatile memory. *Comput. Archit. Letters*, PP(99):1–1, 2014.

[5] ITRS. *International Technology Roadmap for Semiconductors*, 2011.

[6] C. Lefurgy *et al*. Energy management for commercial servers. In *Computer*, 36(12):39–48, 2003.

[7] L. A. Barroso and U. Holzle. The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines. In *Synthesis Lectures on Computer Architecture*, 2009.

[8] Y. Ho, G. M. Huang, and P. Li. Nonvolatile memristor memory: Device characteristics and design implications. In *DAC*, pp. 485–490, 2009.

[9] M. K. Qureshi, V. Srinivasan, and J. A. Rivers. Scalable high performance main memory system using phase-change memory technology. In *ISCA*, pp. 24–33, 2009.

[10] L. E. Ramos, E. Gorbatov, and R. Bianchini. Page placement in hybrid memory systems. In *ICS*, pp. 85–95, 2011.

[11] D.-J. Shin *et al*. Adaptive page grouping for energy efficiency in hybrid PRAM-DRAM main memory. In *RACS*, pp. 395–402, 2012.

[12] S. Cho and H. Lee. 2009. Flip-N-Write: A simple deterministic technique to improve PRAM write performance, energy and endurance. In *MICRO*, pp. 347–357, 2009.

[13] D. Li, *et al*. Identifying opportunities for byte-addressable non-volatile memory in extreme-scale scientific applications. In *IPDPS*, pp. 945–956, 2012.

[14] Q. Li, *et al*. Compiler-assisted preferred caching for embedded systems with STT-RAM based hybrid cache. In *LCTES*, pp. 109–118, 2012.

[15] J. Coburn, *et al*. NV-Heaps: making persistent objects fast and safe with next-generation, non-volatile memories. In *ASPLOS*, pp. 105–118, 2011.

[16] N. Binkert, *et al*. The GEM5 simulator. *SIGARCH Comput. Archit. News*, 39(2):1–7, Aug. 2011.

[17] B. C. Lee, *et al*. Architecting phase change memory as a scalable dram alternative. In *ISCA*, pp. 2–13, 2009.

[18] Y. Li, *et al*. A Software Approach for Combating Asymmetries of Non-volatile Memories. In *ISLPED*, pp. 191–196, 2012.