Dynamic Co-Management of Persistent RAM Main Memory and Storage Resources

Ju-Young Jung
Computer Science Department
University of Pittsburgh
juyoung@cs.pitt.edu

Sangyeun Cho
Computer Science Department
University of Pittsburgh
cho@cs.pitt.edu

ABSTRACT

This paper proposes *Memorage*, a novel system architecture that synergistically manages persistent RAM (PRAM) main memory and a PRAM storage device. Memorage leverages the existing OS virtual memory manager to globally manage PRAM resources and to enhance the utilization of the available PRAM resources. Preliminary experimental and analytical evaluation suggests that Memorage can improve the performance of memory-intensive workloads (by 4.6× on average and up to 9.4× under the examined configuration). It also increases PRAM utilization and significantly extends the longetivity of the PRAM main memory (by 8×).

Categories and Subject Descriptors

C.0 [**Computer Systems Organization**]: General—*System architectures*; D.4.2 [**Operating Systems**]: Storage Management—*Allocation/deallocation strategies*

General Terms

Design, performance, management

Keywords

Persistent RAM (PRAM) main memory, PRAM storage device (PSD), performance, write endurance

1. INTRODUCTION

Researchers are actively exploring the use of emerging persistent RAM (PRAM) technologies such as phase change memory (PCM) to provide a DRAM replacement [1–3] or a fast storage medium [4]. However, most prior work emphasizes only one aspect of PRAM with regard to system design—random accessibility (main memory) or persistency (storage system)—and maintains the traditional main memory and storage dichotomy.

For best performance, a future platform may incorporate a large capacity of PRAM to construct the main memory and the system storage together, directly interfaced via the main memory bus. We observe:

• **There will be little characteristic distinction between main memory resource and storage resource.** Although the main memory and storage may employ different PRAM technologies (e.g., single-level cell PCM for main memory and multi-level cell PCM for storage), there is no significant gap in their characteristics like DRAM and HDD in today's systems.

• **Reducing software latency becomes more critical.** A 4kB data transfer between a PRAM storage device (PSD) and main memory may be done in a sub-microsecond range. It is of critical importance to re-architect the HDD-optimized OS I/O stack to reflect this change.

• **Storage density grows exponentially and a typical user underutilizes the available storage capacity.** The whole capacity of a storage device is rarely filled up, leaving some space unused during its lifetime. Agrawal et al. [5] measured the file system fullness and quantified the annual file system size growth rate to be only 14% on average over a five-year period in a corporate environment. Provisioned but unused storage capacity is, in a sense, a lost resource.

2. MEMORAGE

Memorage advocates federated management of all PRAM resources to improve their utilization for performance and lifetime reliability. The principles behind Memorage are:

1. Don't swap, give more memory: Under high memory pressure, VMM in a conventional system tries to swap out some allocated pages to relieve the pressure. With Memorage, main memory borrows memory capacity from PSD dynamically, effectively eliminating many major page faults which mandate to run the slow-path codes of fault handling.

2. Don't pay for physical over-provisioning: PRAMs with limited write endurance (like Flash and PCM) typically require additional physical capacity that is "hidden" from the user. With Memorage, PSD flexibly donates its free pages to main memory to relax the write endurance problem, achieving "logical" over-provisioning.

3. Don't stop utilizing nearly dead memory capacity: In Memorage main memory and storage have the same building block. This implies that the non-volatility of main memory can be utilized for maintaining the effective storage capacity while keeping main memory fresh. When some PRAM capacity reaches its write endurance limit, it is risky to continue using the capacity. However, the data written to the capacity before the endurance limit can still be retrieved. Memorage can map read-only file data to such PRAM capacity, even if the capacity was originally within the main memory boundary.

We have obtained preliminary results that highlight the benefits of the principles. Figure 1 illustrates how the performance of eight memory-intensive SPEC benchmark ap-

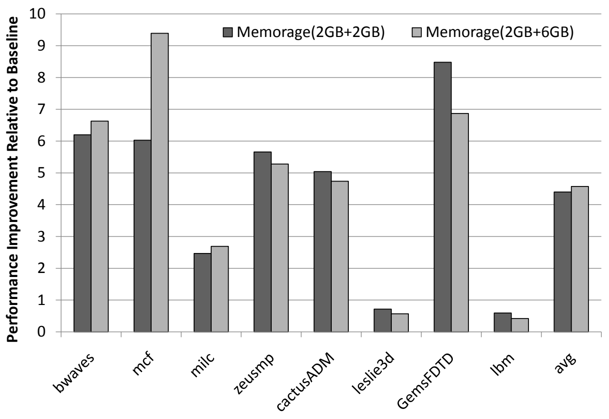

Figure 1: Potential performance improvement with Memorage. The baseline configuration is a conventional system (Intel Core i7 920 @2.66 GHz) with 2 GB memory backed by a 7,200-RPM HDD. The evaluated Memorage configurations have 2 GB memory and borrow from PSD up to 2 GB more (hence exposing 4 GB total) and 6 GB more (total 8 GB). The PSD capacity is 50 GB. All benchmark programs were run simultaneously.

plications, when run together, is improved by Memorage. It is clearly shown that Memorage's capability to temporarily grant more physical memory capacity (borrowed from PSD) helps significantly improve the performance of most applications. Interestingly, leslie3d and lbm performed (slightly) better with the smallest memory capacity; they have a relatively small memory footprint and hog more CPU cycles while other co-scheduled applications are blocked waiting for memory allocation.

Next, we develop a simple analytical framework to reveal the advantage of Memorage in terms of the main memory lifetime. We incorporate memory capacity (C_m), storage capacity (C_s), memory write bandwidth (B_m), and storage write bandwidth (B_s) as input to our model. The two key parameters (or knobs) are: $\alpha = C_s/C_m$ (capacity ratio of storage to memory) and $\beta = B_m/B_s$ (bandwidth ratio). We obtain the lifetime with Memorage (L_{new}) relative to the main memory lifetime without Memorage (L_m):

$$L_{new} = L_m \cdot \frac{(1+\alpha)\cdot\beta}{(1+\beta)} \qquad (1)$$

In the above $L_m = E \cdot C_m/B_m$ where E is the write endurance limit. Based on the formulation, Figure 2 illustrates how the lifetime of PRAM main memory can be improved with Memorage. It is shown that even if the storage bandwidth is very high and matches the memory bandwidth, the lifetime of main memory is increased by 8× and 16× when α is 15 and 30, respectively.

Finally, the overall PRAM resource utilization (U_{pram}) is defined as follows:

$$U_{pram} = \frac{Utilized}{Installed} \qquad (2)$$

where *Utilized* and *Installed* express the amount of utilized PRAM capacity and the total capacity of PRAM installed in the system.

According to Equation (2) and based on the machine configuration of Figure 1 and a conservative assumption of 50% disk fullness, the PRAM utilization of a conventional system

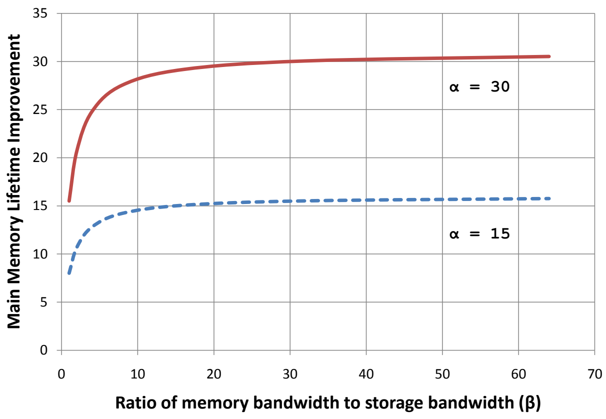

Figure 2: Main memory lifetime improvement (Y axis) with Memorage. Two curves with a different α value represent two different memory-storage capacity configurations (e.g., 16 GB main memory, 240 GB PSD vs. 480 GB PSD).

is 51.9% (= (2 GB + 25 GB)/(2 GB + 50 GB)), since the installed main memory is fully utilized under memory shortage but only half of the PSD resource in the system is used. Meanwhile, the PRAM utilization of a Memorage system will be 63.5% (= (2 GB + 25 GB + 6 GB)/(2 GB + 50 GB)), showing an increase of 11.6% point. Notice that this degree of resource utilization improvement will be higher if we have more available PRAM resources to donate in PSD—a likely situation at early periods of storage lifetime and with a higher capacity PSD that exploits advanced PRAM technologies in the future.

3. FUTURE DIRECTIONS

This work proposed Memorage, a novel system architecture that synergistically co-manages the main memory and the storage resources comprised of PRAM. As a next step, we will investigate design and implementation issues that arise when realizing the Memorage philosophy in a real system. We plan to explore feasible strategies to implement the Memorage architecture in a commodity Linux kernel and evaluate the prototype system in terms of relevant metrics including the system-level performance, resource utilization, lifetime, and energy consumption. In addition, we plan to provide a set of interfaces that allow applications or system software programmers to explicitly control the PRAM resources in a Memorage system.

4. REFERENCES

[1] P. Zhou et al., "A Durable and Energy Efficient Main Memory Using Phase Change Memory Technology," *ISCA'09*, 2009.
[2] B. C. Lee et al., "Architecting Phase Change Memory as a Scalable DRAM Alternative," *ISCA'09*, 2009.
[3] M. K. Qureshi et al., "Scalable high performance main memory system using phase-change memory technology," *ISCA'09*, 2009.
[4] J. Condit et al., "Better I/O through byte-addressable, persistent memory," *SOSP'09*, 2009.
[5] N. Agrawal et al., "A Five-Year Study of File-System Metadata," *TOS*, 3/3, 2007.