CSE530 Project Final Report
Bandwidth-Aware Memory Hierarchy Design with Hybrid Memory Technologies

Cong Xu and Jishen Zhao
{czx102, juz138}@psu.edu

Abstract

Chip-multiprocessor (CMP) is a promising solution for high performance computing systems. However, the limited increases in main memory bandwidth results in a growing bandwidth gap between the processor cores and the main memory, which becomes a potential obstacles to system performance improvement. In this project, we explore bandwidth-aware memory hierarchy design with various emerging memory technologies. Based on the observations obtained from our pre-exploration of various emerging memory technologies, we evaluate systerm performance obtained with various cache hierarchy configurations. With different applications, we evaluate the optimal shared cache configurations in a CMP system, in terms of the number of shared cache levels, memory technology and capacity of each level. We show that cache hierarchy with hybrid memory technologies benefits system performance with non-write-intensive applications.

1 Introduction

Many of modern chip-multiprocessors are designed to perform well on various applications to achieve high performance by exploiting their inherent parallelism. Such systems support large number of threads and single instruction multiple data (SIMD) execution, which puts a lot of pressure on the memory system. Typically, memory latency is not a bottleneck, since the latency can be hidden via multi-threading or hardware prefetching. To this end, bandwidth becomes a potential bottleneck. A high rate of computing often brings in a high rate of data transitions. In some cases, the working set of an application fits in the on-die caches, which can typically provide sufficient bandwidth to keep up with the processing cores. However, if the working set does not fit in the on-die caches, the main memory needs to provide much of the data. Since the bandwidth of off-chip main memory is quite limited, applications with such working sets are potentially bandwidth-bound. Therefore, it is crucial to design the memory hierarchy to overcome the bandwidth limitation.

It is known that performance improvement of a computing system can be achieved via multiple memory levels. Adding an extra level of memory, however, can also help alleviate the bandwidth bottleneck of off-chip memory. Specifically, we examine (1) the number of levels in the optimal memory hierarchy, (2) the appropriate memory technology of each level, and (3) the capacity and bandwidth of each level. We will also explore the energy-efficiency of the memory hierarchy design constrain, and the memory hierarchy design within a fixed power budge.

Emerging memory that provides fast random access, high storage density, and non-volatility within one memory technology becomes possible due to the emergence of various new non-volatile memory (NVM) technologies, such as spin-torque-transfer memory (STT-RAM), phase-change memory (PCRAM), and resistive memory (RRAM). Our goal in this project to to leverage these NVM in devising a memory system with multiple layers of the existing memory hierarchy for modern computers, to supply a wide spectrum of final products that cover from a highly latency-optimized microprocessor cache to a highly density-optimized secondary storage device.

The report is organized as follows. First, we present the motivation of design a memory system with better memory bandwidth. Second, we make a quick summary of current emerging memory technologies. Then we use our NVsim, a circuit-level NVM model to evaluate the read latency, write latency and dynamic energy of different memory technologies. After that, we perform the cycle-accurate architectural simulation to find the best memory system design for different applications. Finally we conclude our work and propose potential future work.

2 Motivation

Throughput computing involves performing a huge number of calculations with a large amount of parallelism. Throughput computing applications span many domains and are already critical on a variety of platforms from high-performance computing (HPC) machines to commercial servers to client machines.

Systems designed to perform well on throughput computing applications achieve high performance by exploiting their inherent parallelism. These systems support large numbers of threads and/or use wide SIMD execution. This puts a lot of pressure on the memory system. Fortunately, memory latency is typically not a bottleneck since the latency can be hidden via multithreading or hardware prefetching, since the data access patterns are relatively simple. On the other hand, bandwidth is a potential bottleneck.

Many throughput computing applications have inherently large working sets (i.e., tens to hundreds of MB). These are unlikely to fit in conventional on-die SRAM caches for the foreseeable future. Further, unlike more traditional workloads (e.g., those similar to SPEC or TPC benchmarks), many throughput computing applications see a sharp drop in performance once caches are too small to hold their working sets. Thus, these applications are likely to be bandwidth-bound at main memory unless some significant changes are made to the memory hierarchy.

Unfortunately, there are few simple techniques to improve bandwidth efficiency of a system. While there is some work in this area [5], this is insufficient for the large bandwidth requirements of some throughput computing applications. So todays systems that tout good performance for throughput computing applications (e.g., nVidias Tesla) do so by providing large main memory bandwidths via the use of GDDR rather than improving bandwidth efficiency. However, GDDR has fairly strict capacity limits and is much more power hungry than conventional DRAM modules. This reduces its desirability for throughput computing, and makes it a bad choice for general purpose systems trying to improve their throughput computing performance.

Emerging memory technologies such as Magnetoresistive Random Access Memory (MRAM), Phase-Change Memory (PCM), Resistive Random Access Memory (RRAM), embedded DRAM, have shown potential to to be used as on-chip caches and the main memory [13]. Therefore, we explore how to enhance the memory hierarchy from the bandwidth point of view.

3 Emerging Memory Technologies

We first review the technology background three types of non-volatile memories, which are STT-RAM, PCRAM, and RRAM.

3.1 STT-RAM

STT-RAM uses the magnetic property of the material and uses Magnetic Tunnel Junction (MTJ) as its binary storage. As shown in Fig. 1, MTJ contains two ferromagnetic layers and one tunnel barrier layer. The direction of one ferromagnetic layer is fixed, which is called the reference layer, while the direction of the other one can be changed by passing a driving current, which is called the free layer. The relative magnetization direction of two ferromagnetic layers determines the resistance of MTJ. If two ferromagnetic layers have the same directions, the resistance of MTJ is low, indicating a "0" state; if two layers have different directions, the resistance of MTJ is high, indicating a "1" state.

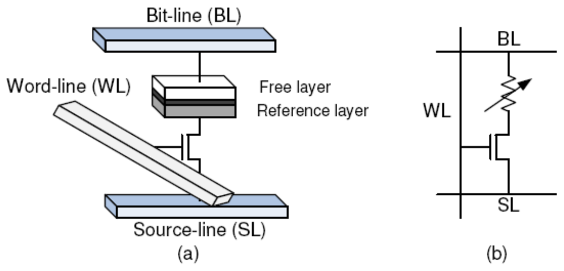


Figure 1: Demonstration of a MRAM cell. (a) Structural view. (b) Schematic view.

3.2 PCRAM

PCRAM uses chalcogenide-based material to storage informations. The chalcogenide-based materials in recent PCRAM research are usually alloys of germanium, antimony, and tellurium (GeSbTe, or GST), which can be switched between a crystalline phase (SET or "1" state) and an amorphous phase (RESET or “0” state) with the application of heat. The crystalline phase shows high optical reflectivity and low electrical resistivity, while the amorphous phase is characterized by low reflectivity and high resistivity.

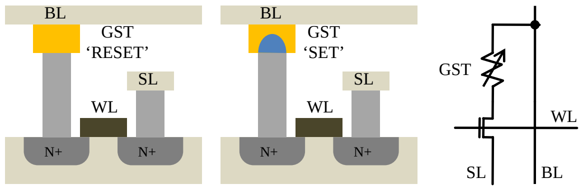


Figure 2: The schematic view of a PCRAM cell with MOSFET selector transistor (BL=Bitline, WL=Wordline, SL=Sourceline

3.3 RRAM

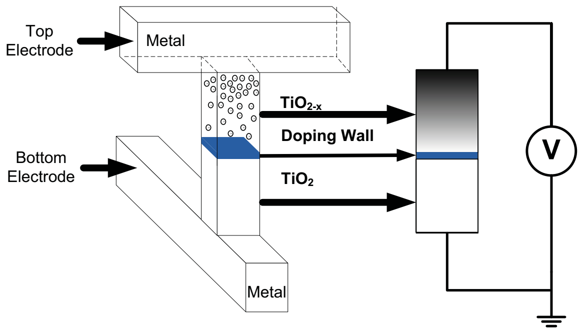
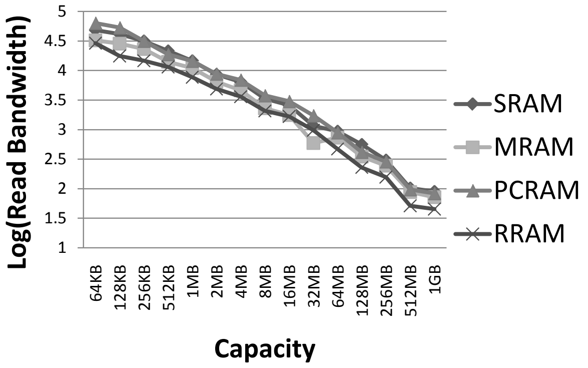


Figure 3: The conceptual view of the structure of memristor cells.

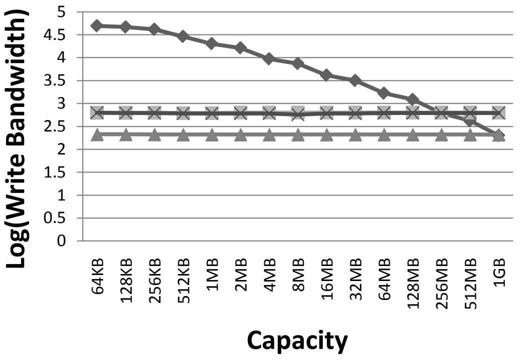
Memristor, a portmanteau of “memory resistor”, is a generalized resistance that maintains a functional relationship between the time integrals of current and voltage. Memristor was first theoretically predicted by Chua in 1971 [2] as the fourth fundamental circuit element from the completeness of relations between the four basic circuit variables, namely, current, voltage, charge, and flux-linkage. The first memristor practical demonstration was presented by Williams *et al.* in 2008 [12]. Fig. 3 shows a conceptual view of the memristor structure [12]. The top electrode and bottom electrode are two metal nanowires on platinum, and the thin titanium dioxide film is sandwiched by the electrodes.

4 Memory Technology Pre-Exploration

Many modeling tools have been developed during the last decade to enable system-level design exploration for SRAM- or DRAM-based cache and memory design. For example, CACTI [6] is a tool that has been widely used in the computer architecture community to estimate the speed, power, and area of SRAM and DRAM caches. In addition, CACTI has also been extended to evaluate the performance, power, and area for STT-RAM [4], PCRAM [3,8], and NAND flash [9]. However, as CACTI is originally designed to model SRAM-based cache, some of its fundamental assumptions do not match the actual NVM circuit implementation, and thereby these CACTI-like estimation tools do not model the NVM array organization in the exact way that the chip is fabricated. In this section, we use *NVSim*, a circuit-level model for NVM performance, energy, and area estimation, which supports various NVM technologies including STT-RAM, PCRAM, RRAM, and conventional NAND flash.



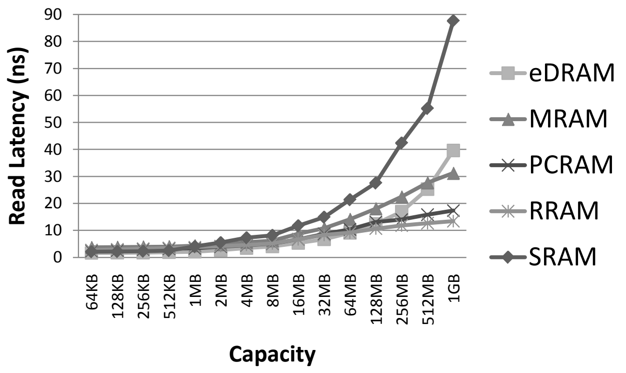
(a)



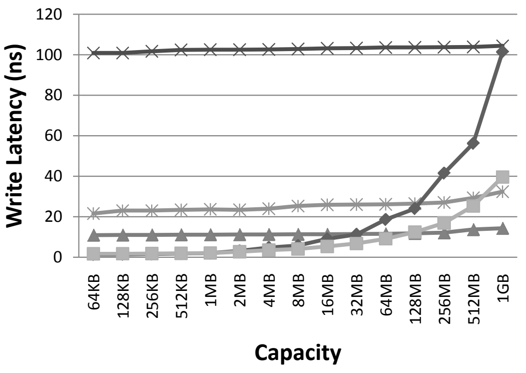
(b)

Figure 4: Read and write bandwidths provided by different memory technologies. (a) Read bandwidth provided by different memory technologes. (b) Write bandwidth provided by different memory technologies.

First of all, we estimate the read and write bandwidths that can be provided by different memory technologies. Figure 4 shows the results, with both x- and y-values in *log* scale. The figure illustrates both the provided read and write bandwidth as a function of memory capacity. Each of the memory technologies actually provide nearly the same read bandwidths, as is shown in figure 4(a). On the other hand, a straight forward observation from figure 4(b) is that the write bandwidth varies among different memory technologies. The shape of the SRAM write bandwidth curve is very similar to the read bandwidth curve. The write bandwidth curves of the other three memory technologies appear to be very different. The reason is that write latencies of the three non-volatile memories are much higher than read latencies. Another observation from figure 4(b) is that the curves cross to each other at different locations.

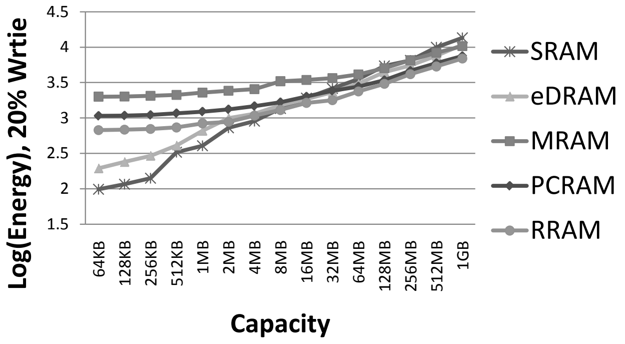


(a)

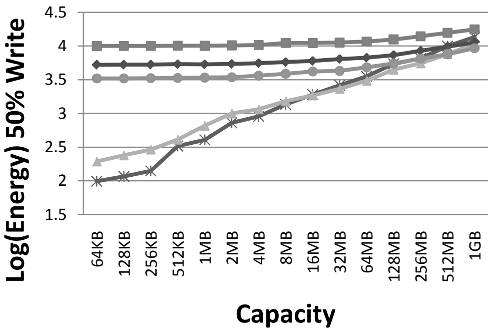


(b)

Figure 5: Latency of different memory technologies. (a) Read Latency. (b) Write Latency.



(a)



(b)

Figure 6: Dynamic energy consumption with the provided bandwidths of different memory technologies. (a) Dynamic Energy with 20% write. (b) Dynamic Energy with 50% write.

Being aware that the read and write latencies of NVM are asymmetric, we consider the read latency at first. In NVsim, we divide the entire cache read latency into the following components:

- (a) H-tree input delay
- (b) Decoder + word-line delay
- (c) Bit-line delay + Sense Amplifier delay

- (d) Comparator Delay (for tag part only)
- (e) H-tree output delay

H-tree latency (a) and (f) are mainly determined by the RC delay of global wires, which is positive proportional to the area of memory macro. Sensing delay (d) is related to the read noise margin of memory cell that is affected by off/on resistance ratio. Figure 5 (a) illustrates the read latency of different memories. We can see that sensing delay dominates the read latency of NVM at small capacity so that PCRAM (with the largest resistance window) is faster than RRAM and MRAM (with the smallest resistance window). While H-tree delay unveils at large capacity so that RRAM (with the smallest cell size) becomes faster than PRAM and MRAM (with the biggest cell size). The read latency of SRAM bank will increase rapidly after 128MB due to large area.

Write latency of NVM are almost dominated by the write pulse width. In this work we assume 10ns, 20ns, 100ns for MRAM, RRAM and PCRAM. While write latency of SRAM and eDRAM is a function of capacity, as similar to the read latency. The results in 5 (b) indicates that NVM is suitable for memory with large capacity.

Figure 6 has demonstrated the dynamic energy of different memory technologies when 20% and 50% write access are assumed. eDRAM will be better than SRAM in terms of dynamic energy after 16MB and this is verified by IBM Power7 L3 cache. The cross-point between NVM and SRAM/eDRAM is postponed for 50% write than 20% write.

Based on these observations, we would like to design a bandwidth-aware hybrid memory hierarchy, which always provides the high memory bandwidth with the given capacity. For example,

- eDRAM has better latency and enegry than SRAM when capacity is larger than 16MB.
- MRAM is more competitive to SRAM and eDRAM when capacity is larger than 128MB.
- PCRAM has serious endurance issue and is targeted as main memory replacement.
- RRAM might fit into cache hierarchy as last level cache replacement when there multiple levels of cache for applications with very few writes.

Table 1: Baseline CMP configuration.

Core	
No. of cores	8
Frequency	1GHz
Core architecture	in-order, 14-stage pipline, 2 ALUs, 2 FPUs
Memory	
Private caches	L1 I-cache: SRAM, 8 x 64KB, 64B line, 2-way, write-through L1 D-cache: SRAM, 8 x 64KB, 64B line, 2-way, write-through
Shared caches	Case 1: Pure SRAM, 64B line, 8-way, write-back Case 2: Hybrid (SRAM, MRAM, RRAM, eDRAM), 64B line, 8-way, write-back
Main memory	4GB

5 Experiments

Based on the parameters of different cache configurations collected from our modified version of CACTI [6], we evaluate both pure SRAM-based and hybrid cache hierarchy designs. In this section, we show how bandwidth-aware memory hierarchy design method help with improving the system performance.

5.1 Experimental Setup

We use Simics [7] as the simulator in our experiments. It is configured to model an eight-core CMP. Each core is in-order, and is similar to UltraSPARC III architecture. The frequency of each core is set to be 1GHz. Table 1 lists the detailed parameters of the baseline. Our baseline contains 8 private L1 instruction and data caches respectively. Since we only evaluate shared cache hierarchies, each of L1 cache is fixed to be SRAM-based, and of 64KB capacity. As regard to lower level caches, we evaluate two cases, pure SRAM-based and hybrid caches with various memory technologies, including SRAM, MRAM, RRAM, and eDRAM. By evaluating both cases, we would like to find out optimal cache design that leads to the peak performance, i.e., the number of cache levels that is reqired, memory technology used for each level, and capacity of each level.

The benchmarks are selected from PARSEC benchmark suite [1] with multithreaded programs, which focus on emerging workloads that are designed to be representative of next-generation shared-memory programs for CMPs. Since the performance of different memory technologies

Table 2: Characteristics of selected benchmarks.

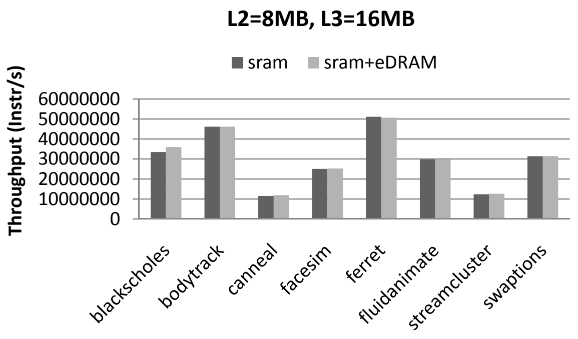
Benchmarks	**RPKI**	**WPKI**	**Benchmarks**	**RPKI**	**WPKI**
blackscholes	22.8	61.7	ferret	5.7	173.4
bodytrack	5.4	139.5	fluidanimate	2.6	70.1
canneal	5.4	29.3	streamcluster	17.3	16.9
facesim	6.0	102.4	swaptions	2.6	121.4

are closely related to read and write intensities, we selected some workloads that vary in the average numbers of L2 cache read per thousand instructions (RPKI) and write per thousand instructions (WPKI), which are listed in Table 2.

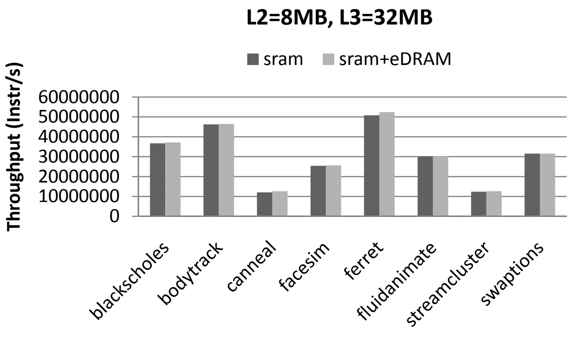
5.2 Results

We evaluate various possible configurations with shared caches by simulation, i.e., with possible numbers of levels, memory technologies, and cache capacities for each cache level. We consider SRAM and eDRAM as the possible memory technologies to implement L2 and L3 caches. Both MRAM and RRAM have higher write latency than SRAM. Furthermore, the endurance of RRAM is too low to be used as lower level caches. Consequently, these two memory technologies are only considered to be used as the last level cache.

In the first set of experiments, we evaluate the system performance with two-level shared caches. Figure 7 shows the system throughput with all the benchmarks. It is illustrated that implementing hybrid cache with eDRAM as the L3 cache helps with performance among most of the benchmarks. Such performance improvement is more than 10% in Figure 7(a). It is indicated by our memory technology exploration that eDRAM shows more latency benefits with larger capacities. As a result, a larger eDRAM-based L3 cache leads to more performance improvement to the pure SRAM implementation, as illustrated by Figure 7(b). However, hybrid cache does not always improve the performance, as shown in Figure 8. With the benchmark *canneal*, hybrid cache configurations outperform the pure SRAM implementation among overall range of various capacities.

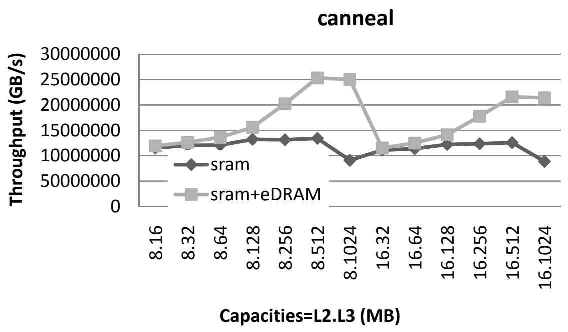


(a)

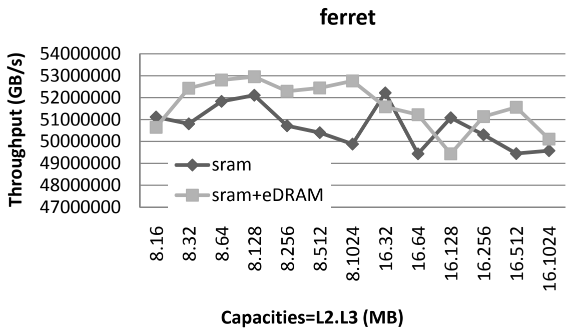


(b)

Figure 7: Performance comparison with two-level caches among various benchmarks. (a) The L3 cache capacity is 16MB. (b) The L3 cache capacity is 32MB.



(a)

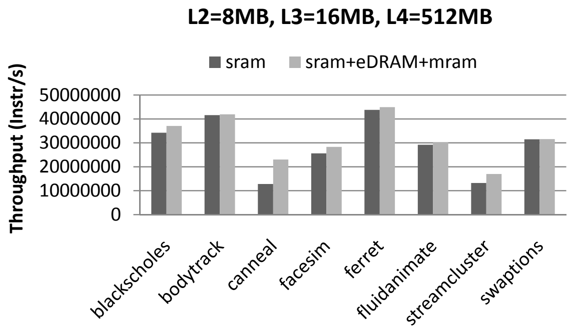


(b)

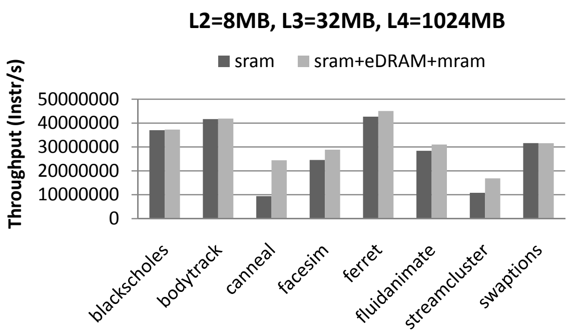
Figure 8: Performance comparison with two-level caches among various cache capacity. (a) The canneal benchmark. (b) The ferret benchmark.

With the benchmark *ferret*, however, pure SRAM implementation results in higher performance than hybrid cache implementation with the same capacity configuration.

In the second set of experiments, we evaluate the system performance with three-level shared caches. When evaluating hybrid cache configurations, we consider implementing the last level cache by MRAM and RRAM memory technologies, since the data transaction intensity is relatively low in the last level cache. Figure 9 shows the results. Figure 9(a) illustrated that the performance more than 15% higher on average with a last level cache implemented by MRAM than pure SRAM implementations. More performance improvement is obtained by increasing the last level cache capacity, as shown in Figure 9(b). Figure 10 compares performance with pure SRAM and hybrid cache implementations with different capacities. In this case, hybrid cache implementation leads to

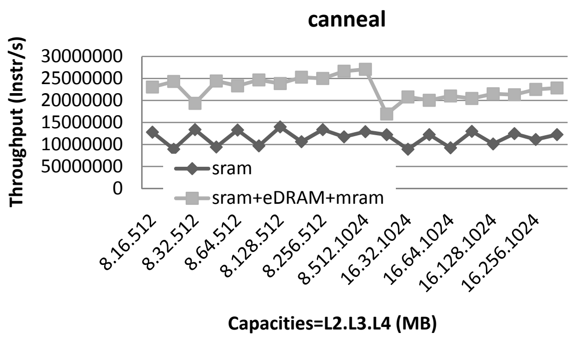


(a)

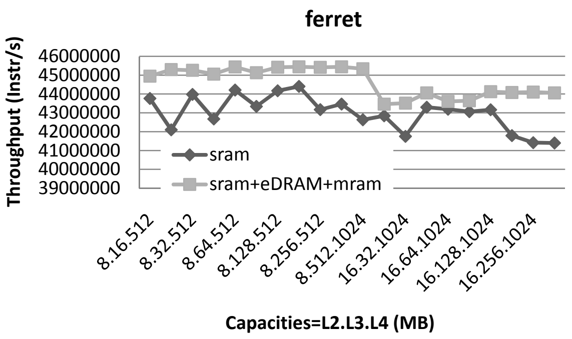


(b)

Figure 9: Performance comparison with three-level caches among various benchmarks. (a) The L3 cache capacity is 16MB, and the L4 cache capacity is 512MB. (b) The L3 cache capacity is 32MB, and the L4 cache capacity is 1GB.



(a)

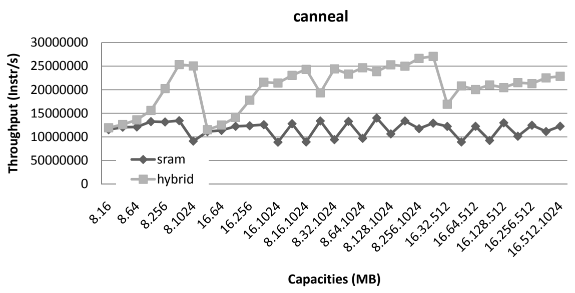


(b)

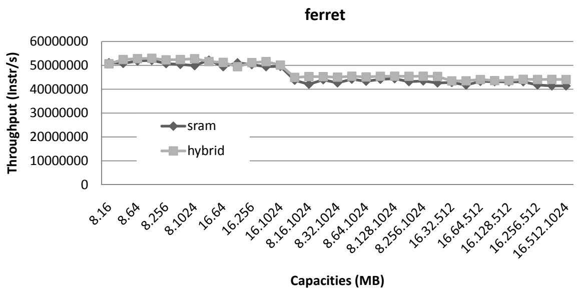
Figure 10: Performance comparison with three-level caches among various cache capacity. (a) The canneal benchmark. (b) The ferret benchmark..

higher performance in both benchmarks. The results of the two sets of experiments are illustrated together in Figure 11. An interesting observation is that hybrid cache implementation shows high performance improvement with one of the applications *canneal*, whereas as little improvement with the other *ferret*. The reason is that write latency of both MRAM and RRAM is higher than SRAM when capacity is less than 1GB. Performance of applications with large number of writes, such as *ferret* is therefore not benefit from hybrid cache design.

Finally, we evaluate all the possible cache hierarchy configurations with exhaustive simulations. The optimal cache configurations of each benchmark is shown in Table 3. Since the characteristics vary among the benchmarks, the optimal cache configurations are different among each of benchmarks. Applications with high write intensities, such as *bodytrack*, *ferret*, and *swaptions* tend to



(a)



(b)

Figure 11: Performance comparison with both two-level and three-level caches among various cache capacity. (a) The canneal benchmark. (b) The ferret benchmark.

favor SRAM-based caches. Other applications benefit more from hybrid cache implementation in terms of performance.

Table 3: Optimal cache configurations for each benchmark.

Benchmarks	L2 Configuration	L3 Configuration	L4 Configuration
blackscholes	SRAM, 8MB	eDRAM, 32MB	RRAM, 1GB
bodytrack	SRAM, 8MB	eDRAM, 128MB	-
canneal	SRAM, 8MB	eDRAM, 512MB	MRAM, 1GB
facesim	SRAM, 8MB	eDRAM, 32MB	MRAM, 1GB
ferret	SRAM, 8MB	eDRAM, 32MB	-
fluidanimate	SRAM, 8MB	eDRAM, 1GB	-
streamcluster	SRAM, 16MB	eDRAM, 128MB	MRAM, 512MB
swaptions	SRAM, 8MB	SRAM, 32MB	SRAM, 1GB

6 Conclusion and Future Work

In this project, we explore performance and energy characteristics of various emerging memory technolgies. Based on our exploration, we evaluate the shared cache hierarchy design of CMPs optimized for performance by filling the off-chip memory bandwidth gap. According to our evaluation, SRAM-based caches leads to reasonable performance with write-intensive applications. Hybrid cache design help with performance with other types of benchmarks.

Furthermore, a variety of continuing studies is remained to be explored related to bandwidth-aware memory hierarchy design. First of all, we only examine cache hierarchy design currently. It is necessary to extend our exploration to the overall memory hierarchy design. On the other hand, we do not consider energy efficiency in this project. We would like to put some power constraints to the memory hierarchy, and examine the energy efficiency of various memory hierarchy configurations.

References

[1] C. Bienia, S. Kumar, J. P. Singh, and K. Li. The PARSEC benchmark suite: characterization and architectural implications. In *Proceedings of the International Conference on Parallel Architectures and Compilation Techniques*, pages 239–249, 2008.

[2] L. Chua. Memristor: The missing circuit element. *IEEE Transactions on Circuit Theory*, 18(5):507–519, 1971.

[3] X. Dong, N. P. Jouppi, and Y. Xie. PCRAMsim: System-level performance, energy, and area modeling for phase-change RAM. In *Proceedings of the International Conference on Computer-Aided Design*, pages 269–275, 2009.

[4] X. Dong, X. Wu, G. Sun, Y. Xie, H. Li, et al. Circuit and Microarchitecture Evaluation of 3D Stacking Magnetic RAM (MRAM) as a Universal Memory Replacement. In *Proceedings of the Design Automation Conference*, pages 554–559, 2008.

[5] B. M. R. et al. Scaling the bandwidth wall: challenges in and avenues for cmp scaling. In *Proceedings of the International Symposium on Computer Architecture*, 2009.

[6] HP Labs. CACTI, http://www.hpl.hp.com/research/cacti/. 2010.

[7] P. S. Magnusson, M. Christensson, J. Eskilson, D. Forsgren, G. Hallberg, J. Hogberg, F. Larsson, A. Moestedt, and B. Werner. Simics: a full system simulation platform. *IEEE Transactions on Computer*, 35(2):50–58, 2002.

[8] P. Mangalagiri, K. Sarpatwari, A. Yanamandra, V. Narayanan, Y. Xie, et al. A low-power phase change memory based hybrid cache architecture. In *Proceedings of the Great Lakes Symposium on VLSI*, pages 395–3–98, 2008.

[9] V. Mohan, S. Gurumurthi, and M. R. Stan. FlashPower: A detailed power model for NAND flash memory. In *DATE*, pages 502–507, 2010.

[10] NASA. NPB, http://www.nas.nasa.gov/resources/software/npb.html.

[11] SPEC. SPEC CPU2006, http://www.spec.org/cpu2006/.

[12] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams. The missing memristor found. *Nature*, 453:80–83, 2008.

[13] G. Sun, X. Dong, Y. Xie, J. Li, and Y. Chen. A novel architecture of the 3D stacked MRAM L2 cache for CMPs. In *Proceedings of the International Conference on High-Performance Computer Architecture*, pages 239–249, 2009.