Bandwidth-Aware Reconfigurable Memory Design with Hybrid Memory Technologies

***Abstract*—In future chip-multiprocessor (CMP) design, memory bandwidth is a potential bottleneck to system performance. Emerging non-volatile memory technologies, such as spin-torque-transfer memory (STT-RAM), resistive memory (RRAM), and embedded DRAM (eDRAM), are promising solutions as on-chip cache and memory for CMPs. In this paper, we propose a bandwidth-aware reconfigurable memory hierarchy design with hybrid memory technologies. We demonstrate a method to dynamically reconfigure the number of shared memory levels and capacity of each level based on statistical prediction. With a set of both multithreaded and multiprogrammed applications, we evaluate system performance obtained with our method. The experimental results show that the proposed reconfigurable hybrid memory design improves the overall system throughput by % compared to pure SRAM memory design. In addition, our design leads to % throughput increase and % dynamic power reduction compared to hybrid but fixed memory hierarchy design.**

I. Introduction

One potential bottleneck for chip-multiprocessor (CMP) performance scaling is the widening gap between the bandwidth demand created by processor cores and the limited access bandwidth provided by off-chip main memory [15][16][6][20]. Such limitation greatly affects the parallelism of memory demanding applications, i.e., applications with a large working set, by consuming additional cycles on off-chip memory access. In addition, even moderate memory demand applications will be affected as the number of cores scales up [17]. The continuing shrink of transistor density not only leads to increasingly powerful processors with a large number of cores, but also puts much bandwidth pressure to off-chip main memory. However, the bandwidth to the off-chip main memory does not improve much compared to the processor core scaling. Potentially, these issues make the applications running on the computing systems be bandwidth-bound at main memory.

While a number techniques can be found in today's systems and research work to alleviate such bandwidth bottleneck, few can improve the system throughput in a power and cost efficient manner. High performance computing machines such as NVIDIA's Tesla [**?**] rely on very high main memory bandwidths (provided by GDDR memories) to feed the requirement from the processor. However, GDDR memories run at higher clock rates and are more power hungry than conventional DRAM modules. It is undesirable for either general purpose or high performance computing systems to improve their computing performance by sacrifacing power efficiency. With the emerging 3D chip technology, caches can be stacked on top of processor cores to provide high memory bandwidth[12][21][22]. However, ...

Caching has long been employed as the most effective approach to reduce memory access latency. Adding an extra level of cache, however, can also help alleviate the increasing bandwidth pressure of off-chip memory. In [dac.1] the authors discuss extensively the problem of limited pin bandwidth to multiprocessor systems. They have focused on program performance in multiprocessor systems and make detailed decomposition of program execution times. They conclude that even more complex on-chip cache structures would prove to be costeffective, with the limited pin bandwidth severely restricting performance increases. In [dac.2] the authors have carefully studied the requirements for on-chip cache structures along with all sorts of optimization techniques that come along with scaling of processor core numbers. Their study shows that cache size need to grow much faster than processor core numbers to compensate for the limited off-chip bandwidth. Because of that, the near future processors need to allocate a huge percentage of chip area for caches, which means much less core counts than expected. The study also shows that effective bandwidth optimization techniques can help reduce cache size requirement and thus help scaling processor cores. However, the effectiveness of caching in the mid/large scale multiprocessor systems could be highly suboptimal due to significant contention across the parallel tasks. The more bandwidth requirement by a workload can be satisfied by the last level cache (LLC), the less bandwidth will be required from the off-chip main memory. Our memory technology pre-exploration shows that the bandwidth provided by a memory decreases with the increase of its capacity. Based on these observations, adding more levels of cache can potentially fill in the bandwidth gap.

On-chip caches with fast random access, high storage density, and non-volatility become 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). These emerging memory technologies are believed to be promising solution as on-chip caches [1]. In addition to the benefit of non-volatility, we demonstrate that these memory technologies provide higher bandwidth than SRAM at a high capacity. Consequently, it is beneficial to use NVM as lower level caches when large capacity.

This paper presents a bandwidth-aware memory hierarchy design to enhance system performance of CMPs. Our goal is to enhance the memory system from the bandwidth point of view, by leveraging these emerging memory technologies in devising a hybrid cache hierarchy. The contributions we present in this paper include:

- Bandwidth-aware hybrid on-chip memory hierarchy design.

- Hardware support that enables prediction of cache performance on the different sized caches.
- OS scheduler support to make use of the prediction capability and appropriately schedule applications on to core with suitable cache capacity.

II. Related Work

The bandwidth problem of multicore processors has drawn much attention recently. Yu *et al.* proposed a LLC partitioning algorithm to minimize bandwidth requirement to off-chip main memory [**?**]. While most cache partitioning techniques focus on cache miss rates, our work takes a different approach in which tasks memory bandwidth requirements are taken into account when identifying a cache partitioning for multi-programmed and/or multithreaded workloads. Cache resources are allocated with the objective that the overall system bandwidth requirement is minimized for the target workload. The key insight is that cache miss-rate information may severely misrepresent the actual bandwidth demand of the task, which ultimately determines the overall system performance and power consumption.

A large body of recent research focuses on exploring new memory technologies to trade off between latency, bandwidth, and cost. Wu et al. explored various memory technologies - SRAM, eDRAM [23], magnetic RAM (MRAM) [12] and PCM to best construct L3 caches in terms of performance and power [24]. The eDRAM and MRAM technologies are evaluated as last-level caches by stacking each memory on top of the processor die. Both studies mainly focus on reducing the latency gap between L2 cache and external memory and do not examine the bandwidth. ¡nvm cache research: most for latency, not bandwidth-aware opt.¿

Application behavior prediction is a critical component of reconfigurable architectures. Zhou *et al.* monitored memory access patterns and estimated memory behavior of workloads for energy efficient memory allocation [predictor.26]. Duesterwald et al. [10] describe different statistical and table based predictors for within- and across-metric predictions of performance monitoring information. They show that the table-based predictor generally outperforms the other predictors they tested. Sarikaya et al. [19] describe an optimal prediction technique based on a predictive least squares minimization. Sarikaya *et al.* showed the benefit of statistical metric modeling for tracking varying pattern history lengths and modeling long term patterns [**?**]. However, ...

III. Background

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.

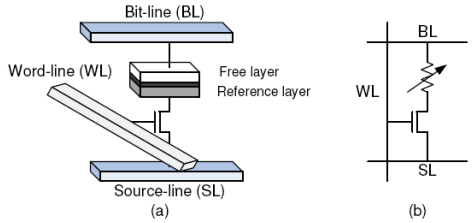


Fig. 1. Demonstration of a MRAM cell. (a) Structural view. (b) Schematic view.

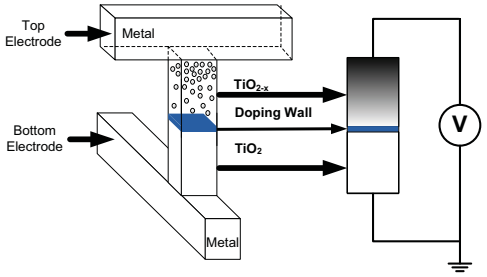


Fig. 2. The conceptual view of the structure of memristor cells.

RRAM

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 [3]. Fig. 2 shows a conceptual view of the memristor structure [3]. The top electrode and bottom electrode are two metal nanowires on platinum, and the thin titanium dioxide film is sandwiched by the electrodes.

IV. Memory Technology 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 [4] 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 [5], PCRAM [6], [7], and NAND flash [8]. 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.

First of all, we estimate the read and write bandwidths that can be provided by different memory technologies. Fig-

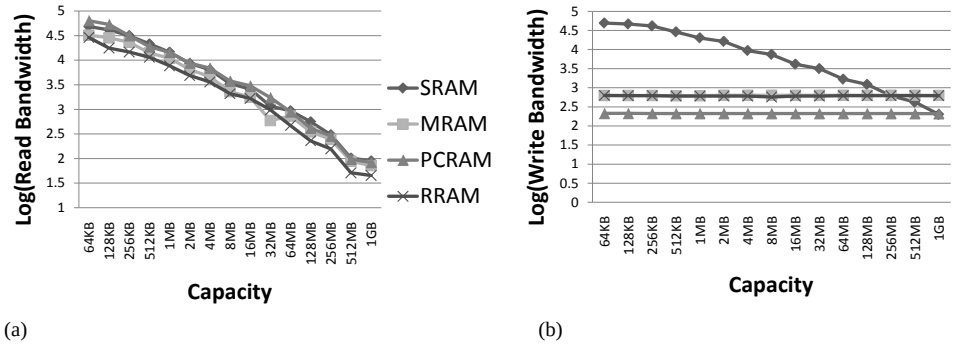


Fig. 3. Read and write bandwidths provided by different memory technologies. (a) Read bandwidth provided by different memory technologes. (b) Write bandwidth provided by different memory technologies.

ure 3 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 3(a). On the other hand, a straight forward observation from figure 3(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 3(b) is that the curves cross to each other at different locations.

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:

1) H-tree input delay
2) Decoder + word-line delay
3) Bit-line delay + Sense Amplifier delay
4) Comparator Delay (for tag part only)
5) H-tree output delay

H-tree latencies 1) and 5) are mainly determined by the RC delay of global wires, which is positive proportional to the area of memory macro. Sensing delay 4) is related to the read noise margin of memory cell that is affected by off/on resistance ratio. Figure 4 (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 4 (b) indicates that NVM is suitable for memory with large capacity.

Figure 5 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.

V. Design Method

VI. Experiments

Based on the parameters of different cache configurations collected from our modified version of CACTI [4], 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.

A. Experimental Setup

We use Simics [9] 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 I 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

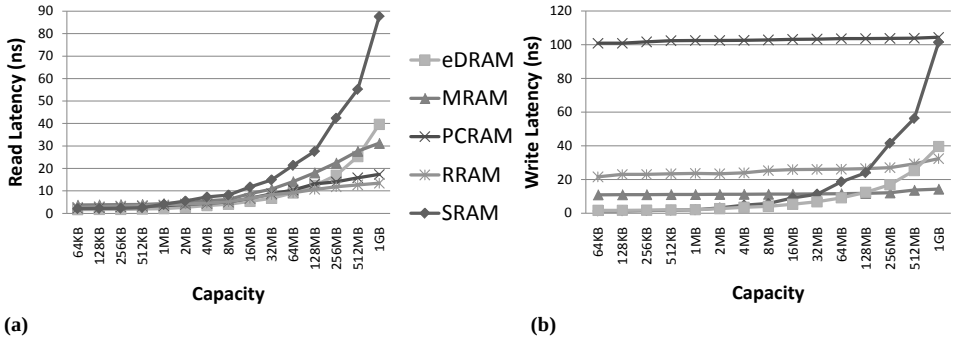


Fig. 4. Latency of different memory technologies. (a) Read Latency. (b) Write Latency.

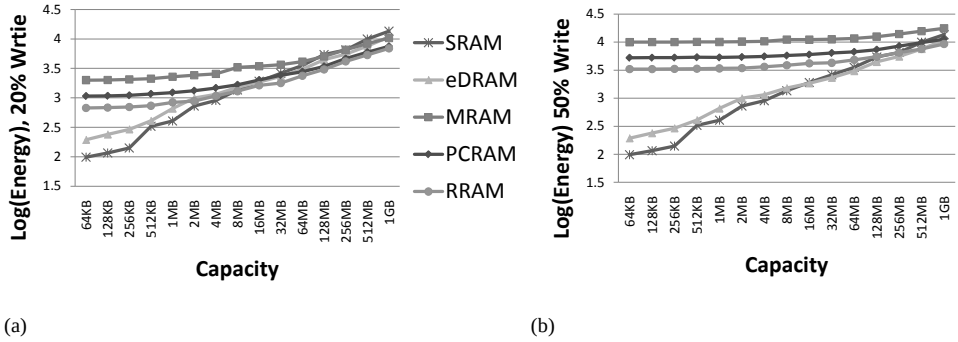


Fig. 5. Dynamic energy consumption with the provided bandwidths of different memory technologies. (a) Dynamic Energy with 20% write. (b) Dynamic Energy with 50% write.

TABLE I
BASELINE CMP CONFIGURATION.

Core	
No. of cores	8
Frequency	1GHz
Core architecture	in-order, 14-stage pipeline
Memory	
Private caches	L1-I/D caches: 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

TABLE II
CHARACTERISTICS OF SELECTED BENCHMARKS.

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

that is reqired, memory technology used for each level, and capacity of each level.

The benchmarks are selected from PARSEC benchmark suite [10] 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 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 II.

B. 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 6 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 6(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 6(b). However, hybrid cache does not always improve the performance, as shown in Figure 7. With the benchmark *canneal*, hybrid cache configurations outperform the pure SRAM implementation among overall range of various capacities. 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 8 shows the results. Figure 8(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 8(b). Figure 9 compares performance with pure SRAM and hybrid cache implementations with different capacities. In this case, hybrid cache implementation leads to higher performance in both benchmarks. The results of the two sets of experiments are illustrated together in Figure 10. 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 III. 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 favor SRAM-based caches. Other applications benefit more from hybrid cache implementation in terms of performance.

VII. 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] 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*, 2009, pp. 239–249.

[2] L. Chua, "Memristor: The missing circuit element," *IEEE Transactions on Circuit Theory*, vol. 18, no. 5, pp. 507–519, 1971.

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

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

[5] 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*, 2008, pp. 554–559.

[6] 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*, 2008, pp. 395–3–98.

[7] 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*, 2009, pp. 269–275.

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

[9] 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*, vol. 35, no. 2, pp. 50–58, 2002.

[10] 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*, 2008, pp. 239–249.

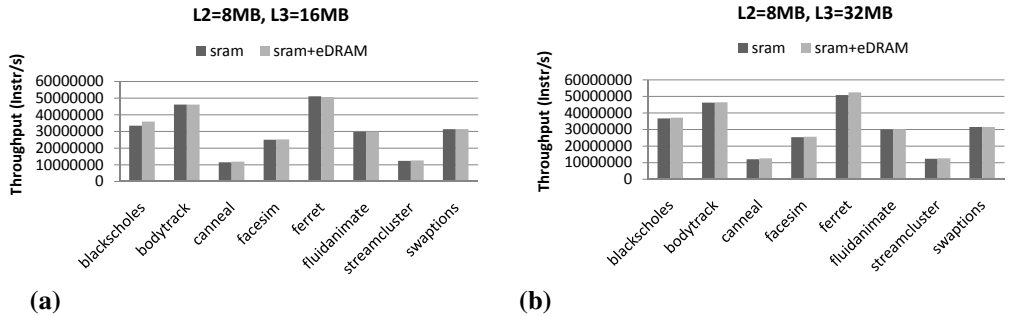


Fig. 6. Performance comparison with two-level caches among various benchmarks. (a) The L3 cache capacity is 16MB. (b) The L3 cache capacity is 32MB.

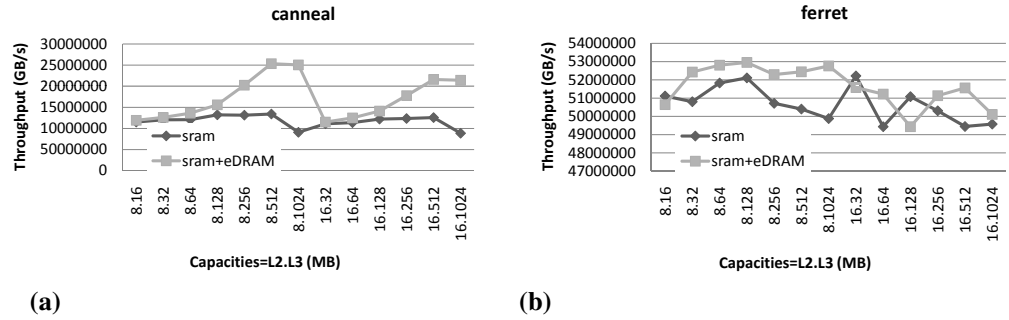


Fig. 7. Performance comparison with two-level caches among various cache capacity. (a) The canneal benchmark. (b) The ferret benchmark.

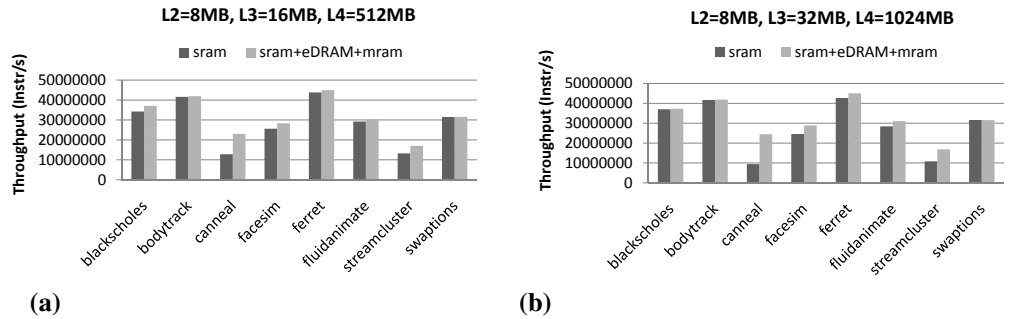


Fig. 8. Performance comparison with three-level caches among various benchmarks. (a) The L3 cache capacity is 16MB, and the L4 cache capacity is 512MB. (b) The L3 cache capacity is 32MB, and the L4 cache capacity is 1GB.

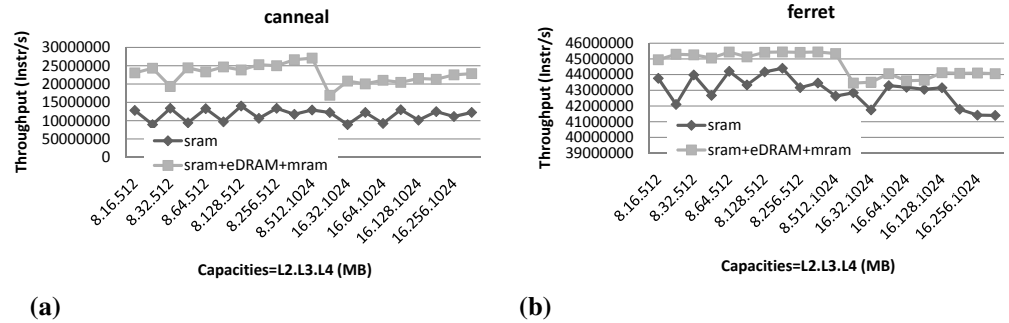
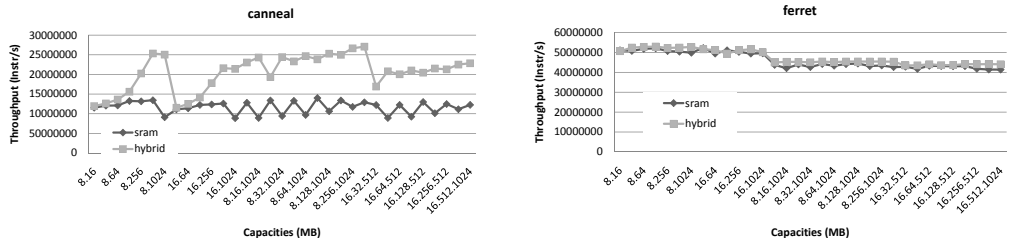


Fig. 9. Performance comparison with three-level caches among various cache capacity. (a) The canneal benchmark. (b) The ferret benchmark..



(a) **(b)**

Fig. 10. Performance comparison with both two-level and three-level caches among various cache capacity. (a) The canneal benchmark. (b) The ferret benchmark.

TABLE III
OPTIMAL CACHE CONFIGURATIONS FOR EACH BENCHMARK.

Benchmarks	**L2**	**L3**	**L4**
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