Architectural Design for Next Generation Heterogeneous Memory Systems

Alan Bivens, Parijat Dube, Michele Franceschini, John Karidis, Luis Lastras, Mickey Tsao
IBM T.J. Watson Research Center
Yorktown Heights, NY 10598
{jbivens, pdube, franceschini, karidis, lastrasl, mtsao}@us.ibm.com

***Abstract*—New enterprise workloads requiring fast, reliable access to increasing amounts of data have pushed today's memory systems to power and capacity limits while creating bottlenecks as they ensure transactions are persistently tracked for reliability. New storage class memory technologies (such as phase change memory) have the potential to offer high capacity within latency and bandwidth ranges acceptable for a computer memory system and persistence which may help ease the system-level burden of balancing performance and reliability. This paper describes architectural options for addressing the challenges of future, heterogeneous memory systems as well as the attributes required of the next generation memory devices.**

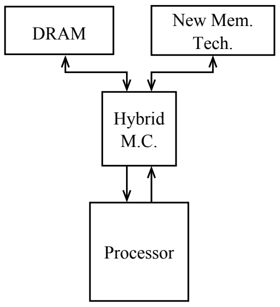

Fig. 1. Schematic diagram of a hybrid memory system

I. Introduction

Recent advances in new memory technologies are drawing the attention of computer manufacturers and challenging, for the first time in more than 30 years, the role of DRAM in the main memory system. Phase-change memory (PCM), Flash memory, and magnetic RAM (MRAM), to name a few, are memory technologies that may play an important role in future main memory systems. These memory technologies present several interesting characteristics that differentiates them from DRAM:

- A cost model that is following the trends of the storage market (e.g. Flash) which has the potential to provide very large capacity in the main memory system.
- Non-volatility allowing ultra low standby power and enabling efficient persistence of data (can allow some applications to avoid a lengthy IO trip to a slow storage device just to ensure data is persistent).
- A tradeoff of density and performance (e.g. PCM and Flash) that could lead to specific fitting of the memory subsystem to the computer system and workload. This also leads to a natural tiering of the memory system.

We refer to these memory technologies collectively as Storage Class Memory (SCM).

On the flip side, either due to intrinsic limitations in the technology or to the relatively early stage of development, certain performance parameters of SCM may not meet the requirements for their direct use in the memory space. These include large read latency, low write bandwidth, non-deterministic write times, limited endurance, large active power, short and long term drifts in cell resistance, heterogeneous cell capacity etc. New memory architectures and solutions are needed to build memory systems using SCM which intelligently leverage the appealing attributes of SCM (increased capacity, persistence, etc.) in the memory space while minimizing the impact of their performance constraining attributes.

In this paper, we will discuss a versatile memory architecture known as *hybrid memory* that mitigates the impact of some of the drawbacks associated with the use of these new memory technologies. First, we will introduce the basic architecture structure. Then, we will focus on a number of low level characteristics of the memory devices, such as read latency, read and write throughput, endurance and reliability, providing working ranges for these parameters, assuming a sample system.

II. SCM Based Memory Architecture

A. Overall Architecture

A hybrid memory architecture (Fig. 1) is characterized by a memory sub-system containing at least two memory regions having significantly different performance attributes like fast, expensive DRAM coupled with a slower, cheaper SCM. The heterogeneity of memory in such architectures can either be made transparent to software or can be exposed to the software. In the former, transparent case the physical memory of the system can be entirely SCM based with DRAM acting as hardware cache for SCM. The idea is that a relatively small sized DRAM acting as a last level cache (L4/L5) for a much larger (though slower) memory can have a sufficiently high hit rate (90-95%) and mitigate the slowness and some of the endurance challenges of the slower memory. This is based on the standard principle for caching exploiting the temporal correlation in memory references. This architecture is currently being studied [4], [3]. When the hybrid memory architecture is exposed to software, the software can determine where to place data (based on application needs). In this architecture, a portion of DRAM can still act as a cache for the SCM region of memory to mitigate its slowness; however, as less DRAM cache is used, the memory reference penalty for using SCM rises. The overhead of software memory management also needs to be accounted for when evaluating its benefits. For the purpose of this paper, we will concentrate on hybrid architectures that are transparent to software.

B. Example System

For the numerical evaluations in this paper we consider a sample Hybrid Memory system roughly sized on a modern enterprise system (considering only one memory channel). The memory channel bandwidth is 16GB/s for reads, or *upstream*, and 8GB/s for writes, or *downstream*. The cacheline size is 128B. We assume that the speed of the processor and the total size of the memory are large enough for the performance to be bound only by the organization, the latency, and the bandwidth of the memory itself. The assumed upstream probe rate is 125 Mlines/s. The processor is connected to the hybrid memory controller (HMC) which in turn controls a pool of DRAM chips totaling 16GB. HMC also controls a pool of 64 SCM chips also divided into ranks of 16 chips. We assume 16Gb SCM chips, for a total of 128GB of SCM. The DRAM in this system acts as a hardware cache for SCM memory. The heterogeneity of the memory is not exposed to the software.

III. Performance of SCM Based Memory System

The hybrid memory architecture in our study has a very large last level cache. Understanding the performance of this architecture will require innovative tools and techniques to cover cache management dimensions which are unachievable with traditional trace based simulations due to the limited length and breadth of available traces. Full system simulation of such dimension consume prohibitively large time and is undesirable during early design phase. We developed a multi-processor system [1] which runs the applications of interest (for any duration) with a coherently attached FPGA that can emulate any memory architecture (also for any period of time). Different memory subsystem configuration and management attributes like line size, cache size, associativity, replacement algorithms are being studied with this prototype.

Figure 2 shows the miss ratio for direct mapped last level cache vs. cache size for B-SE, a search engine index benchmark with a 2GB index set. We observe a power law behavior between miss ratio and cache size for fixed line size, with the power law exponent being workload dependent. This reaffirms the power law type dependency reported in earlier works for much smaller sized caches (Kilobytes) and for some other set of workloads. Further, the gain with increasing cache size is workload dependent and simple square root type decay with doubling cache size will result in incorrectly over/under estimating this gain for specific workloads. We also observed that for fixed cache size, increasing the line size decreases miss ratio initially but later the miss ratio increases with line size (see Fig. 3 for B-SE benchmark). This sensitivity to line size is dependent on both the cache size and the workload. We next

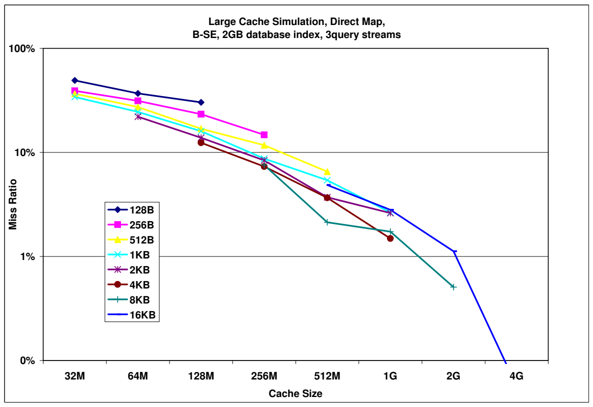

Fig. 2. Miss ratio Vs cache size for different line sizes

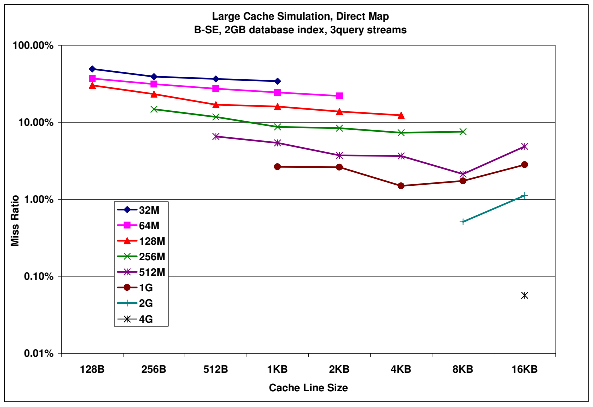

Fig. 3. Miss ratio Vs cache line size with different cache sizes

use these numbers and obtain SCM chip level performance characteristics for our hybrid memory system.

For performance studies, we consider our system in a virtualized environment where multiple instances (each instance operates on a Virtual Machine (VM)) of the same application are concurrently running. Let N be the number of active VMs. We assume fair sharing of the cache by different VMs, thus each VM virtually operates with a cache of about $1/N$th of the total DRAM size. We take $N = 32$. which gives about 512MB of DRAM cache per VM for a 16GB total DRAM size on the system. We consider direct map DRAM cache as having associativity in such large caches may be prohibitively expensive due to the size and power of the required directory. The latency of the directory for the large cache is very important, as that latency will be added to all memory operations. Therefore, the directory would most likely be implemented in SRAM or EDRAM. Naively storing the directory in the DRAM cache would involve at least two DRAM reads per memory access.

A. Chip Bandwidth Requirement

Reading and writing large block sizes to memory requires sufficient bandwidth availability between the DRAM cache and the memory. A miss in DRAM cache triggers a block to be read from memory and sometimes a subsequent write to memory. A line miss in cache results in a memory write only if the corresponding line being evicted is dirty. Further on the line being dirty, either the complete block (a block has one or more cache lines) is evicted from DRAM and written to memory (block write) or only the dirty line is replaced in the memory (selective write). The write bandwidth required in the block-write mode will be the read bandwidth times the probability that at least one line in the block is dirty. The amount of write bandwidth required in the the selective-write mode will simply be the read bandwidth times the probability of a line being dirty. Since the memory block is retrieved from 16 chips in our example system, the amount of data retrieved per chip is block_size(B)/16. Thus if m is the miss ratio and λ is the arrival rate of requests to DRAM cache then the rate of DRAM misses is $m\lambda$. Since each miss involves $B/16$ worth of data to be read from a chip, the read bandwidth required per chip (R_r) is given by:

$$R_r = m\lambda\left(\frac{B}{16}\right). \tag{1}$$

Let p_d be the probability that a line is dirty. Then for a cache line size L, the write bandwidth W_r required is given by:

$$W_r = \begin{cases} m\lambda\left(\frac{B}{16}\right)\left(1-(1-p_d)^{\frac{B}{L}}\right) & \text{block-write} \\ m\lambda\left(\frac{B}{16}\right)p_d & \text{selective-write} \end{cases} \tag{2}$$

We next calculate the read and write bandwidth required as a function of the chip data size. Since the DRAM cache per VM is 512MB, the miss ratio seen by each VM in the DRAM cache can be approximated by the miss ratio in a non-virtualized system with a 512MB DRAM cache running single instance of the application. Thus we use the the miss ratio in a 512MB cache from our FPGA measurements as the miss ratio seen by a VM in the DRAM. For write bandwidth calculations we took $p_d = 0.5$ which we also observed in our experiments. Fig. 4 shows the read and write bandwidth required for two workloads, B-SE and B-SJ. B-SE is a web search application against a 2GB database index and B-SJ is the SpecJbb, ran with eight warehouses. For these workloads, the bandwidth required for a 4KB block size (corresponding to 256B chip data) is about 1GB/s. Fig. 5 shows the zoom of Fig. 4 for smaller block sizes. The bandwidth required mostly increases with increasing block size, thus the benefits due to prefetching comes at the cost of increased read/write bandwidth per chip. One exception is B-SE for 16B chip data size, where the read bandwidth is lower than its value for 8B case. This may be attributed to the gain in miss ratio as the corresponding block size in DRAM cache increases from 128B to 256B. However, the write bandwidth required is higher(lower) than the 8B case for block-write(selective-write) mode. Observe that the write bandwidth required is initially lower than the read bandwidth since not all evictions result in a block write. With selective-write the write bandwidth will always be one-half of the read bandwidth. For large block sizes the probability of at least one line in a block being dirty becomes close to 1 and hence write bandwidth is practically the same as the read bandwidth.

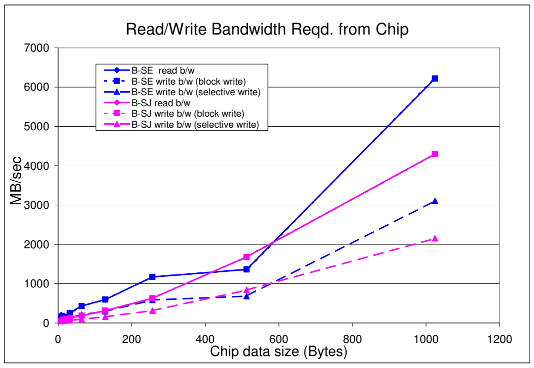

Fig. 4. Read and write bandwidth required as a function of data from a SCM chip in a system with 16GB total DRAM cache running 32 VMs.

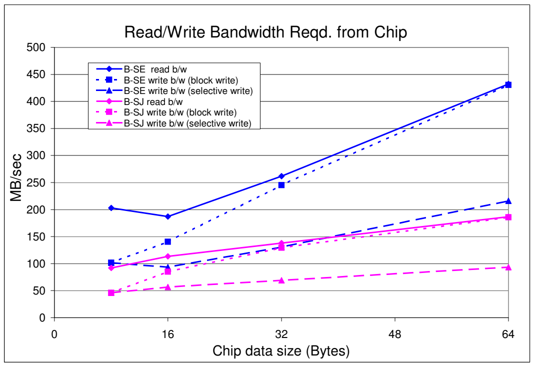

Fig. 5. Zoom of Fig. 4 for lower values of chip data size.

B. Chip Endurance Requirement

We next obtain lower bounds on endurance required from a SCM memory chip in hybrid memory. Endurance requirement is given in number of write cycles that the chip needs to sustain during the system life. Assuming a system life time, the number of times the chip is written completely during this lifetime gives the minimum endurance required for proper functioning of the system. While this assumes perfect wear leveling (i.e. even wear of the memory), it has been shown that simple hardware mechanisms can deliver near-optimal wear distribution [2]. Thus the chip endurance required is calculated as

$$\text{required\_chip\_endurance} = \frac{\text{write\_bandwidth*system\_lifetime}}{\text{chip\_size}}.$$

For endurance calculations, we assume a 3 year system lifetime with a 2GB chip memory and selective-write mode. Observe that for a given miss rate in the cache, the number of write cycles increase with the block size as the write bandwidth increases. Figure 6 shows the endurance required for the two workloads B-SE and B-SJ using the write bandwidth numbers from Figure 4. The endurance requirement for block-write and selective write mode is of the same order.

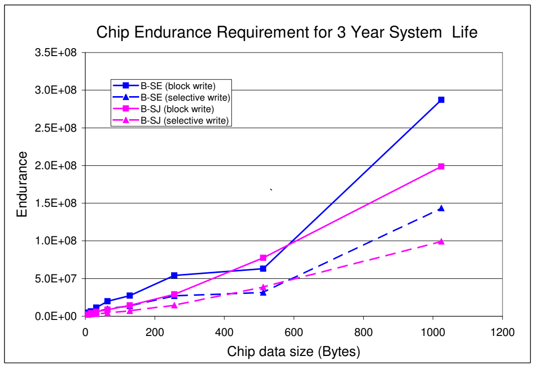

Fig. 6. Endurance required from a 2GB chip for a system life cycle of three years.

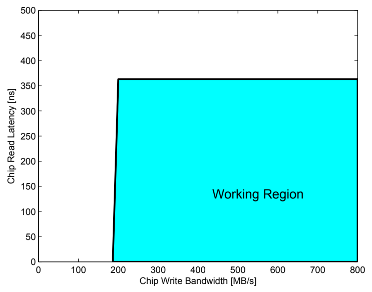

Fig. 7. Feasible working region assuming DRAM cache latency of 100ns, per chip write bandwidth requirement of 200 MB/sec and a 2% miss ratio in DRAM cache.

C. Feasible Working Region

In Figure 7, the region for which the hybrid memory system has an estimated performance loss of 5% or less is shown. The performance is estimated as

$$P = \frac{\min\{W, W_r\}}{W_r} * \frac{l_{\text{dram}}}{m l_{\text{scm}} + (1-m) l_{\text{dram}}}$$

where W is the chip write bandwidth, W_r is the write bandwidth required by the combination of workload and system structure, l_{scm} is the read latency of the storage class memory, and l_{dram} is the latency of dram (assumed to be 100ns). The shaded region represents the pairs of read latency/write throughput parameters that would deliver a negligible performance loss. As an example, it is clear that current NAND Flash memory, with 25000ns read latency cannot be used as main memory without incurring unacceptable performance loss.

IV. Data Organization, Reliability, Coding

In Figure 8, the impact of reliability of a memory technology on the actual amount of memory that needs to be deployed to provide the proper reliability is shown. The reliability of the device is defined as the raw bit error rate (BER) associated with the particular memory technology (coding inside the chip cannot provide a better performance with the same raw access granularity). We assume an access granularity of 32B per chip and 16 chips are devoted to the data. To provide a probability of unrecoverable data loss equal or better than 10^{-20} (enterprise grade reliability), we encode the data with an ECC, adding a proper number of chips to store the required redundancy. Figure 8, shows the maximum effective memory density multiplier predicted by the Shannon limit, which assumes an infinite codeword length, and predicted by the sphere-packing bound (or Hamming bound, H.B.). The redundancy required by a Reed-Solomon (R.S.) code is also shown. In addition, the same curves are shown (for the H.B. and the R.S. codes) adding 2 redundant chips which is a rough estimate of the redundancy required to cope with a sudden chip failure. Reliability starts having an impact on effective density, and ultimately on the actual memory cost whenever the raw reliability is lower than 10^{-5}, with significant penalties beyond 10^{-3}. The computational complexity of ECC should also be taken into account (this is dependent on the device reliability).

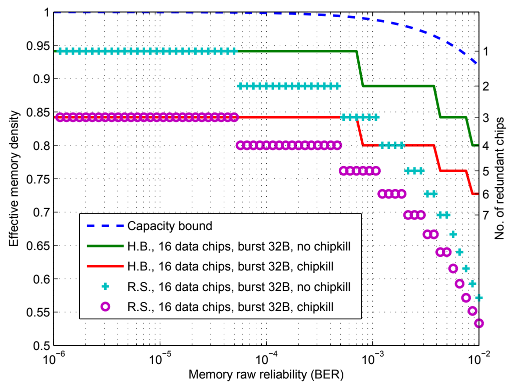

Fig. 8. Effective memory density as a function of the memory reliability. A number of chips are accessed in parallel to store and retrieve the equivalent of 16 chips worth of data.

V. Conclusion

New memory technologies such as Storage Class Memories offer tremendous benefit for the memory subsystem, but all present challenges that can be mitigated through hardware and software architecture. This paper illustrates the range of bandwidth, latency, endurance, and reliability needed for a new memory technology to fit within these architectures. This approach can be used as a guideline in the optimization of the SCM design point allowing vendors to maximize the appealing attributes of SCM (cost/density, persistence, flexibility) while maintaining suitability for main memory use.

References

[1] P. Dube, M. Tsao, D. Poff, L. Zhang, and A. Bivens. Program behavior characterization in large memory systems. In *Proceedings of ISPASS 2010: IEEE International Symposium on Performance Analysis of Systems and Software*. IEEE, 2010.

[2] M. K. Qureshi, J. Karidis, M. Franceschini, V. Srinivasan, L. Lastras, and B. Abali. Enhancing lifetime and security of PCM-based main memory with start-gap wear leveling. In *Micro-42: Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*, pages 14–23, New York, NY, USA, 2009. ACM.

[3] M. K. Qureshi, V. Srinivasan, and J. A. Rivers. Scalable high performance main memory system using phase-change memory technology. *SIGARCH Comput. Archit. News*, 37(3):24–33, 2009.

[4] W. Wang, Q. Wang, W. Wei, and D. Liu. Modeling and Evaluating Heterogeneous Memory Architectures by Trace-driven Simulation. In *2008 Workshop on Memory Access on Future Processors: a solved problem?*, May 2008.