Active Low-Power Modes for Main Memory with MemScale

Qingyuan Deng, David Meisner[†], Luiz Ramos,
Thomas F. Wenisch[†], and Ricardo Bianchini

Rutgers University
{qdeng,luramos,ricardob}@cs.rutgers.edu

[†] *University of Michigan, Ann Arbor*
{meisner,twenisch}@umich.edu

Abstract

Main memory accounts for a growing fraction of server power. MemScale introduces active low-power modes for main memory, which trade memory bandwidth for energy savings while tightly limiting the associated performance impact.

Introduction

Over the last several years, it has become clear that the massive energy consumption of data centers represents a serious burden on their operators and on the environment. Concern over this energy consumption has prompted numerous efforts to improve energy efficiency. Today, the energy consumption of the servers dominates that of the cooling and power delivery infrastructures.

Historically, within the server, the processor has dominated energy consumption. However, as processors have become more energy-efficient and more effective at managing their own power consumption, their contribution has decreased. In contrast, main memory energy consumption has been growing, fueled by the increasing memory bandwidth and capacity demands of multi-core processors, especially in the face of server virtualization. According to Google [1], main memory's contribution to overall server power ranges from 30% to 40%.

The early works on memory energy conservation exploited DRAM's idle low-power states. Because servers typically do not exhibit long periods of memory idleness, their focus was on creating idleness through access scheduling, batching, and layout transformations (e.g., [2,3,4]). Those studies generally assumed the rich, chip-level power management permitted in older technologies, such as RDRAM. More recent studies consider reducing the number of DRAM chips that are accessed at a time and even changing the micro-architecture of the DRAM chips themselves (e.g., [5,6,7]). A common theme of these latter works is to reduce the number of chips or bits actually touched as a result of a memory access, thereby reducing dynamic energy consumption.

Although the prior techniques have been successful in many cases, we argue that none of them is ideal. Creating enough idleness is hard in modern DDR* memories, since power management is available only at coarse granularity (entire ranks spanning multiple chips). Thus, deep idle low-power states can rarely be used without excessively degrading performance. Moreover, changes to the architecture of the memory DIMMs are expensive and increase latency, whereas changing DRAM chip micro-architecture may have negative implications on capacity and yield. Memory controller power has not been considered in prior work.

In contrast, we propose MemScale, a set of low-power modes, hardware mechanisms, and software policies to conserve energy while respecting the applications' performance requirements. Specifically, MemScale creates *active* low-power modes for the main memory subsystem (formed by the memory channels, the DRAM devices, and the memory controller). Our approach is based on the key observation that server workloads, though often highly sensitive to memory access latency, only rarely demand peak memory bandwidth. To exploit this observation, we propose to apply dynamic voltage and frequency scaling (DVFS) to the memory controller and dynamic frequency scaling (DFS) to the memory channels and DRAM devices. By dynamically varying voltages and frequencies, these mechanisms trade available memory bandwidth to conserve energy when memory activity is low.

Background and Motivation

DRAM Organization: Figure 1 illustrates the organization of the memory subsystem. To service memory accesses, the memory controller (MC) sends commands to the DIMMs on behalf of the CPU's last-level cache (LLC) across a memory bus. To enable greater parallelism, the width of the memory bus is split into multiple channels. These channels act independently and can access disjoint regions of the physical address space in parallel. Multiple DIMMs may be connected to the same channel. The subset of DRAM chips that participate in each access is called a rank. The number of chips in a rank depends on how many bits each chip produces/consumes at a time. Each

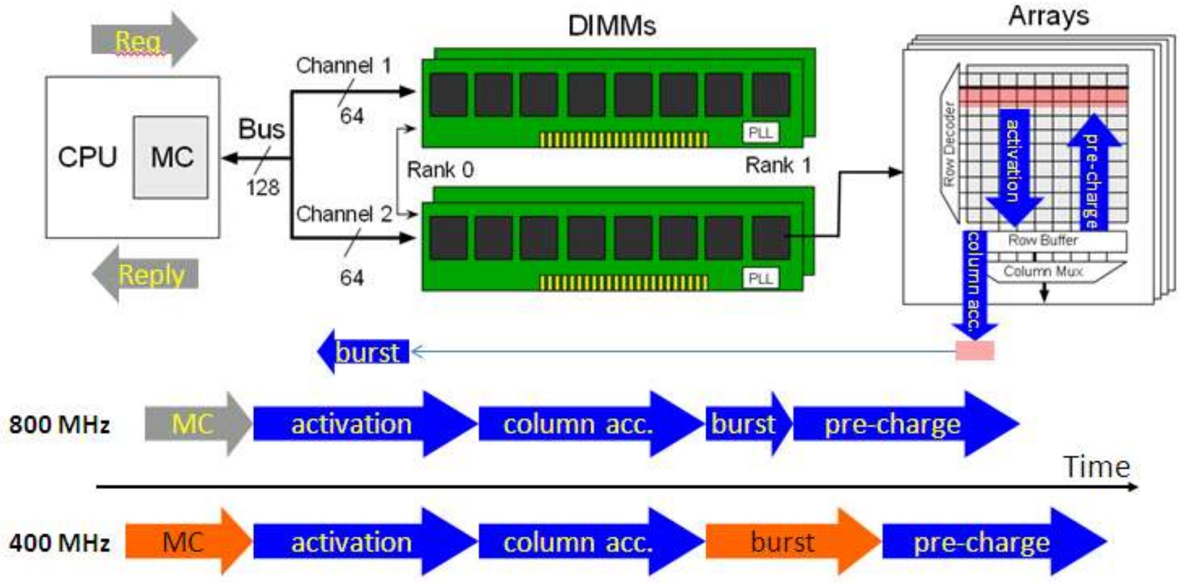

Figure 1. DRAM organization and timing.

DIMM can have up to 16 chips (or 18 chips with ECC), organized into 1-4 ranks. Each DRAM chip typically contains 8 banks. When a bank is accessed, an entire multi-KB row is transferred to a row buffer. This operation is called a "row activation". Then, any column of the row can be read/written over the channel in one burst. Because the activation is destructive, the corresponding row eventually needs to be "pre-charged", i.e., written back to the corresponding bank.

DRAM Timing: The commands sent by the MC to the DRAM chips must be properly ordered and obey certain timing restrictions. For example, a row activation first requires a pre-charge of the data in the row buffer, if a row is currently open. Otherwise, the activation can proceed without any delay. An example timing restriction is the amount of time between two consecutive column accesses to an open row. The latest DDR3 devices perform each pre-charge, activation, or column access in around 15ns, regardless of the memory frequency. Transferring a 64-byte cache line over the channel takes 4 bus cycles, since data is transferred on both clock edges in DDR technology.

MC and DRAM Power: We break down the power consumption of the memory subsystem into three categories: DRAM, register/PLL, and MC power. The DRAM power can be further divided into background, activation/pre-charge, read/write, and termination powers. The background power is independent of activity, and is due to the peripheral circuitry, transistor leakage, and refresh operations. The activation/pre-charge power is due to these operations on the memory arrays. The read/write power is due to column accesses to row buffers. The termination power is due to terminating signals of other ranks on the same channel. The three latter classes of DRAM power are often referred to as "dynamic DRAM power", but they also include a level of background power. We observe that: (1) background power contributes a significant fraction of total power consumption, regardless of workload; (2) dynamic power is only high when the memory activity is high; and (3) although the MC and register/PLL powers are often disregarded by researchers, they contribute significantly to the total power.

Impact of Voltage and Frequency Scaling: Lowering frequency affects bandwidth more than latency—for example, for DDR3 DRAM, scaling frequency from 800MHz to 400MHz decreases bandwidth by 50%, but only increases latency by around 10%. Lowering frequency makes data bursts longer and the MC slower, both by linear amounts. (The wall-clock performance of other operations is unaffected.) Because of these delays, queues at the MC may become longer, increasing memory access times further. Nevertheless, these latency increases in certain stages of the memory access process do not translate into linear increases in overall memory access time. The bottom of Figure 1 illustrates the effect of a frequency change on the memory access time.

Lowering frequency affects power consumption in many ways. First, it lowers the background and register/PLL powers linearly. Second, it lowers the MC power approximately by a cubic factor, as lowering the MC frequency is accompanied by lowering its voltage, which provides similar dynamic power advantages as CPU DVFS (Power ~ Voltage^2 × Frequency). Third,

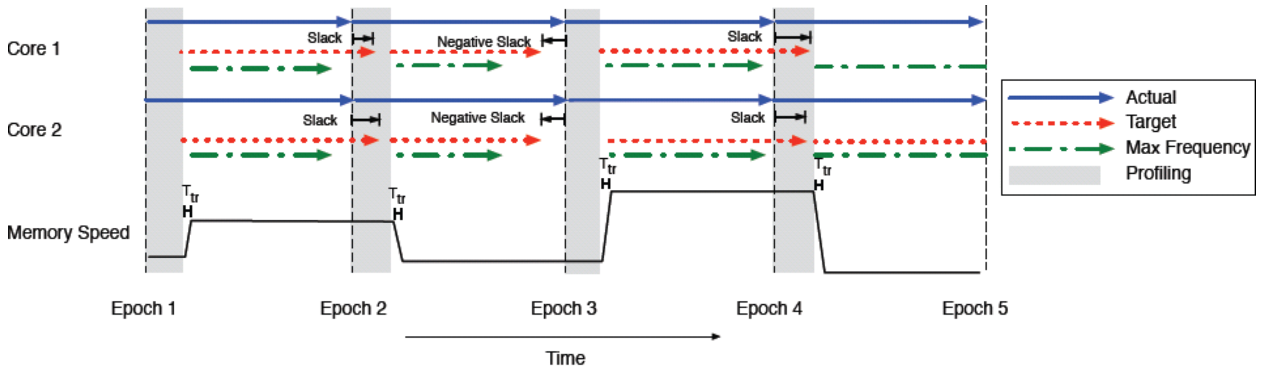

Figure 2. MemScale example.

lowering frequency increases read/write and termination energy almost linearly (power is not affected but accesses take longer). Finally, if lowering frequency degrades application performance, the energy consumed by the server's non-memory-subsystem components will increase accordingly.

In essence, MemScale trades peak channel bandwidth for lower energy when memory utilization is low, at only a modest cost in memory latency.

MemScale

The key enhancement we add to modern memory subsystems is the ability to adjust MC, bus, and DIMM frequencies during operation. Furthermore, we propose to adjust the supply voltage of the MC (independently of core/cache voltage) in proportion to frequency. The JEDEC standard already provides mechanisms for changing frequency; we leverage the pre-charge power-down state for frequency re-calibration. The majority of recalibration latency is due to DLL synchronization time, which consumes approximately 500 memory cycles. The DIMM clocks lock to the bus frequency (or a multiple thereof), while the MC frequency is fixed at double the bus frequency.

It is important to note that MemScale requires only limited hardware changes—none involving the DIMMs or DRAM chips—as most of the required mechanisms are already present in current memory systems. The key hardware additions are new performance counters.

Performance Counter Monitoring: MemScale requires counters that track (1) the number of instructions and LLC misses (used to estimate forward progress and sensitivity to memory latency), (2) the number of requests outstanding at each bank and channel (used to estimate queuing delays), (3) row buffer performance (used to estimate average row access latency), and (4) time spent in various DRAM states (used to estimate DRAM power). The counters for outstanding requests and DRAM states do not exist in current architectures and are being proposed in this work. The other counters already exist in many architectures, and are accessible through the CPU's performance-monitoring interface. MemScale reads the counters to implement its power management policy.

Power Management Policy: MemScale's power management policy is epoch-based. It selects power-bandwidth operating points using online profiling and memory power/performance models. With input from the performance counters, the models estimate the performance and energy consumption of each core during an epoch, based on the behaviors observed during a short profiling phase that starts the epoch. The policy is performance-aware in that it is driven by a user-supplied threshold on maximum allowable performance degradation.

In more detail, in each epoch, MemScale estimates the performance degradation applications would suffer at each memory frequency relative to their performance at the peak frequency. MemScale then selects the most energy-efficient operating point that meets the user-supplied degradation threshold. To correct for performance mis-estimation from epoch to epoch, MemScale tracks positive or negative performance slack relative to the threshold for each application in each epoch, and uses this slack to adjust its performance target for the next epoch. Thus, across several epochs, MemScale maintains the desired performance for each application while minimizing overall system energy consumption.

Figure 2 illustrates the behavior of MemScale, assuming two cores. Each epoch begins with a profiling phase, shown in gray. Using the profiling output, the system estimates the performance at the highest memory frequency ("Max Frequency"), and then sets a target performance ("Target") using the performance model. Based on the target, a frequency is selected and the system transitions to it. In Epoch 1, the example shows that the actual execution ("Actual") is faster than the target. Hence, the additional slack is carried forward into Epoch 2, slightly widening the gap between Max Frequency and Target, allowing the memory frequency to be lowered. However, at the end of Epoch 2, performance falls short of the target, and the negative slack must be made up in Epoch 3 (or later epochs, if necessary) by raising memory frequency. By adjusting the slack from epoch to epoch, MemScale tries to ensure that the allowable performance degradation is not exceeded.

Performance Model: Our management policy relies on a performance model to predict the relationship between an application's CPU cycles per instruction (CPI) and the memory frequency. The purpose of our model is to determine the runtime and power/energy implications of changing memory performance. Given this model, the operating system can set the frequency to both maximize energy-efficiency and stay within the pre-defined limit for CPI loss.

MemScale estimates the runtime of a program as:

$$\mathrm{E}[CPI] = (E[TPI_{CPU}] + \alpha \cdot E[TPI_{Mem}]) \cdot F_{CPU}$$

Where α is the fraction of instructions that miss in the LLC and F_{CPU} is the processor's operating frequency. TPI_{CPU} represents the average time that instructions spend on the CPU, whereas TPI_{Mem} represents the average time that an LLC-missing instruction spends stalled waiting for main memory. The value α of can easily be calculated as the ratio of LLC misses and total instructions.

Whereas the expected time per CPU operation is insensitive to changes in memory speed (for simplicity, we assume it is fixed), TPI_{Mem} varies with memory subsystem frequency. To model the time per cache miss, we decompose the memory access time into DRAM bank processing time, memory bus transfer time, and MC queuing time, each of which is estimated using the performance counters discussed above.

Full System Energy Model: Simply meeting the CPI loss bound for a given workload does not necessarily maximize energy-efficiency. In other words, though additional performance degradation may be allowed, it may save more energy to run faster. To determine the best operating point, we construct a model to predict full-system energy usage. For memory frequency f_{mem}, we define the *system energy ratio* (SER) as:

$$\mathrm{SER}(f_{mem}) = \frac{T_{f_{mem}} \cdot P_{f_{mem}}}{T_{Base} \cdot P_{Base}}$$

$\mathrm{T}_{f_{mem}}$ is the performance estimate for an epoch at frequency f_{Mem} . $\mathrm{P}_{f_{mem}} = P_{Mem}(f_{mem}) + P_{NonMem}$, where $P_{Mem}(f)$ is calculated according to the model for memory power in [8], and P_{NonMem} accounts for all non-memory subsystem components (for simplicity, we assume P_{NonMem} to be fixed). T_{Base} and P_{Base} are the corresponding values at a nominal frequency. At the end of the profiling phase of each epoch, MemScale calculates SER for all frequencies that can meet the performance constraint given by the slack, and selects the frequency that minimizes SER. As we consider only ten frequencies, it is reasonable to exhaustively search the possibilities and choose the best. In fact, since this search is only performed once per epoch (5 ms by default), its overhead is negligible.

Evaluation

Methodology: Since the few hardware mechanisms that we propose are not yet available, our evaluation is based on simulations. To reduce turnaround times, our simulations are done in two steps. In the first step, we use M5 [9] to collect memory access (LLC misses and writebacks) traces from a variety of multi-programmed SPEC2000 and SPEC2006 workloads running on a 16-core server. In the second step, we replay the traces using our own detailed memory system simulator. This simulator models all aspects of the operating system, MC, and memory devices that are relevant to our study, including behavior profiling, memory channel and bank contention, memory device power and timing, and row buffer management. Our nominal simulation configuration includes 4 DDR3 channels, each populated with two registered, dual-ranked DIMMs with 18 DRAM chips each. Each channel supports 10 possible frequency steps between 800MHz and 200MHz. Frequency transitions require 512 memory cycles plus 28 ns. We estimate DRAM power according to a model published by Micron [8]. We assume that MC voltage varies over the same range as the cores (0.65V-1.2V), as its frequency changes.

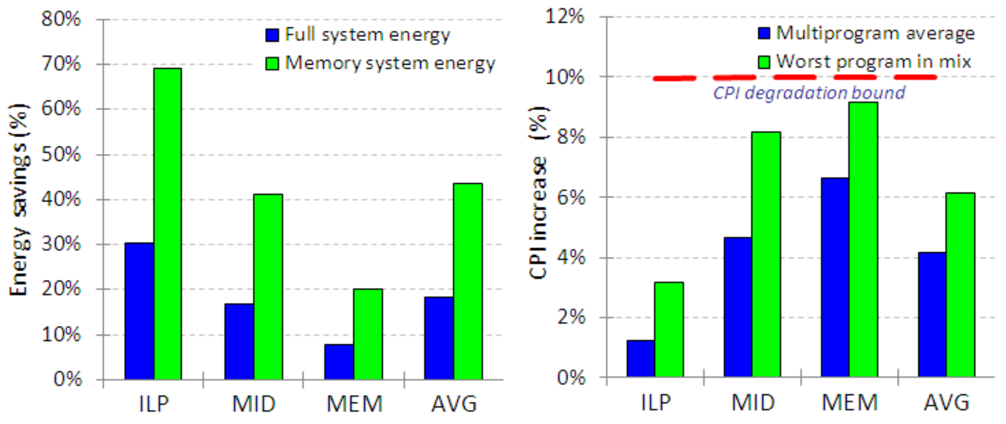

Figure 3. MemScale energy savings (left) and CPI increase (right).

Energy and Performance: We evaluate MemScale with three categories of multi-programmed workload mixes: ILP (compute-bound mixes), MEM (memory-bound mixes), and MID (intermediate mixes). Figure 3 (left) shows the average memory and full-system energy savings that MemScale achieves for the three classes of mixes, compared to a baseline system that keeps the memory subsystem at its highest voltage and frequency. MemScale is configured with a maximum performance degradation bound of 10%. The memory energy savings range from 20% to 69%, whereas the system energy savings range from 8% to 30%. As one would expect, the ILP workloads achieve the highest gains (system energy savings of at least 29%. These workloads can keep the voltage and frequency of the memory system at their lowest possible value all the time. The savings achieved by the MID workloads are lower but still significant (system energy savings of at least 15%). The MEM workloads achieve the smallest savings (system energy savings of at least 6%), since their greater memory traffic reduces the opportunities for significant voltage and frequency scaling.

Figure 3 (right) shows the corresponding average and worst-case performance degradation, all of which fall under the configured 10% bound. When we average the performance degradations of all the applications in each workload, this average is never higher than 7.2%. Again, as one would expect, the degradations are smallest for the ILP workloads, followed by the MID workloads, and then the MEM workloads. On average, MemScale reduces *system energy* by 18% and degrades performance by 4% across all applications and workloads.

Dynamic Behavior: To further illuminate our results, let us consider MemScale's dynamic behavior for an example workload. Figure 4 shows (a) the memory subsystem frequency selected by MemScale for workload MID3, and (b) the CPI of each application in the workload. The figure shows several interesting frequency transitions. Initially, MemScale quickly reduces memory system frequency to the minimum until the abrupt phase change of application *apsi* near the 46ms time quantum. In the subsequent time quantum (around 51 ms), MemScale's policy detects the phase change and increases the frequency. Despite this brief reaction delay, MemScale maintains performance degradation for *apsi* (8.2%) well under the 10% bound.

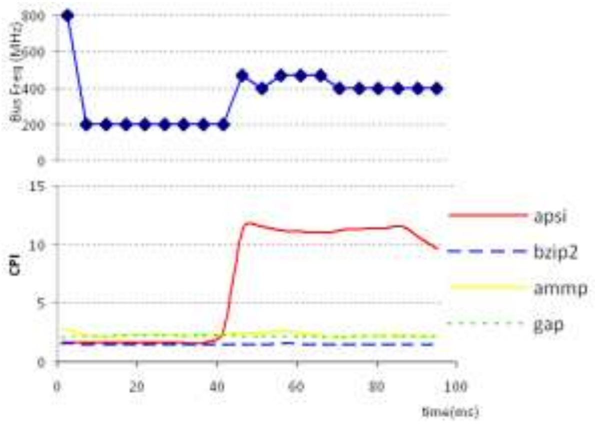

Figure 4. Dynamic behavior.

Comparison with Other Policies: We compare MemScale for the MID workloads with state-of-art memory energy conservation techniques in Figure 5. "Fast-PD" models the DDR3 fast-exit power-down idle state exploited by many current MCs. "Decoupled" refers to the Decoupled DIMM proposal of Zheng et al. [10], which combines low-frequency memory devices with high-frequency channels. "Static" represents the scenario in which the best single frequency for the MC, channels, DIMMs, and DRAM devices is selected statically. "MemScale" refers to our approach, whereas

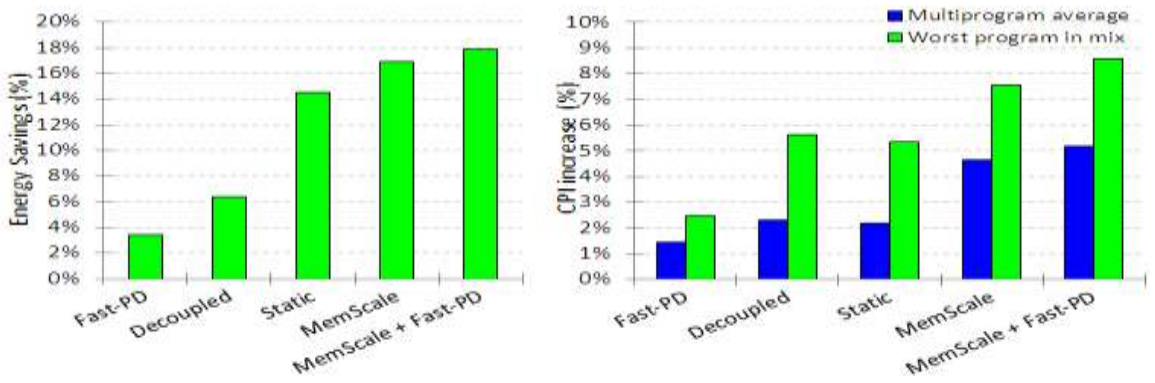

Figure 5. Comparing MemScale to other approaches.

"MemScale + Fast-PD" combines MemScale with fast-exit power-down. Figure 5 demonstrates that Fast-PD provides the least leverage for energy savings, but also incurs only a small performance loss. Decoupled provides greater savings by reducing the energy consumption of the DRAM devices, but disproportionately slows down memory intensive workloads. Static conserves more memory and system energy than either of the prior approaches, but is not implementable in practice because it requires priori knowledge of the optimal frequency. MemScale provides greater savings than any of the exiting approaches, while still ensuring that performance degradation does not exceed the configured 10% bound for any workload. MemScale reduces DRAM, PLL/register, and MC energy more than the other approaches. MemScale's greater savings come from its ability to adjust frequency (as well as MC voltage) dynamically to react to phase changes in each workload. Combining MemScale with fast-exit power-down improves savings, at the cost of a further slight increase in performance degradation.

Other Studies: In our more detailed article on MemScale [11], we examine sensitivity to a wide variety of parameters, including the impact of the performance degradation bound, number of channels, fraction of memory power, and power-proportionality of the MC. These studies show that MemScale achieves substantial energy savings across this entire space, always assuring the selected performance bound is not violated.

Conclusions and Potential Impact

In this paper, we proposed MemScale, an approach to manage memory subsystem energy under performance constraints. Our evaluation demonstrated that MemScale conserves significant energy, while staying within pre-set performance limits. Our results also showed that MemScale is superior to four competing energy management techniques.

We conclude that MemScale's potential benefits far outweigh its small hardware costs. In fact, we believe that MemScale exhibits a few characteristics that may produce a lasting impact on both academia and industry. Next, we discuss these characteristics in turn.

1. Active Low-Power Modes: Many classes of systems require *active* low-power modes to enable energy savings without excessive performance degradation. For example, although servers are often under-utilized [1], it is rare for entire memory ranks to be idle long enough to justify entering a sleep state. When idle times are short, transitioning to these states can produce significant performance degradation.

Interestingly, the memory bandwidth of these servers is often over-provisioned, even when server utilization is fairly high [12,13]. In fact, although server workloads are often sensitive to memory latency, they rarely are sensitive to peak memory bandwidth.

Finally, background energy represents a significant contributor to overall memory energy. Techniques that address solely dynamic, access-driven memory energy miss substantial savings opportunities.

Considering all these facts, it becomes clear that active low-power memory modes like those introduced by MemScale have a significant role to play in server systems. Corroborating this observation, since MemScale was first published, an ISCA paper [12] has argued that *only* active low-power modes can conserve substantial energy for Google's search workloads. David et al. have also considered active low-power modes for main memory in [14].

2. MC Power Management: Server processors are integrating multiple increasingly sophisticated MCs (e.g., [15]), partly due to the growing impact of memory on performance and energy. It is clear that techniques that do not tackle MC energy consumption will miss significant energy savings opportunity. MemScale seamlessly combines the memory channel and device management with MC energy management.

3. Simplicity: Although often overlooked in academia, simplicity is a key requirement for practical adoption by industry. MemScale relies primarily on features that already exist in current systems. Specifically, today, most servers can configure memory channel and device frequency statically at boot time (via the BIOS). MemScale enables dynamic frequency changes by leveraging simple performance/power models (implemented in software), and a modest number of new performance counters.

4. Effectiveness: MemScale provides consistent energy savings while limiting performance degradation to user-defined bounds. It thus enables users to strike any desired trade-off between energy conservation and performance. Techniques that do not enable such trade-offs are less appealing for practical adoption.

5. Rich Possibilities for Future Work: It is difficult to over-estimate the impact that CPU DVFS has had in both academia and industry. Hundreds of papers have been written on CPU DVFS, studying how best to use it for energy conservation, limiting power, and controlling temperature. These studies have proposed new languages, compilers, operating and distributed systems, and architectures that leverage CPU DVFS. Modern processors include DVFS mechanisms and policies in hardware. MemScale opens up a similar set of opportunities and challenges. Active low-power modes can be used in the same ways and at the same levels in memory as in CPUs. Hence, we expect to see many future papers tuning and exploiting these modes.

Importantly, we believe that MemScale will have a significant long-term impact despite the fact that the benefits of voltage scaling will decrease over time. Recall that MemScale uses voltage scaling only for the MC. Frequency scaling the memory subsystem has substantial background energy benefits.

Acknowledgements

This research has been partially supported by NSF under grants #CCF-0916539, #CSR-0834403, #CCF-0811320, and gifts from Google.

References

[1] L. Barroso and U. Hoelzle, The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines. *Morgan Claypool*, 2009.

[2] V. Delaluz, M. Kandemir, N. Vijaykrishnan, A. Sivasubramaniam, and M. J. Irwin. Hardware and Software Techniques for Controlling DRAM Power Modes. *IEEE Transactions on Computers,2001.* 50(11). 1154-1173.

[3] A. Lebeck, X. Fan, H. Zeng, and C. Ellis. Power Aware Page Allocation. In *Proceedings of ninth the International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-IX).* ACM, March 2000. 105-116.

[4] X. Li, Z. Li, F. M. David, P. Zhou, Y. Zhou, S. V. Adve, and S. Kumar. Performance-Directed Energy Management for Main Memory and Disks. In *Proceedings of the 11th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-XI).* ACM, March 2004. 271-283.

[5] H. Zheng, J. Lin, Z. Zhang, E. Gorbatov, H. David, and Z. Zhu. Minirank: Adaptive DRAM Architecture for Improving Memory Power Efficiency. *In Proceedings of the the 41st annual IEEE/ACM International Symposium on Microarchitecture (MICRO 41).* IEEE Computer Society, December 2008. 210-221.

[6] E. Cooper-Balis and B. Jacob. Fine-grained Activation for Power Reduction in DRAM. *IEEE Micro*, 30(3), May/June 2010. 34-47.

[7] A. Udipi, N. Muralimanohar, N. Chatterjee, R. Balasubramonian, A. Davis, and N. P. Jouppi. Rethinking DRAM Design and Organization for Energy-Constrained Multi-Cores. In *Proceedings of the 37th annual International Symposium on Computer Architecture (ISCA '10).* ACM, June 2010. 175-186.

[8] Micron. Calculating Memory System Power for DDR3. Technical Note TN-41-01, Micron, 2007.

[9] N. Binkert, R. Dreslinski, L. Hsu, K. Lim, A. Saidi, and S. Reinhardt. The M5 Simulator: Modeling Networked Systems. *IEEE Micro,* July 2006. 52-60.

[10] H. Zheng, J. Lin, Z. Zhang, and Z. Zhu. Decoupled DIMM: Building High-Bandwidth Memory System Using Low-Speed DRAM Devices. In *Proceedings of the 36th annual International Symposium on Computer Architecture (ISCA '09).* ACM, June 2009. 255-266.

[11] Q. Deng, D. Meisner, L. Ramos, T. F. Wenisch, and R.Bianchini. MemScale: Active Low-Power Modes for Main Memory. In *Proceedings of the sixteenth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '11).* ACM, March 2011. 225-238.

[12] D. Meisner, C. Sadler, L. Barroso, W-D. Weber, T. F. Wenisch. Power Management of Online Data-Intensive

Services. In *Proceedings of the 38th annual International Symposium on Computer Architecture (ISCA '11).* ACM, June 2011. 319-330.

[13] C. Kozyrakis, A. Kansal, S. Sankar, and K. Vaid. Server Engineering Insights for Large-Scale Online Services. *IEEE Micro*, 30, 4 (July), 2010. 8-19.

[14] H. David, C. Fallin, E. Gorbatov, U. R. Hanebutte, and O. Mutlu. Memory Power Management via Dynamic Voltage/Frequency Scaling. In *Proceedings of the 8th International Comference on Autonomic Computing, (ICAC '11).* ACM, June 2011. 31-40.

[15] E. Ipek, O. Mutlu, J. F. Martinez, and R. Caruana. Self-Optimizing Memory Controllers: A Reinforcement Learning Approach. In *Proceedings of the 35th annual International Symposium on Computer Architecture (ISCA '08).* ACM, June 2008. 39-50.