Evaluating Power Management Strategies for High Performance Memory (DRAM): A Case Study for Achieving Effective Analysis by Combining Simulation Platforms and Real Hardware Performance Monitoring

Karthick Rajamani, Juan Rubio, Wael El-Essawy, Kartik Sudeep and Ram Rajamony
karthick@us.ibm.com, rubioj@us.ibm.com, wrelessa@us.ibm.com, kartik@us.ibm.com, rajamony@us.ibm.com

IBM Austin Research Lab

Abstract

Developing dynamic power management strategies for high performance systems requires a good understanding of the power-performance trade-offs underlining the workload-strategy-system interactions. The feasibility of detailed simulations to help here is severely limited by the range of problem sizes that can be simulated and simulation speed. For our work we want to understand the effectiveness of DRAM power management for a 'realistic' workload - RF-CTH - about 500K lines of Fortran-C benchmark developed at Sandia National Labs. The target architecture is a CMP-based building block for a future high performance computing architecture. We address the scalability and speed limitations for detailed simulation by adopting a multi-step characterization and simulation approach incorporating analysis on current generation hardware using processor performance counters, a fast memory hierarchy simulation engine, a detailed system-level simulator and a power-performance simulator for the DRAM subsystem.

1. Introduction

Detailed full-system simulation is emerging as a key component of computer systems design. Simulators such as Mambo [**1**] and Simics [**2**] provide the ability to model system-level activity in addition to key application activity on a detailed architectural model for the system. Often, such detailed modeling is critical to evaluate the true impact of different design alternatives. While the increased computational power of commercial hardware (that forms the infrastructure for such simulations) makes system-level modeling and simulation feasible, combining the detailed modeling of system-level interactions and detailed timing for architectural-level events still places significant limits on the scale of simulations. Further, when the systems being simulated are larger than the infrastructure for simulations it brings a fresh set of limitations. In this work, we present our approach to tackling these limitations while evaluating the impact of memory (DRAM) power management for a future system design.

The increasing densities from silicon technology and system integration combined with ever-growing demands for computational capacity has brought system power management issues to the forefront in computer design. While processors are dominant power consumers in any computing system, for mid-range and high-end systems, the power consumption of the DRAM sub-system tends to dominate even that of the processors [**3**]. Adopting performance-sensitive power management solutions for the memory system will be a critical component of any system-level power management strategy for larger systems.

Evaluating a memory system power management strategy requires detailed modeling of

- Workload-induced memory system activity,
- System-level and architectural interactions that together with the workload determine the spatial and temporal intensity of DRAM activity,
- DRAM timing and current consumptions, and
- Interaction of all three with the power management strategies.

While the task requires detailed system- and architectural-level modeling it is also critical that real applications be used to drive the simulation as they often exhibit very different behavior from kernel-level benchmarks often used for design-time system parameterization. In this work, we use the RF-CTH code [**4**], a solid mechanics code developed at the Sandia National Labs, as representative of a real workload and

the Stream benchmark as representative of the memory intensive kernel.

Our detailed simulation environment, using Mambo, is limited both in the speed of simulation because of the detail we want to model and in the scale of the problem/system we can simulate as the underlying hardware is significantly small relative to the system we want to model. We address these limitations by adopting a multi-step analysis and modeling approach.

In the first step, we analyze the application on current generation hardware – identify key phases of the application and memory-system related characteristics of each phase. In this step, we collect application characteristics information to drive the next two stages. Based on step 1, we decide on a 'reduced' problem size that can be used to simulate the 'real' problem size. In step 2, we then use a fast memory hierarchy simulator to arrive at the 'revised' system configuration for running the 'reduced' problem size so that it mimics (with respect to activity at the DRAM level) the 'real' problem size on the 'real' system configuration. Finally, in step 3, using the 'reduced' problem size and 'revised' system configuration arrived at in step 2, we evaluate the power management strategy. For this step, we use detailed power and performance simulations by combining the Mambo simulator with a detailed DRAM sub-system power-performance simulator, Memsim [**5**].

There have been a number of efforts to support detailed power and/or thermal simulations by coupling them with detailed architectural level simulators [**6**, **7**, **8**]. While their work is critical to provide a basis for joint power-performance modeling they still face the same limitations with respect to simulation speed and scale as our base infrastructure. An alternative approach to tackling the simulation scale is a proposal for re-scaled benchmarks that presume to capture the key characteristics of real commercial workloads [**9**]. In contrast to their generic solution to the simulation scaling issue, we adopt a more custom approach, where the scale down is tailored to the specific workload(s) and evaluation of interest. This obviously necessitates additional effort but provides the benefits of being a better match with respect to the scenario for evaluation.

The rest of the paper is organized as follows. Section 2 details the issues relevant to investigating DRAM power management – memory power consumption, organization and power management strategy, workloads used for evaluation and the problems faced in evaluating power management strategies. Section 3 gives a brief overview of the infrastructure we use for conducting this investigation. Section 4 outlines our methodology that addresses the problems underlying the investigation. In section 5, we present the results of the analysis by employing our methodology and briefly conclude in section 6.

2. Investigating DRAM Power Management

Independent studies [3] have shown that the DRAM sub-system can be the dominant component of server power consumption. To address this we investigate the usage of the *powerdown* mode of commercial SDRAM. In powerdown, the *CLKE* signal to the DRAM device is disabled and no accesses can be made. As it eliminates the propagation of the *CLK* signal within the device it cuts down the DRAM switching power. Figure 1 shows the estimated relative DRAM power consumption for different states for a DDR3 device (same is modeled in our experiments).

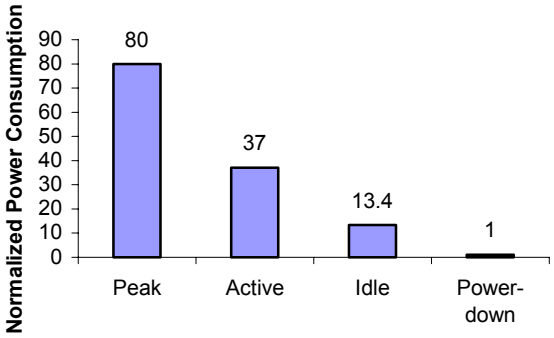

Figure 1: Relative DRAM Power Consumption for Different States.

While powerdown can save considerable power for an idle device it should be used judiciously in performance-sensitive systems as there are entry and exit latencies associated with its transitions and also a minimum duration for each CLKE state (i.e. in powerdown and powered-up). In our evaluations, we model 2 memory cycle latencies for entry and exit and 3 memory cycles for the minimum duration – same as for DDR2, as the DDR3 specifications for these parameters were not yet available. We call the strategy utilizing powerdown that we evaluate in this paper as *queue-aware immediate powerdown* – the memory controller tracks activity in each *rank* (set of devices servicing a single request) and as soon as all the *banks* of a rank are idle and the memory controller queues have no more requests for that rank, the devices of the rank would be transitioned to the *powerdown* mode. Powered-down ranks would be powered up for device *refresh* or for servicing new requests. Our approach provides a simple, easily implement-able solution to DRAM power management.

2.1 System Description and Memory Organizations

The system we are modeling is a chip-multiprocessor (CMP) based building block for a high-performance computing system design. Each CMP chip has a) 4 cores running at 6GHz, b) each with its own 64K L1 D, c) two 2MB L2s (each shared by two cores) and d) a 32MB L3. Each chip has a resident memory controller with 2 4-channel high-speed interfaces connected to fully buffered DIMMS for DDR3-1600 SDRAM (similar to the industry-standard fully-buffered DIMM architecture proposed for DDR2 and beyond), with up to 4 *quad-ranked* DIMMs per channel and a total DRAM capacity of 128GB. All requests to DRAM are at 128-byte cache line size. As memory system performance is closely tied to memory organization, we investigate the impact of powerdown by varying the organizations along two dimensions – the number of channels servicing a single memory request and the number of ranks.

When more channels service a cache request, the data transfer time from DRAM is reduced. However, more devices are utilized for the transfer. The device access energy cost (or active energy) is composed of two main components – activating the row in the DRAM device and the actual read or write operation. The activation energy is solely dependent on the number of devices. The energy for the actual read or write data transfer is roughly the same whether fewer channels (devices) are used to service the request with a longer burst length or more channels are used with a shorter burst length. Thus, fewer channels servicing each request result in lower access energy costs (and active power components) at the cost of slightly longer response times. In our evaluations, we consider two alternatives – 2 channels servicing a request corresponding to a DRAM request burst length of 8 (i.e. 8 transfers are required on the DRAM data pins for a 128-byte transfer) and 4 channels servicing a request corresponding to a DRAM request burst length of 4 (i.e. 4 transfers are required on the DRAM data pins).

When we vary the number of ranks in the system, we vary the available concurrency in the system. In our evaluations, we use the *close page* policy typically followed in server systems – in this policy once the request is serviced the corresponding *bank* in the DRAM devices is pre-charged, which implies that a new request to the same bank would require an activation command to be issued before the actual read or write command. The precharge-activation sequence imposes a minimum restriction of *tRC* between two subsequent requests serviced from the same *bank.* To mitigate this limitation, DRAM devices provide multiple banks on a device, which can be utilized for even a sequential stream by interleaving consecutive addresses across the banks. However, large server systems typically also use multiple ranks of devices with interleaving across the ranks so that the concurrency across the ranks further mitigates the limitation. In general, the ranks (and banks) must be sufficient so that the device bandwidth rather than *tRC* becomes the limiting factor, if any, in servicing memory requests. Note that interleaving across the ranks and banks has an important implication on the simulation approach – a smaller memory sub-system can be effectively substituted in the simulation for a larger real memory system as the interleaving ensures that the intensity of accesses would be similar on both systems as long as the upper level memory hierarchy behavior is the same.

In this evaluation, we consider three different number of ranks – 4, 8 and 16 for each channel. In order to consider the same problem sizes, we match the 4 rank system with 4Gb device size, 8 rank system with 2Gb device size and 4 rank system with 1Gb device sizes – the total memory capacity in all three cases being 128GB. Note that device currents do not increase in proportion to device sizes. So the idle energy component of a system with fewer ranks (larger devices) is lower than the idle energy component of a system with more ranks (smaller devices) for the same total memory capacity.

2.2 Workloads

The key focus of our work is analyzing the impact of power management for real workloads. With our system targeting high-performance computing, we consider the RF-CTH application as a candidate workload. As a comparison point we also evaluate the strategy with the Copy kernel from the STREAM benchmark [**10**].

2.2.1 RF-CTH/CTH

CTH is a multi-material, large deformation, strong shock wave, solid-mechanics code developed at Sandia National Laboratories. It has models for different materials and meshes. It uses second-order accurate numerical methods to reduce dispersion and dissipation and produce accurate, efficient results. It is used extensively within the Department of Energy laboratory complexes for studying armor/anti-armor interactions, warhead design, high explosive initiation physics and weapons safety issues. CTH is a sensitive application because of its modeling capability. The application we use in this paper, RF-CTH (reduced functionality CTH) is a sanitized version of CTH with the sensitive physics calculations removed. However, it is supposed to capture the essential computing characteristics of the original application.

RF-CTH is implemented in Fortran and consists of approximately 500,000 lines of code. We use MPI for its parallelization with each processor working in its own address space. Given the voluminous nature of this application, it is impractical to analyze it directly on the simulator. Section 4 discusses our approach to this task and section 4.1 provides RF-CTH's characteristics pertaining to our analysis. For the system, described in section 2.1, we decide on the RF-CTH problem size by the largest memory footprint it can accommodate i.e. a problem size of approximately 128GB.

2.2.2 MP STREAM - Copy Microkernel

The STREAM benchmark is widely used to assess system bandwidth limits in high performance computing. It consists of four kernels/loops. We use the Copy loop from the benchmark working on a footprint large enough to miss in all the caches as a stress-test case for the power management strategy. Essentially, the bandwidth requirements of such a workload should offer the minimum idle periods at DRAM device-level and, consequently, the minimum opportunity of employing powerdown. Alternatively, if improperly used i.e. used when there are requests to be serviced and, thus, delaying them, it would also show the maximum performance penalty for employing powerdown. For the 4-way core, we use a multi-processor version of the kernel with each processor handling one-fourth the array space.

2.3 Issues in Evaluating DRAM Power Management

To obtain the performance and power impact of utilizing powerdown it is critical to capture the extent of idleness of each DRAM device and the interval between requests to the same device rank and bank. Capturing this requires an adequately detailed modeling of DRAM level activity. However, activities at the DRAM level are dependent on the entire memory hierarchy. So it is also important to capture the access rates and hit rates for all levels in the memory hierarchy.

The two main limitations for a straightforward detailed simulation to obtain activity at the DRAM level are:

- The problem scale that we desire to model is 128GB while the simulation platform has only 4GB of physical memory.
- The detailed simulation with Mambo has a speed in the range of 500,000 instructions per second (for a 4-way multiprocessor simulation) while running a problem size that is a factor of four smaller than the desired problem size on existing hardware (Power4+ processors at 1.45 GHz) was timed to take over 11 hours (equivalent simulation time of 798 days at 500kips).

In the next two sections, we present the additional infrastructure and methodology used to address these limitations.

3. Infrastructure

As outlined in section 1, we use a variety of tools to assist in our analysis. In this section we give a brief description of them.

3.1 Power4 Performance Counters

Our platform for characterizing CTH is an IBM p650 8-way SMP using 1.45GHz Power4+ processor. The Power4 architecture supports 64 different groups of 8-set performance event counters i.e. they can be programmed to monitor 8 events concurrently. For our analysis, we use a set of groups to obtain the IPC and entire memory hierarchy access information. We use a custom performance monitor library for the Power4 counters on Linux that allows us to sample the counters at up to 10ms intervals with little overhead – we use a sampling interval of 100ms for our analysis. The library allows us to track the IPC with memory hierarchy usage over time enabling us to identify key application phases and their characteristics pertaining to the memory hierarchy such as cache miss rates, access rates etc.

3.2 Mambo

Mambo [**1**] is a full-system simulation toolset for modeling PowerPC-based systems. It provides a range of processor simulators ranging from functional to detailed timing-accurate models. Mambo simulates the entire hardware system: processors, memory hierarchy, disks, network, and other devices, which enable it to boot the operating system. PowerPC applications run on the simulated operating system without modification. Mambo has a large set of configuration parameters, which allows the user to model a wide range of full computer systems. We perform our Mambo simulations on a 2-way Opteron-based Linux box with 4GB of memory.

3.3 E-mambo

E-mambo [**11**] is an execution-driven simulation engine that maintains statistics for applications while executing them natively on a PowerPC machine. Instructions that are executed on the processor are simultaneously simulated by e-mambo thereby maintaining the state of the simulated general-purpose registers and special registers. Architectural studies can be conducted with the help of e-mambo, which provides

a very simple and efficient interface for plugging in any simulated architectural feature. Some of the event handlers that are provided in e-mambo as plug-ins:

- Count - reports simple execution statistics for the program (number of instructions, number of memory references, number of branches, etc.)
- Cache - simulates a cache hierarchy with specified size and associativity for each level and reports hit/miss statistics for the application.
- Locality - simulates an infinite L1-cache to obtain different measures of locality of memory references for the application.
- Tracer - collects instruction traces and memory reference traces for the application.

We use the cache plug-in for tracking the memory hierarchy characteristics. Note that e-mambo is not a full-system simulator. It runs on top of an operating system and we primarily use it for application analysis and preliminary architectural explorations. As instructions are executed on the native hardware it is quite fast and often less constrained with respect to problem sizes that can be modeled compared to detailed full-system simulators.

3.4 Memsim

Memsim [**5**] is a power-performance simulator for SDRAM-based server memory systems. It supports modeling of complex memory system organizations, different interleaving options, different page-mode options, and power management strategies including low-power modes such as powerdown. It supports detailed DRAM timing models and computes DRAM power/energy consumption by tracking activities at the device level accurately. Memsim is highly parameterizable with respect to DRAM device characteristics and memory system organization. It is implemented using an event-driven model and written in C using the CSIM toolkit [**12**]. It can be used in two modes: a memory trace-driven mode and a component-model mode where it can be integrated into a larger simulation setup as a plug-in memory system simulator. In our setup, as Mambo was augmented to model the detailed memory system organization for performance modeling we use Memsim in its trace-driven mode with traces generated by Mambo runs, while ensuring that an identical memory system is modeled in both simulators.

4. Methodology

As mentioned in section 1, we use a multi-step process for our analysis to circumvent these problems. Our steps consist of

- Characterizing the workload on hardware to determine distinct phases, characteristics of the phases, and impact of problem size. Analysis in this step also determines what problem size we choose to use for the detailed simulation.
- Identifying suitable system (cache) parameters so that detailed simulation for the simulator problem size yields similar behavior to the real problem size on the desired system configuration.
- Detailed simulation with power management strategy to quantify impact on power and performance.

4.1 Characterizing Workload

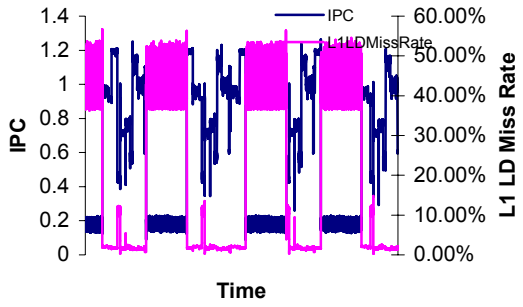

Figure 2: Key Phases for CTH

Figure 2 shows the data from performance counter analysis for a 9GB problem size on one processor – it shows two key characteristics, the IPC and L1D Miss ratio, varying with time. We can identify two distinct phases – a low IPC phase characterized by high L1D Miss ratio and a higher IPC phase characterized by significantly lower L1D Miss ratio. The phase durations are relatively uniform and the phases alternate at regular intervals. Examination of characteristics for other levels in the memory hierarchy provides unique, repeating characteristics relevant to those levels. But, the most distinct phase identification is based on the L1 Miss ratios and associated IPC impact. Based on this behavior, we decide that for detailed simulation we need to capture the memory hierarchy behavior for these two phases accurately. Further, as these phases repeat, we can reduce simulation time by simulating execution of a single "iteration" that contains these two phases (we actually simulate 2 iterations). Note that as RF-CTH employs MPI-based parallelization (with a process per processor) there is little inter-processor sharing of cached data in multi-processor executions. And with equal-sized partitioning that we observed during the hardware execution, the processes share the caches equally which makes it relatively straightforward to infer multi-processor memory hierarchy behavior from a single processor execution.

4.2 Choosing a Simulation Problem Size

Because of the MPI-based parallelization, a 128GB problem size for a 4-way creates similar memory hierarchy activity as a 32 GB problem size for a 1-way with one-fourth the size for the shared cache levels (L2 and L3). So we can use a uniprocessor 32GB problem size execution to obtain the memory hierarchy activity for the 4-way 128GB problem size. However, given our infrastructure, a 32GB uniprocessor execution was also not feasible because of the physical memory size limits of our hardware. We decided to then identify what smaller uniprocessor problem size could emulate the memory hierarchy behavior of a 32GB uniprocessor problem size.

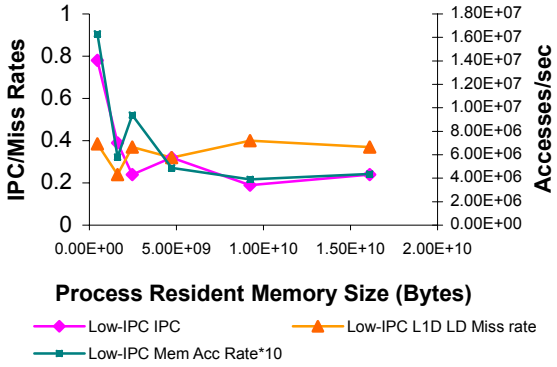

Figure 3: Change in Performance/Characteristics with Problem Size

Figure 3 shows the variation in IPC, L1D LD hit rate and the DRAM access rate with problem sizes on current IBM Power4-based hardware. The different problem sizes are generated by using a different set of valid xyz dimension size for each. As the problem size increases we see the statistics stabilize (the behavior is shown for three statistics for the low-IPC phase, a similar trend is seen for other statistics and for the high-IPC phase). We see that the statistics are essentially similar for problem sizes beyond the 4.7GB range (the fourth point from left in each statistic's line). We conclude that we can emulate the memory hierarchy behavior of a 32GB uniprocessor problem size with any problem size larger than 4.7GB. From this, we decide to use a 9GB problem size for a uniprocessor execution to obtain the memory hierarchy characteristics (this is the 'standard' problem sizes recommended for analyzing the RF-CTH code by its authors).

By this two-step reduction, we conclude that we can faithfully capture the memory hierarchy behavior of a 4-way 128GB problem size RF-CTH execution on a target memory hierarchy of 4x64K L1D, 2x2MB L2 and 32MB L3 execution by a 1-way 9GB problem size execution on a memory hierarchy of 1x64K L1D, 1x1MB L2 and a 8MB L3.

4.3 Choosing System Model Parameters for Simulation

For detailed simulation, we are faced with the constraint that the sum of the application and operating system physical memory requirements should fit within the simulator's physical memory (4GB). From the available smaller problem sizes (determined by valid xyz dimension sizes dictated by RF-CTH sizing rules), we choose the largest one that fits this constraint – a problem size of 1.6GB. This is the problem size for a 4-way execution and a 1-way problem size would be about 4 times smaller. However, as seen in Figure 3, such a problem size would not share the same memory hierarchy statistics as a 9GB problem size that we identified as the equivalent 1-way problem size for our real workload. This implies we need to reduce the cache sizes for the chosen simulator problem size of 1.6GB.

We use e-mambo for identifying the revised cache sizes. For this, we first simulate the 9GB problem size with the cache sizes identified at the end of last section - 1x64K L1D, 1x1MB L2 and a 8MB L3. We obtain detailed time-wise variation of the memory hierarchy statistics – access rates (accesses per instruction) and miss rates for loads and stores at each level. Figure 4 shows a representative L1D LD Miss ratio curve (note that the cache organization modeled here is more aggressive than the hardware caches for Figure 3, hence, the statistics are not directly comparable to those given in Figure 3). We then run the smaller problem size, one-fourth the 1.6GB problem size chosen for the 4-way detailed simulation run, on e-mambo varying the cache sizes and plotting it versus the number of completed instructions (as a measure of progress of execution). For each cache level, we identify the size that gives a stat-curve as close as possible for that level to the corresponding curve of the 9GB problem size. The stat-curves are matched on their peak magnitudes, shape and duty cycle of the two phases among other things.

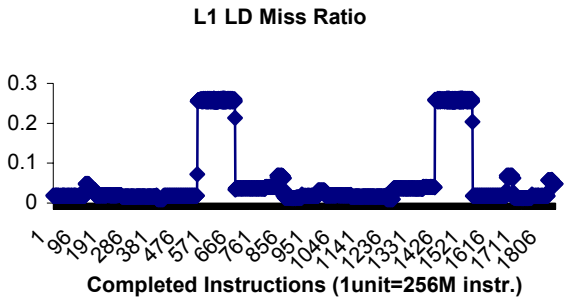

Figure 4: L1 Miss Ratio for the 9 GB problem size

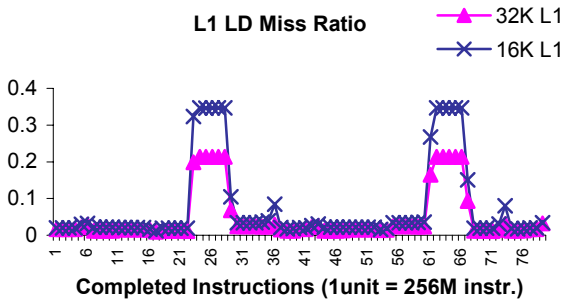

Figure 5: L1 miss ratio for 1-way 400MB (4-way 1.6GB) 'simulator' problem size with different L1 cache sizes.

Figure 5 shows the L1D LD Miss ratio curve for the 400MB problem size for two different L1 cache sizes. Comparing it with Figure 4, one can see that the 32KB size for the L1 D cache provides a better match for this statistic than the 16KB size. A similar matching approach is taken to identify the right size for each of the cache levels by iteratively identifying the right L1 cache size first, then using it when obtaining the L2 size and then using both when obtaining the L3 size. The result of this process is captured in Table 1. Note that this somewhat elaborate but correct procedure results in a not very obvious cache size scale-down for the simulation problem size.

Cache level	Target System (4-way)	Simulation Setup (4-way)
L1D per processor	64KB	32KB
L2 per chip	2x2MB	2x512KB
L3 per chip	32MB	2MB

Table 1: Cache Sizes for Target System and Simulation Environment

4.4 Detailed Simulation Setup

Figure 6 shows the interactions for the detailed simulation. Mambo is fed the execution environment (OS, file system), the application binary, and the appropriate configuration parameters. The execution is driven by an input script file that directs when statistic collection and trace generation are to be done (skipping the non-repetitive initialization phase of the application etc). The memory traces generated from Mambo are fed into Memsim. Memsim simulates the detailed activity and power at the DRAM as a result of the request stream, DRAM timing constraints and power management strategy; and, outputs both the time-varying DRAM power consumption and overall DRAM power and performance statistics.

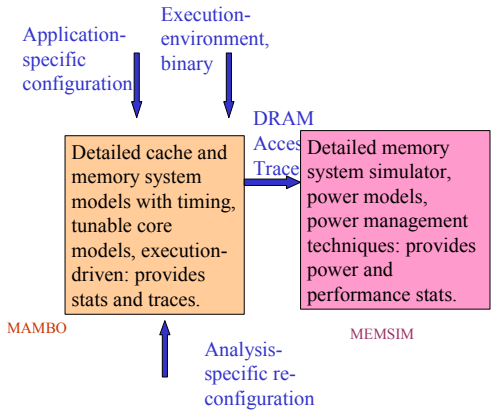

Figure 6: Detailed Power-Performance Simulation with Mambo and Memsim

5. Detailed Simulation Results

In this section we give an overview of the results obtained from the detailed simulation. While these results are the goal for our work, they are not the focus for this paper and so are discussed only briefly.

5.1 Impact of Power Management

Figure 7 shows the impact of power management using *queue-aware powerdown* (see section 2 for description), for RF-CTH. As a contrast, the savings for the STREAM benchmark's Copy kernel is also shown. RF-CTH shows a 72% reduction in power consumption while even the bandwidth-intensive Copy also obtains a 42% reduction in power consumption. Figure 8 provides

the breakdown of the power consumption for the two to illustrate the reason for the difference in benefits seen by the two. In the real workload, RF-CTH, a significant portion of the power consumption is in idle devices which is the component directly addressed by using the powerdown mode. For the benchmark, more than half the power is active that is consumed by the servicing of access requests and so not addressable by the usage of powerdown. The data shown in the figures is obtained for a 16 rank 4-channel wide configuration.

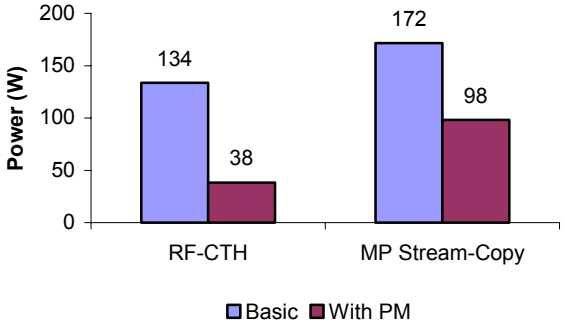

Figure 7: Impact of Power Management

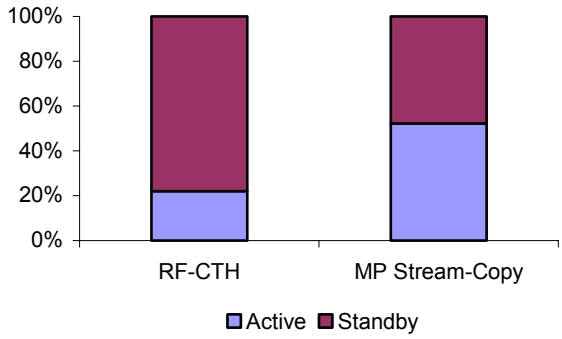

Figure 8: Distribution of Power Components

5.2 Sensitivity to DRAM Organization

In this section, we present the results for different DRAM organizations. Figure 9 shows the performance for RF-CTH and Figure 10 the power consumption. The 1chgrp numbers refer to using all 4 channels in a 4-wide channel group being grouped together to service a request and 2chgrp refers to just 2 channels of a 4-wide channel group being grouped together. Because of RF-CTH's modest bandwidth requirements, the performance is not very sensitive to DRAM organization. However, the larger rank configurations have higher number of devices and so consume significantly higher power (large fraction of which is in idle state). Using powerdown-based power management reduces the power consumption of the larger number of ranks significantly, thus, reducing the difference in power consumption between having larger and smaller number of ranks**.**

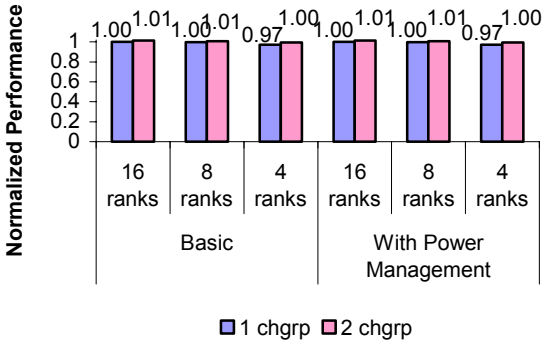

Figure 9: RF-CTH Performance for different DRAM Organizations

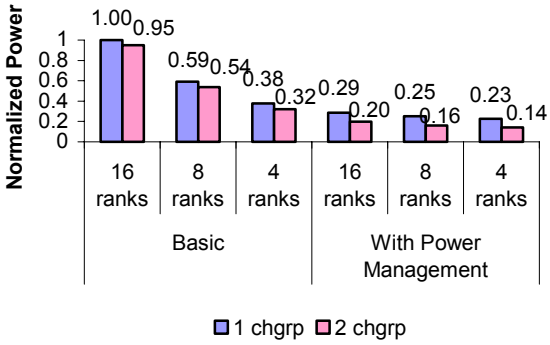

Figure 10: RF-CTH DRAM Power for different DRAM Organizations

Figure 11 and Figure 12 show the performance and power consumption for the MP Stream-Copy kernel executing on all the four processors. Stream has higher bandwidth requirements than RF-CTH. 4 ranks are inadequate to provide this and lower performance compared to 8 or 16 ranks.

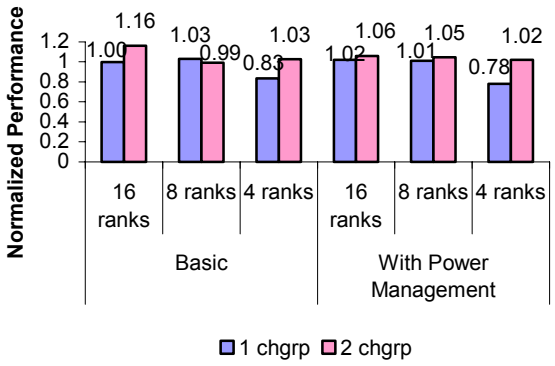

Figure 11: STREAM Performance for different DRAM Organizations

Also with a greater proportion of DRAM power in active state – 2-wide (2chgrp) channel groups has noticeably lower power consumption than 4-wide (1chgrp). Since 2-wide channel group allows more independent requests in flight, it even helps performance a little for the 12 request streams in flight for the benchmark.

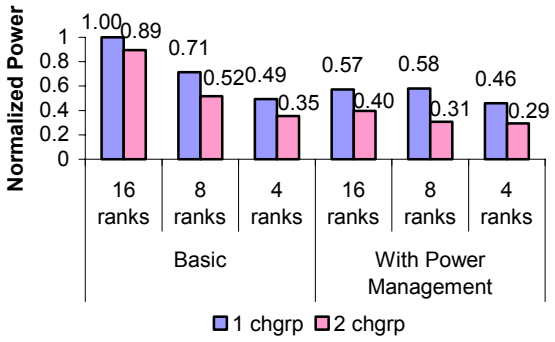

Figure 12: STREAM DRAM Power for different DRAM Organizations

To summarize:

- Power management reduces difference between rank organizations allowing a design with higher available bandwidth (16 ranks) to enjoy power consumption similar to that of a more energy-efficient design (4 ranks).
- Combination of 2-wide channel groups and power-down management attacks both active and idle power consumption yielding the best results.

6. Conclusions

In this paper, we present a methodology for tackling the scaling and speed limitations of detailed simulation-based analysis with real-world workloads. A combination of real hardware analysis and focused (and fast) simpler simulations driven analysis is used to reduce the scale of the simulation problem while retaining the driving characteristics pertinent to the analysis. While we present our approach in the context of evaluating DRAM power management strategies, we believe it to be generally applicable to any memory hierarchy related exploration.

7. Acknowledgments

We gratefully acknowledge Ahmed Gheith, William Speight, Lixin Zhang, Freeman Rawson, Tom Keller, and Mootaz Elnozahy for tools and other help; and, Daniel Phipps, Brett Tremaine, and Paul Coteus for system information that made this work possible. We also thank the anonymous reviewers for their comments and suggestions. This paper is based upon work done in the context of the PERCS project at IBM, which is supported in part by the Defense Advanced Research Projects Agency (DARPA) under Contract No. NBCH3039004.

8. References

[**1**] Mambo -- A Full System Simulator for the PowerPC Architecture - P. Bohrer, M. Elnozahy, A. Gheith, C. Lefurgy, T. Nakra, J. Peterson, R. Rajamony, R. Rockhold, H. Shafi, R. Simpson, E. Speight, K. Sudeep, E. Van Hensbergen, and Lixin Zhang, *ACM SIGMETRICS Performance Evaluation Review*, Volume 31 (4), March 2004.

[**2**] SimICS/sun4m: A Virtual Workstation - Peter S. Magnusson, Fredrik Dahlgren, Håkan Grahn, Magnus Karlsson, Fredrik Larsson, Fredrik Lundholm, Andreas Moestedt, Jim Nilsson, Per Stenström, and Bengt Werner, *In Proceedings of Usenix Annual Technical Conference*, June, 1998.

[**3**] Energy Management for Commercial Servers - Charles Lefurgy, Karthick Rajamani, Freeman Rawson, Wes Felter, Michael Kister, and Tom W. Keller, *IEEE Computer*, Volume 36 (12), December, 2003.

[**4**] CTH: A Software Family for Multi-Dimensional Shock Physics Analysis - E. S. Hertel, Jr., R. L. Bell, M. G. Elrick, A. V. Farnsworth, G. I. Kerley, J. M. McGlaun, S. V. Petney, S. A. Silling, P. A. Taylor, and L. Yarrington, *In Proceedings of the 19th International Symposium on Shock Waves*, Volume 1, July 1993.

[**5**] MEMSIM Users' Guide, Karthick Rajamani, *IBM Research Report RC23431*, October 2004.

[**6**] Wattch: A Framework for Architectural-Level Power Analysis and Optimizations - David Brooks, Vivek Tiwari, and Margaret Martonosi, *In Proceedings of the 27th International Symposium on Computer Architecture (ISCA)*, June 2000.

[**7**] Microarchitecture-level power-performance simulators - Modeling, validation, and impact on design Z. Hu, D. Brooks, V. Zyuban, and P. Bose., Dec. 2003. *Tutorial at Micro-36.*

[**8**] Temperature-aware microarchitecture - K. Skadron, M. R. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan, *In Proceedings of the* 30th International Symposium on Computer Architecture (ISCA), April 2003.

[**9**] Simulating a $2M Commercial Server on a $2K PC: Alaa R. Alameldeen, Milo M.K. Martin, Carl J. Mauer, Kevin E. Moore, Min Xu, Mark D. Hill, David A. Wood, IEEE Computer, Volume 36 (2), February 2003.

[**10**] STREAM: Sustainable Memory Bandwidth in High Performance Computers – John McCalpin, *http://www.cs.virginia.edu/stream/*

[**11**] Application Analysis Using Memory Pressure – Kartik Sudeep and Ahmed Gheith, *To appear in Proceedings of the Third Annual ACM SIGPLAN Workshop on Memory Systems Performance (MSP)*, June, 2005.

[**12**] CSIM 19 Simulation Toolkit, Mesquite Software Inc., *http://www.mesquite.com/documentation/.*