MN-GEMS: A Timing-aware Simulator for a Cloud Node with Manycore, DRAM, and Non-Volatile Memories

Woomin Hwang, Ki-Woong Park, and Kyu Ho Park
Computer Engineering Research Laboratory, KAIST
{*wmhwang, woongbak*}*@core.kaist.ac.kr, kpark@ee.kaist.ac.kr*

***Abstract*—In this paper, we describe a part of our ongoing research project aimed at the management of manycore and Hybrid Main Memory with DRAM and Non-Volatile RAMs (NVRAMs). By the needs of simulation and through investigation of the requirements for the target management system, we found that the simulation platform requires support for manycore, a timing-aware simulation of hybrid memory with DRAM and NVRAM, and a Performance Monitoring Unit (PMU). Therefore, we built MN-GEMS, a full-system simulator for the consolidated VMs of a cloud node satisfying all these requirements.**

***Keywords*-virtual machines; simulator; memory contention; scheduling; NVRAM; hybrid main memory**

I. Introduction

As manycore processors and non-volatile RAMs (NVRAMs) have become promising hardware components for emerging computing systems, many studies have investigated system designs enhanced with the above components for energy efficiency and improvements in performance. In this research direction, we have performed a project, MN-MATE [1], using a novel architecture and management techniques for resource allocation of a number of cores with large sizes of DRAM and NVRAM. As a fundamental way of performing this kind of research, simulation-based designs and evaluations stand out as the most widely used mechanisms. The use of software simulators allows the validation of architecture designs and the exploration of new concepts before actual implementation. Thus, there is a need for a well-defined simulation environment for the study on the manycore system and the NVRAM-based system design. By thoroughly investigating our research direction, we identified three requirements for the simulation environment construction: 1) support for manycore simulation; 2) timing-aware simulation of hybrid main memory based on the DRAM and NVRAM access characteristics; and 3) a Performance Monitoring Unit (PMU).

Table I shows the functionalities of conventional simulation platforms. The last row in Table I clarifies our simulation design goal with respect to desirable functionalities to be achieved. Among the five requirements, M5 and Simics provide only one functionality, even though they support a full-system simulation. They give little consideration to the different access latencies that are in accord with the NVRAM access type and to the memory access request ordering generated from manycore. Even though GEMS has attained a consensus that it can enable DRAM timing simulation functionality as well as manycore support, it cannot support an NVRAM timing simulation, runtime feedback of performance statistics. Finally, previous simulators have not offered much of a solution to reducing long simulation times when timing-awareness is enabled with a full-system simulation.

Table I
Functionalities of full-system simulators

	Manycore	Timing Simulation		PMU	Acceleration
		DRAM	NVRAM		
M5 [2]	X	O	X	X	X
Simics [3]	O	X	X	X	X
GEMS [4]	O	O	X	X	X
MN-GEMS	O	O	O	O	O

To alleviate these limitations, we realized a more advanced simulation platform, MN-GEMS, which meets the above requirements. GEMS is considered a promising foundation for our simulation platform. Our simulation platform addresses the need for manycore support, a timing-aware simulation of multiple types of main memory, and a PMU by devising a memory traffic multiplexer and a reconfigurable memory controller. Moreover, the simulation platform is open and modular, allowing simulation users to produce any kind of NVRAM that they may desire. Our MN-GEMS simulation platform is used as a base simulator for ongoing research issues in the MN-MATE project. The remainder of this paper is organized as follows: In Section II, we elaborate on our simulation platform and its three novel mechanisms, then we conclude with the current status and the next steps.

II. MN-GEMS simulation platform

The primary purpose of MN-GEMS is to simulate a cloud node equipped with manycore and a hybrid main memory with regard to the various access latencies of the DRAM and NVRAMs. To meet this requirement, the simulator models a hardware shown in Figure 1 and performs a full-system simulation. The hardware model is composed of processors containing a number of cores, a two-level cache,

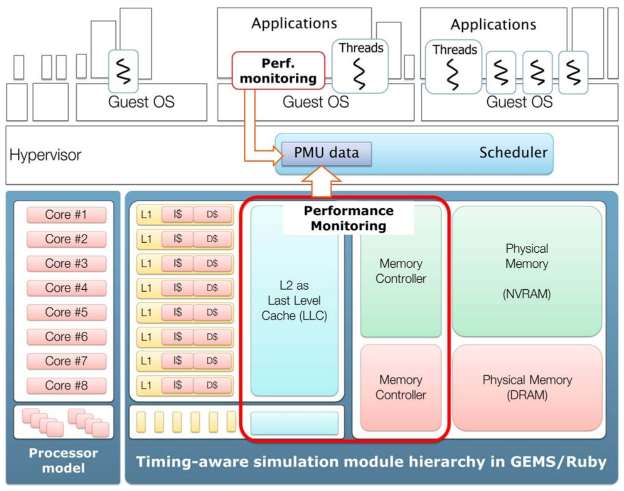

Figure 1. Modeled hardware and execution environment in the simulator

a hybrid main memory, and a PMU that collects per-task events related to the application performance. We built the MN-GEMS by configuring GEMS to support manycore and adding a NVRAM timing simulation module and a PMU.

A. Manycore support

To simulate manycore processors, we configured GEMS so that a processor consists of 8 cores, core-private L1 caches, and a shared L2 last level cache (LLC). All cores in the processor share same LLC. Access to the caches for any task is delayed in accordance with the memory timing model specified.

B. Timing-aware simulation of hybrid main memory with DRAM and NVRAMs

The goal of a timing-based simulation is to get results consisting of not only the execution result itself, but also the latency generated by event handling. GEMS basically provides a timing-based simulation module with the timing parameters of DDR2 SDRAM. We built another timing-based simulation module for the timing parameters of NVRAM and added a request controller to both of memories. The request controller schedules memory access requests according to its scheduling policy. Therefore, the hierarchy for the overall timing-aware simulation module for hybrid memory is shown in Figure 1. We used PRAM [5] and STT-RAM timing parameters as examples of NVRAM.

The overall procedure for the timing-aware memory access timing simulation is shown in Figure 2. If a task accesses memory, a memory access request is generated and issued to the request controller. The issued request is inserted into one of two controllers according to the target memory type of the request. In the request controller, requests are sorted so that requests with the highest priority are issued first. If issued, the state of the issued request is changed to *issued* and waits for the specified time. When the request is done waiting, a response is sent back to the requester.

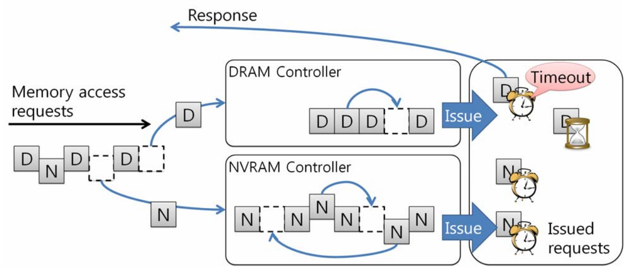

Figure 2. Internal procedure of timing-aware simulation

C. Performance monitoring of tasks

The memory access pattern of a task is one of the most useful hints to predict future memory contention when scheduled simultaneously with other tasks. We added a PMU to the hybrid main memory and the caches in the processor to monitor per-task access patterns. With a number of counter registers, it monitors the occurrence of concerned events such as an LLC miss, a DRAM read, a DRAM write, and a PRAM write. When an administrative task in the hypervisor, guest OS, or any application needs to get values from the PMU, it executes a magic instruction. The magic instruction pauses the current simulation and calls a function that copies values from the PMU to the predetermined memory location within the address space of the caller task. The simulation resumes after the call returns and the caller can access the collected data. By collecting periodically acquired values, the administrative task can efficiently avoid possible memory contention among running tasks.

III. Current status and next steps

We are still in the phase of accelerating simulation speed to prove our idea regarding resource management for the consolidated VMs on a cloud node. Once the simulator is ready to execute fast enough, we plan to investigate performance bottlenecks to exhibit our motivation and to prove the efficiency of our solution regarding resource management.

References

[1] K. H. Park, Y. Park, W. Hwang, and K.-W. Park, "MN-Mate: Resource Management of Manycores with DRAM and Nonvolatile Memories," in *High Performance Computing and Communications (HPCC), 2010 12th IEEE International Conference on*, Sept. 2010, pp. 24 –34.

[2] N. Binkert, R. Dreslinski, L. Hsu, K. Lim, A. Saidi, and S. Reinhardt, "The m5 simulator: Modeling networked systems," *Micro, IEEE*, vol. 26, no. 4, pp. 52 –60, July - Aug. 2006.

[3] P. S. Magnusson, M. Christensson, J. Eskilson, D. Forsgren, G. Hållberg, J. Högberg, F. Larsson, A. Moestedt, and B. Werner, "Simics: A full system simulation platform," *Computer*, vol. 35, pp. 50–58, 2002.

[4] M. M. K. Martin, D. J. Sorin, B. M. Beckmann, M. R. Marty, M. Xu, A. R. Alameldeen, K. E. Moore, M. D. Hill, and D. A. Wood, "Multifacet's general execution-driven multiprocessor simulator (gems) toolset," *SIGARCH Comput. Archit. News*, vol. 33, pp. 92–99, November 2005.

[5] K.-J. L. et al., "A 90nm 1.8v 512mb diode-switch pram with 266mb/s read throughput," *Solid-State Circuits Conference, 2007. ISSCC 2007. Digest of Technical Papers. IEEE International*, pp. 472 –616, feb. 2007.