International Journal of Advance Research in Science and Engineering
Volume No.06, Special Issue No.(01), December 2017
www.ijarse.com
IJARSE
ISSN: 2319-8354

PROPOSED TECHNIQUE TO ANALYZE OPERATING SYSTEM IMPLICATIONS OF FAST, CHEAP, NON-VOLATILE MEMORY

K.Bharath Reddy [1],D.Swaroopa [2]

[1,2]Computer Science and Engineering, Sri Satya Sai
University of Technology and Medical Sciences, Sehore, (India)

ABSTRACT

The goal of a computer system is to run an application work load securely, reliably, efficiently, and fast. A computer's hardware architecture and operating system exist to support this goal, and it would be nice if they cooperated as effectively as possible. Yet there is a growing gap between architectural research and OS research, which seems to be the result of poor communication about what actually matters. In this paper, we discuss this gap and what to do about it. We illustrate the opportunities for closing the gap using examples from some recent operating system research.

The existence of two basic levels of storage (fast/volatile and slow/non-volatile) has been a long-standing premise of most computer systems, influencing the design of OS components, including file systems, virtual memory, scheduling, execution models, and even their APIs. Emerging resistive memory technologies - such as phase-change memory (PCM) and memristors - have the potential to provide large, fast, non-volatile memory systems, changing the assumptions that motivated the design of current operating systems. This study examines the implications of non-volatile memories on a number of OS mechanisms, functions, and properties.

I.INTRODUCTION

It is a decades long belief that the fast memory is volatile and persistent storage is slow. The boom to remember and retrieve, initiated research in the computer field to contribute various storage systems. The earlier forms of memory, magnetic tape and disk laid the basic foundation of storage. The limitation of the Microprocessor and Operating System in the wide range of memory accessibility aroused a need of small primary memory and large secondary memory. Since the primary memory demanded with a fast access it was implemented with RAM modal. This Random Access Memory stores the information about programs which are currently running. RAM is a solid state memory unit. It is faster than writing and reading information from the hard drive because it has no physical components that need to move, just electrical signals. In secondary storage we have to wait for the platters to rotate and the header to move. Only sequential bits can be read without additional operations in between. The volatility and the random accessibility feature of the main memory have many pros and corns. The complex concepts of booting, demand paging and virtual memory happened due to the above features. Today we watch the appearance of many new memory technologies that pledge a noteworthy change in the long way

existence of present memory systems. Memristors (Memory Resistors) offers a capable alternative to conventional volatile memory systems.

The existence of two basic levels of storage (fast/volatile and slow/non-volatile) has been a long-standing premise of most computer systems, influencing the design of OS components, including file systems, virtual memory, scheduling, execution models, and even their APIs. Emerging resistive memory technologies - such as phase-change memory (PCM) and memristors - have the potential to provide large, fast, non-volatile memory systems, changing the assumptions that motivated the design of current operating systems.

Most applications manipulate persistent data, yet traditional systems decouple data manipulation from persistence in a two-level storage model. Programming languages and system software manipulate data in one set of formats in volatile main memory (DRAM) using a load/store interface, while storage systems maintain persistence in another set of formats in non-volatile memories, such as Flash and hard disk drives in traditional systems, using a file system interface. Unfortunately, such an approach suffers from the system performance and energy overheads of locating data, moving data, and translating data between the different formats of these two levels of storage that are accessed via two vastly different interfaces. Yet today, new non-volatile memory (NVM) technologies show the promise of storage capacity and endurance similar to or better than flash at latencies comparable to DRAM, making them prime candidates for providing applications a persistent single-level store with a single load/store interface to access all system data. Our key insight is that in future systems equipped with NVM, the energy consumed executing operating system and file system code to access persistent data in traditional systems becomes an increasingly large contributor to total energy. The goal of this work is to explore the design of a Persistent Memory Manager that coordinates the management of memory and storage under a single hardware unit in a single address space. Our initial simulation-based exploration shows that such a system with a persistent memory can improve energy efficiency and performance by eliminating the instructions and data movement traditionally used to perform I/O operations.

Non-volatile, solid-state memories (NVMs) are poised to revolutionize storage in systems for high-performance computing. Flash memory is already finding applications in large-scale systems, and emerging technologies, such as phase-change memories (PCM), spin-torque transfer memories (STTM), and more exotic technologies (e.g., the memristor and carbon nanotube based memories) will provide orders of magnitude better performance than either conventional disks or flash-based solid-state drives can deliver. What impact these emerging technologies will have on future systems is not yet clear, but they will likely be more disruptive than flash memory has been (and continues to be). The painfully slow performance of non-volatile storage has been an unfortunate reality for system designers for several decades. Systems designers have gone to great lengths to try to mitigate this poor performance: Operating systems employ complex schedulers for IO, and most of the complexity in database management systems is in buffer management and query optimizations designed, in large part, to minimize IO.

II.STATEMENT OF THE PROBLEM

The main purpose of the study is to analyse of operating system and its implications of fast, cheap and non-volatile memory and examine the various problems associated into it.

III.DELIMITATIONS

1. The study is delimited to the operating system.
2. The study is also delimited to use in fast, cheap and non-volatile memory.

IV.LIMITATION

The facts discussed in this study will be based entirely on the responses to the questionnaire and modeling of OS for cheap memory therefore, ascertaining the genuineness of the responses will identify as the limitation of the study.

V.HYPOTHESIS

- There is no significant difference between present and past operating system.
- There is no significant difference between operating system and cheap, fast and non-volatile memory.

VI.DEFINATIONS AND EXPLANATION TERMS

6.1 Operating System

An operating system is system software that manages computer hardware and software resources and provides common services for computer programs. All computer programs, excluding firmware, require an operating system to function.

6.2Memory

Computer memory is any physical device capable of storing information temporarily or permanently. For example, Random Access Memory (RAM), is a volatile memory that stores information on an integrated circuit used by the operating system, software, and hardware.

6.3Non-Volatile Memory

Non-volatile memory is typically used for the task of secondary storage, or long-term persistent storage. The most widely used form of primary storage today is a volatile form of random access memory (RAM), meaning that when the computer is shut down, anything contained in RAM is lost.

VII.SIGNIFICANCE OF THE STUDY

We suspect that operating systems and programming languages will need to let programmers and applications help manage issues of volatility and durability. Only applications are aware of the meaning of certain types of data, and in some cases, only users will be able to indicate the risks of losing (or retaining) different pieces of information. Devising appropriate abstractions and mechanisms for managing this at the OS, language, and application level is an open problem. It consists the followings:-

1. The study would deal in operating system.
2. The study would coordinate the cheap, fast and non-volatile memory.

VIII.REVIEW OF RELATED LITERATURE

Ruia (2015) Virtualization technology is powering today's cloud industry. Virtualization inserts a software layer, the hypervisor, below the Operating System, to manage multiple OS environments simultaneously. Offering numerous benefits such as fault isolation, load balancing, faster server provisioning, etc., virtualization occupies a dominant position, especially in IT infrastructure in datacenters. Memory management is one of the core components of a hypervisor. Current implementations assume the underlying memory to be homogenous and volatile. However, with the emergence of NVRAM in the form of Storage Class Memory (SCM), this assumption remains no longer valid. New motherboard architectures will support several different memory classes each with distinct properties and characteristics. The hypervisor has to recognize, manage, and expose them separately to the different virtual machines. This study focuses on building a separate memory management module for Non-Volatile RAM in Xen hypervisor. We show that it can be efficiently implemented with a few code changes and minimal runtime performance overhead.

DeBrabant (2014) discussed the emerging non-volatile memory (NVM) technologies require us to rethink this dichotomy. Such memory devices are slightly slower than DRAM, but all writes are persistent, even after power loss. We explore two possible use cases of NVM for on-line transaction processing (OLTP) DBMSs. The first is where NVM completely replaces DRAM and the other is where NVM and DRAM coexist in the system. For each case, we compare the performance of a disk-oriented DBMS with a memory-oriented DBMS using two OLTP benchmarks. We also evaluate the performance of different recovery algorithms on these NVM devices.

Cully et al (2014) Strata is a commercial storage system designed around the high performance density of PCIe flash storage. We observe a parallel between the challenges introduced by this emerging flash hardware and the problems that were faced with underutilized server hardware about a decade ago. Borrowing ideas from hardware virtualization, we present a novel storage system design that partitions functionality into an address virtualization layer for high performance network-attached flash, and a hosted environment for implementing scalable protocol implementations. Our system targets the storage of virtual machine images for enterprise environments, and we demonstrate dynamic scale to over a million IO operations per second using NFSv3 in 13u of rack space, including switching.

Shuichi Oikawa (2014) Byte addressable non-volatile (NV) memory, such as STT-RAM, MRAM, and PCM, is the next generation memory that can be used as both main memory and secondary storage. While it can persistently store data without power supply, its access speed is comparable to DRAM. While there have been the active researches on its use for either main memory or secondary storage, these researches were conducted independently. This study presents the integration methods of the main memory and file system management for NV memory, so that it can be used as both main memory and storage. The presented methods use a file system as their basis for the NV memory management; thus, the internal data structures of a file system can have impacts upon the performance of the integration methods. We implemented the proposed methods in the Linux kernel, and performed the evaluation on a system emulator. The evaluation results show that 1) the proposed methods can perform comparably to the existing DRAM memory allocator and significantly better than the page swapping, 2) their performance is affected by the internal data structures of a file system, and 3) the data structures appropriate for traditional hard disk drives do not always work effectively for byte addressable NV memory.

Gao (2013) In this paper, we propose a novel in-memory check pointing system, named Mona, for reducing the checkpoint overhead of hybrid memory systems with NVRAM and DRAM. To minimize the in-memory check point overhead, Mona dynamically writes partial checkpoints from DRAM to NVRAM during application execution.

To reduce the interference of partial check-pointing, Mona utilizes runtime idle periods and leverages a cost model to guide partial check pointing decisions for individual DRAM ranks. We further develop load-balancing mechanisms to balance checkpoint overheads across different DRAM ranks. Simulation results demonstrate the efficiency and effectiveness of Mona in reducing the checkpoint over-head, downtime and restarting time.

Condit (2012) Modern computer systems have been built around the assumption that persistent storage is accessed via a slow, block-based interface. However, new byte-addressable, persistent memory technologies such as phase change memory (PCM) offer fast, fine-grained access to persistent storage. In this paper, we present a file system and a hardware architecture that are designed around the properties of persistent, byte addressable memory. Our file system, BPFS, uses a new technique called short-circuit shadow paging to provide atomic, fine-grained updates to persistent storage. As a result, BPFS provides strong reliability guarantees and offers better performance than traditional file systems, even when both are run on top of byte-addressable, persistent memory. Our hardware architecture enforces atomicity and ordering guarantees required by BPFS while still providing the performance benefits of the L1 and L2 caches.

REFERENCES

[1.] C. Diaconu, C. Freedman, E. Ismert, P.-A. Larson, P. Mittal, R. Stonecipher, N. Verma, and M. Zwilling, Hekaton: SQL Server's Memory-optimized OLTP Engine, In SIGMOD, pages 1243-1254, 2013.

[2.] D. E. Difallah, A. Pavlo, C. Curino, and P. Cudre-Mauroux, Oltp-bench: An extensible test bed for benchmarking relational databases, PVLDB, 7(4):277-288, 2013.

[3.] Adrian M Caulfield, Arup De, Joel Coburn, Todor I Mollov, Rajesh Gupta, and Steven Swanson, Moneta: A High-performance Storage Array Architecture for Nextgeneration, Non-volatile Memories, MICRO, 2010.

[4.] Kemper and T. Neumann, HyPer: A hybrid OLTP&OLAP main memory database system based on virtual memory snapshots, ICDE, pages 195-206, 2011.

[5.] H. Kim, J. Ahn, S. Ryu, J. Choi, and H. Han, In-memory file system for non-volatile memory, In RACS, pages 479-484, 2015.

[6.] N. Malviya, A. Weisberg, S. Madden, and M. Stonebraker, Rethinking main memory OLTP recovery, In ICDE, 2014.

[7.] J. Ousterhout, P. Agrawal, D. Erickson, C. Kozyrakis, J. Leverich, D. Mazieres, S. Mitra, A. Narayanan, G. Parulkar, M. Rosenblum, S. Rumble, E. Stratmann, and R. Stutsman, The Case for RAM Clouds: Scalable High-Performance Storage Entirely in DRAM, SIGOPS Operating Systems Review, 43(4), December 2009.

[8.] Pavlo, C. Curino, and S. Zdonik, Skew-aware automatic database partitioning in shared-nothing, parallel OLTP systems, In SIGMOD, pages 61-72, 2012.

[9.] Mohammed Affan Zidan, Hossam Aly Hassan Fahmy, Muhammad Mustafa Hussain, Khaled Nabil Salama, Memristor-based memory: The sneak paths problem and Solutions" Microelectronics Journal, Volume 44, Issue 2, February 2013, Pages 176-183.

[10.] S. Oikawa, Integrating memory management with a file system on a non-volatile main memory system, SAC, 2015.