Operating System Implications of Fast, Cheap, Non-Volatile Memory

Katelin Bailey, Luis Ceze, Steven D. Gribble, Henry M. Levy
University of Washington

Summary:

Submitted by: Mehnaz Yunus (16CO124), Sharanya Kamath (16CO140)

This paper examines the implications of non-volatile memories on a number of OS mechanisms, functions, and properties. It discusses the ways that cheap byte-addressable NVRAM could substantially affect OS design. The crux of the paper talks about the scenario where there is only NVM as memory in a computer. It discusses the advantages and disadvantages of this concept in different aspects of computing and how the usage of computers or the lives of programmers differ if there is just one memory or no hierarchy in the system.

Hardware alternatives discussed in the paper include cell technology (PCM - Phase Change Memory) and architectural alternatives. One of the key challenges with current PCM technology is its relatively limited write lifetime (order of 10^8 writes). Several architectural alternatives that are relevant to OS design, focusing on the hardware-software interface are discussed. A progression of options is given, from a conventional system with CPU, DRAM, and disk to a system with only CPU and NVRAM.

NVMs are a memory technology that retains its memory state even when no power is supplied. It basically could be called as an intermediate between memory and storage. It is used like DRAMs, providing byte-level storage, at the same time, providing persistence of storage like secondary memory.

Advantages of a system with only CPU and NVRAM are:

- Paging is essentially eliminated as the same device is used as both memory and storage. There is no longer a necessity to move data from a "storage" to "fast" memory.
- Protection systems and how data granularity is implemented in memory. RAMs provide for page-level granularity, while storage provides for a higher level of granularity.
- An application does not have to be installed by the user and then used. An intermediate state, i.e in the middle of execution could be packaged and distributed.
- No concept of "booting" and all processes are always "active".

Disadvantages:

- It has very poor endurance, i.e fails on fewer write cycles. You can't write into the memory as much as DRAM. The memory hardware fails after a certain number of writes.
- When the application experiences a fault. If something breaks a persistent address space becomes a liability. So we'd need to have something software to systematically check-point program execution over time and then recover the program by rolling back.

Questions:

1. In a system with NVRAM, what does a power loss ensue? Does the system resume working from the last state or does a “reboot” occur? How can these alternatives be compared?
2. What does the higher latency of NVRAM indicate for the performance of the system?