Has the Time for EMT Finally Come?

Sam H. Noh
UNIST (Ulsan National Institute of Science and Technology)

Emerging memory technologies (EMT), such as STT-MRAM and PCM, have been hyped in academia and industry for the last decade or so, but more so recently, with the anticipated introduction of 3D XPoint products into the market. As a researcher who has closely followed its development for the last 15 years or so, I ask myself, has the time finally come? If so, will it bring revolutionary changes to our computing environment, or will it just be an evolutionary step forward.

With the advent of EMT, multiple key changes to the computer system are expected. First, from the resource management point of view, the two key changes expected are extremely large memory and the blurring of the boundary between storage and memory.[1] Elaborating more on the latter, a large portion of system development in the modern computer era has been based on the mismatch assumption of slow "peripheral" storage, fast memory, and even faster CPU, the so-called memory hierarchy. This assumption has slowly been crumbling as the gap between slow storage and fast memory narrows with the advent of flash memory. With EMT anticipated to be orders of magnitude faster than flash memory, this assumption may disappear such that traditional views of the computing system may not be adequate. Already, we see reports of traditionally I/O-bound applications transforming into a CPU-bound one as the bulk of the traditional I/O activity is turned into a compute activity.[2] Such change requires a whole new look on the design of the traditional resource manager role of the operating system that we know today. For example, how is the role of a file system different from a memory manager aside from the naming issue?[3] Do traditional data structures adequately serve their role with much faster "storage"?[4,5] Is it, perhaps, time for a revolutionary new operating system?

The other key change is in the nonvolatility of the contents residing in memory. That is, all memory content is now nonvolatile. This is a "double-edged" new feature that will bring about performance boosts never before imagined by allowing something like superfast storage, but that which comes with headaches as one must make sure what is in memory is consistent.[6] While studies in this regard have recently been quite active, I would like to draw our attention to other possible new features that nonvolatility could bring along. Features such as instant boot and instant failure recovery through persistent computing,[7] where a system can come alive from power off in tens of milliseconds or less, may allow us to keep our phones "off" reducing battery consumption to the minimum. Furthermore, if applied to the cloud, such a feature could free us from provisioning for peak loads as systems may be turned on and off "instantly."[8] Realizing these features into our daily computing environment will allow us to be far more energy efficient. Similar new features that exploit the nonvolatility of memory should be more actively sought out.

Digital Object Identifier 10.1109/MM.2018.2885587
Date of current version 21 February 2019.

The key technical challenge in seeking boosted performance along with these added new features is providing transparency at the same time. For this, we must be able to 1) provide the same benefits of EMT to existing legacy software and 2) allow new applications to be developed in the traditional manner, without having to be keen about the notion of nonvolatility. Regarding 1), this has, so far, been shown to be difficult to do.[9] Furthermore, if its execution is not properly coordinated, legacy applications may end up recovering to an inconsistent state from a fault as the memory manager and file system, now both dealing with nonvolatility, recovers on their own. Methods such as uSnap have been proposed to remedy such problems.[10] However, such a solution can only be a single step in the long path ahead as it is still at an early experimental stage. Better and more innovative methodologies need to be developed. Regarding 2), we must learn from the past. In particular, in the late 1980s when parallel computing was at its prime, activities sputtered as a standard programming model was lacking and with programming parallel machines being so difficult and *ad hoc*. This time around, with EMT, it would be best to simply hide nonvolatility from developers and yet, somehow, provide them with the performance gains that are promised with EMT. Research still has ways to go.

A large portion of system development in the modern computer era has been based on the mismatch assumption of slow "peripheral" storage, fast memory, and even faster CPU, the so-called memory hierarchy. This assumption has slowly been crumbling as the gap between slow storage and fast memory narrows with the advent of flash memory.

REFERENCES

1. H. Song and S. H. Noh, "Towards transparent and seamless storage-as-you-go with persistent memory," in *Proc. 10th USENIX Workshop Hot Topics Storage File Syst.*, 2018.
2. F. Hady *et al.*, "Platform storage performance with 3D XPoint technology," *Proc. IEEE*, vol. 39, no. 10, pp. 1822–1833, Sep. 2017.
3. S. Baek *et al.*, "Energy efficient and high performance software architecture for storage class memory," *ACM Trans. Embedded Comput. Syst.*, vol. 12, no. 3, Mar. 2013.
4. S. K. Lee *et al.*, "WORT: Write optimal radix tree for persistent memory storage systems," in *Proc. 15th USENIX Conf. File Storage Technologies*, 2017.
5. I. Oukid *et al.*, "FPTree: A hybrid SCM-DRAM persistent and concurrent B-tree for storage class memory," in *Proc. ACM Int. Conf. Manage. Data*, 2016, pp. 371–386.
6. S. Pelley *et al.*, "Memory persistency," in *Proc. 41st ACM Int. Symp. Comput. Archit.*, 2014, pp. 265–276.
7. H. Song and S. H. Noh, "FSL: Fast system launch through persistent computing with nonvolatile memory," in *Proc. 6th IEEE Non-Volatile Memory Syst. Appl. Symp.*, 2017.
8. I. H. Doh *et al.*, "Towards greener data centers with storage class memory," *Future Gener. Comput. Syst.*, vol. 29, no. 8, pp. 1969–1980, Oct. 2013.
9. V. J. Marathe *et al.*, "Persistent memcached: Bringing legacy code to byte-addressable persistent memory," in *Proc. 9th USENIX Workshop Hot Topics Storage File Syst.*, 2017.
10. J. H. Kim *et al.*, "uSnap: Embracing traditional programming models for persistent memory through OS support," in *Proc. 7th IEEE Non-Volatile Memory Syst. Appl. Symp.*, 2018, pp. 1–6.

Sam H. Noh is a Professor at the School of Electrical and Computer Engineering, UNIST (Ulsan National Institute of Science and Technology), Ulsan, South Korea. Contact him at samhnoh@unist.ac.kr.