New age of Persistent Systems

Vasily A. Sartakov
TU Braunschweig
sartakov@tu-bs.de

January 15, 2015

Non-volatile memory (NV-RAM), i.e., memory that retains stored information without external source of power, entered a new stage of development in recent years. New memory technologies, such phase-change random-access memory [6] or spin-transfer torque random-access memory [7], provide comparable performance to commodity volatile main memory and at the same time offer higher capacity. Thus, NV-RAM could replace DRAM in the not too far future and pave the way for persistent systems.

The conception of persistent systems, e.g., systems where an object can outlive the process that created it [4], is not new, and persistent systems were actively developed [5, 11, 10] in 90th and early of 00th. Technically, most of these systems were based on the idea of checkpointing — a mechanism for saving the memory image of a program (or part of the program) and all its execution state to persistent storage, offering the option for resorting the snapshot at a later time. Now, by using NV-RAM, the main memory is persistent, and programs are persistent per se. That leads to the question: if systems should adapt to this new situation? – and is likely the stating point of new solutions, services and use cases.

Persistence of data and programs stored in memory affect the architecture of systems significantly. For example, removing classical storage from the system architecture changes life cycle of programs. Now, programs are presented in memory all time, there are no process of loading, while loader extracts data and code segments from executable file on storage and places them in memory. Then, conception of files, programs and others will need to be revised.

Meanwhile, use of persistent memory as main memory leads to several technical issues. Persistent memory could retain stored information, but most of active used data is stored in volatile CPU and lost in time of power outage. Moreover, persistent software like drivers interact with devices and they have linked states: at start driver configures devices and after that starts interaction. If the system has power outage after configuration of devices, and after that recovery, then recovered driver and restarted device will have state mismatching.

Scientific community responded on information about developments of persistent memory technologies by various research papers about future architectures of persistent systems. Some of them are devoted developments of file systems for byte-addressable storages [3]. Researchers motivated this work by an idea, that current file-systems were developed for block-addressable storages, and since future storages is based on byte-addressable persistent memory, they will need special file system. In other words, this way of research describes persistent memory as storage. Another approach interpret persistent memory as a memory for executing persistent objects. Various research projects are devoted do discussion of what the level of abstraction should provide persistence of objects. In different research projects, persistent objects could be processes [12], variables and data structures [2], whole operating system [8]. In our project — *Temporality* — we offer persistence at the level of virtual machines provided by NV-RAM aware hypervisor. Thus, we do not interpret persistence memory only as storage for data or as storage for

programs but target a holistic approach.

The Temporality is devoted to the development of an architecture for future cloud systems offering a NV-RAM provided paradigm of persistence. "Persistence", for us, it is not just an idea, that the memory should retain stored data in time of power outage (but we see power outage is typical use-case for persistent systems). Persistence could decrease complexity of a future systems, make it more compact and more efficient. But before that, there are several architectural issues should be resolved.

The project Temporality is based on an idea of persistent hypervisor. In previous work [9], we proposed conceptions of *NV-Hypervisor* which provides persistence of virtual machines by using non-volatile memory. Hypervisor, as lowest layer of software stack, hides hardware features of platform and provides transparent persistence for proprietary and legacy software.

NV-Hypervisor provides ability to save the context of virtual machines in case of a power outage. By integrating the awareness for NV-RAM at the level of system virtualization, NV-Hypervisor resolves issues due to state mismatches of hardware and persistent software. Moreover, we recently extended our original architecture of the NV-Hypervisor by mechanisms of virtualization of persistent memory, and now support multiple persistent VMs that can even be swapped out to secondary storage.

The current prototype of NV-Hypervisor is based on QEMU virtualization platform and NV-DIMMs [1] as persistent memory. We evaluated recovery ability of virtual machines by using memory-heavy service. NV-Hypervisor survived virtual machine with Database from power outage. Moreover, NV-Hypervisor required in factor three less time to recovery performance compare to commodity system.

References

[1] Viking Technology. ArxCis-NV (TM) Non-Volatile Memory Technology. `http://www.vikingmodular.com/products/arxcis/arxcis.html`, 2012.

[2] Coburn, J., Caulfield, A. M., Akel, A., Grupp, L. M., Gupta, R. K., Jhala, R., and Swanson, S. NV-Heaps: making persistent objects fast and safe with next-generation, non-volatile memories. In *ACM SIGARCH Computer Architecture News* (2011), vol. 39, ACM, pp. 105–118.

[3] Condit, J., Nightingale, E. B., Frost, C., Ipek, E., Lee, B., Burger, D., and Coetzee, D. Better I/O through byte-addressable, persistent memory. In *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles* (2009), ACM, pp. 133–146.

[4] Cooper, T., and Wise, M. Critique of orthogonal persistence. In *Object-Orientation in Operating Systems, 1996., Proceedings of the Fifth International Workshop on* (1996), IEEE, pp. 122–126.

[5] Dearle, A., Di Bona, R., Farrow, J., Henskens, F., Lindström, A., Rosenberg, J., and Vaughan, F. Grasshopper: An orthogonally persistent operating system. *Computing Systems 7*, 3 (1994), 289–312.

[6] Lee, B. C., Ipek, E., Mutlu, O., and Burger, D. Architecting phase change memory as a scalable dram alternative. In *ACM SIGARCH Computer Architecture News* (2009), vol. 37, ACM, pp. 2–13.

[7] Li, H., and Chen, Y. An overview of non-volatile memory technology and the implication for tools and architectures. In *Design, Automation & Test in Europe Conference & Exhibition, 2009. DATE'09.* (2009), IEEE, pp. 731–736.

[8] Narayanan, D., and Hodson, O. Whole-system persistence. In *ACM SIGARCH Computer Architecture News* (2012), vol. 40, ACM, pp. 401–410.

[9] Sartakov, V. A., and Kapitza, R. Nv-hypervisor: Hypervisor-based persistence for virtual machines. In *Dependable Systems and Networks (DSN), 2014 44th Annual IEEE/IFIP International Conference on* (2014), IEEE, pp. 654–659.

[10] Shapiro, J. S., Northup, E., Doerrie, M. S., Sridhar, S., Walfield, N. H., and Brinkmann, M. Coyotos microkernel specification. *The EROS Group, LLC, 0.5 edition* (2007).

[11] Shapiro, J. S., Smith, J. M., and Farber, D. J. *EROS: a fast capability system*, vol. 33. ACM, 1999.

[12] Wang, X. L. K. L. X., and Zhou, X. NV-process: A Fault-Tolerance Process Model Based on Non-Volatile Memory.