

# General-Purpose DRAM-Based Storage

John Ousterhout

Department of Computer Science, Stanford University

ouster (at) cs (dot) Stanford (dot) edu

650-721-6325

High-performance storage is a fundamental requirement for Exascale systems. It is unlikely that any single approach will meet all the needs of Exascale systems, so Exascale systems will probably require a portfolio of storage systems with different capabilities. For example, disk-based systems may be able to meet the capacity requirements for Exascale systems, but they cannot provide low-latency random access to small objects, which is essential for many applications. This position paper argues that the Exascale storage portfolio must include DRAM-based storage, where DRAM is used not just as a cache but as the location of record for data. DRAM-based storage offers 1-3 orders of magnitude lower latency than other storage systems, especially for random accesses to small objects; it will enable a class of applications that is not feasible otherwise, and for access-intensive applications it offers the most energy-efficient storage.

DRAM has played a role in storage systems for many decades, but its role for storage-like applications has become more fundamental over the last decade, driven by large-scale Web applications such as search and social networking. For example, all of the major search indexes keep their data entirely in DRAM, and Facebook uses hundreds of terabytes of memory to cache recently-accessed data.

However, current uses of DRAM for storage suffer from two drawbacks. First, they are typically special-purpose systems built at high cost for particular applications. Second, they do not provide durability: it is up to application developers to use other mechanisms (such as existing file systems or database systems) to store persistent copies of information. As a result, it is difficult for developers to use DRAM-based storage in applications; for example, they must manage consistency between information in DRAM and information in persistent storage. It is also difficult to achieve the full performance potential of DRAM: cache misses and synchronous writes to secondary storage can squander much of the performance advantage of DRAM.

The next step in the evolution of DRAM-based storage is the creation of general-purpose systems that make it easy for application developers to harness the full performance potential of DRAM-based storage. Such a system should have the following properties:

- **Large scale:** it must be capable of scaling to 10's of petabytes with 5-10 years. To achieve this, the system must aggregate the memories of thousands (and perhaps tens of thousands) of servers into a single coherent storage system.
- **Low latency:** the performance of the system should be limited only by the speed of memory and the interconnection network, not by any secondary storage device such as a disk or flash. With a high-performance network it should be possible to provide 5 $\mu$ s access times, measured end-to-end for an application running in one machine to request an object from any server in the datacenter and receive a response containing a few hundred bytes. With radical new approaches to networking, the latency can conceivably be reduced to around 2 $\mu$ s, at which point speed-of-light delays begin to dominate.
- **Durable and available:** the system must offer data durability and availability as good as today's disk-based systems (for example, data must not be lost on server crashes or datacenter-wide power failures, and must remain available in spite of individual server crashes). This will require data to be backed up on secondary storage without impacting the performance of reads and writes.
- **Convenient data model:** the system should provide at least a key-value store; ideally it will also offer multiple tables, and higher level features such as secondary indexes and multi-object

transactions. It is an open research question whether such higher-level features can be implemented at the scale and latency required for the system.

In the Stanford RAMCloud project [6, 7] we are building a DRAM-based storage system along the lines described above. RAMCloud is a pure-DRAM storage system: every byte of data is present in DRAM for its entire life. Our current design supports capacities of around a petabyte, and future DRAM improvements should permit even larger capacities. The current system supports read access times of approximately 5 microseconds (but in a small cluster of only 80 nodes). We have developed durability mechanisms that store redundant copies information on disk. With modest power extension capabilities (such as small batteries on each storage server) durability can be achieved with no impact on read performance and only a small impact on write performance (writes take 2-3x as long as reads). Our approach to durability only requires a single copy of information in DRAM (which minimizes system cost) and harnesses the resources of the entire cluster to recover quickly after server crashes. RAMCloud can typically resume normal operation within 1-2 seconds of a storage server crash.

**Related work:** most existing large-scale storage systems, such as GFS [4], Bigtable[2], HDFS[8], and PNUTS [3] are based on disk storage and focus on capacity rather than latency. The only widely used DRAM-based storage system that I know of is memcached [1]. Memcached provides a volatile cache with no promise of either durability or availability, so application developers must manage a separate backing store if they need durability. In recent years there has been a resurgence of interest in main-memory databases such as H-store [5], but these do not yet offer either the scale or latency of RAMCloud.

**Challenges addressed:** DRAM-based storage addresses primarily the memory hierarchy challenge for Exascale OS/R: it will create a new level in the memory hierarchy, intermediate between local memory and disk- or flash-based storage. It will also address several issues related to resilience and scalable management.

**Maturity:** the RAMCloud system already satisfies some of the requirements for a DRAM-based storage system, including latency and durability. However, several challenges remain. We have only been able to test RAMCloud on small-scale clusters (80 nodes), so there are likely to be issues in the scaling the system to thousands or tens of thousands of servers; the current system contains almost no management tools. In addition, the data model is still an open question. From an application standpoint a more powerful data model than a key-value store is desirable; many applications would prefer to have a traditional relational data model. However, there is considerable evidence that relational database systems with ACID semantics cannot scale up to the number of servers required, nor can they scale down to the latency required. Additional research is needed to fully understand the trade-off between the richness of the data model and the scale and latency that it can support.

**Unique to Exascale systems:** the need for DRAM-based storage is not unique to Exascale systems; for example, the RAMCloud project was originally motivated by the needs of large-scale Web applications.

**Novelty:** the most novel aspect of this approach is the idea that DRAM can serve as a long-term storage medium for data; DRAM is the location of record, not just a temporary cache. In addition, the combination of large scale and low latency is a novel aspect of DRAM-based storage systems.

**Applicability to other areas:** DRAM-based storage is relevant to many applications in many areas. Most large-scale applications end up limited by their storage systems (for example, it is widely accepted that rapidly growing Web sites must redesign their storage systems for every 10x increase in site scale). DRAM-based storage increases performance by 100-1000x over traditional systems, which will enable the creation of new applications that could not exist previously.

**Effort:** DRAM-based storage systems can be developed with relatively small teams. For example, the RAMCloud project consists of two faculty and a half-dozen graduate students; with this level of effort we are building a production-quality system that we expect to be usable by real applications.

## References

- [1] *memcached: a Distributed Memory Object Caching System*, January 2011, <http://www.memcached.org/>.
- [2] F. Chang, J. Dean, S. Ghemawat, et al., “Bigtable: A Distributed Storage System for Structured Data”, *ACM Transactions on Computer Systems*, Vol. 26, No. 2, 2008, pp. 4:1 - 4:26.
- [3] B. Cooper, R. Ramakrishnan, U. Srivastava, et. al., “PNUTS: Yahoo!’s Hosted Data Serving Platform,” *VLDB ’08, Proc. VLDB Endowment*, Vol. 1, No. 2, (2008), pp. 1277-1288.
- [4] S. Ghemawat, H. Gobiuff, and S.-T. Leung, “The Google File System,” *Proc. 19th ACM Symposium on Operating Systems Principles*, October 2003, pages 29–43.
- [5] R. Kallman, H. Kimura, J. Natkins, et al., “H-store: a High-Performance, Distributed Main Memory Transaction Processing System,” *VLDB ’08, Proc. VLDB Endowment*, Vol. 1, No. 2, (2008), pp. 1496-1499
- [6] D. Ongaro, S. Rumble, R. Stutsman, J. Ousterhout and M. Rosenblum, “Fast Crash Recovery in RAMCloud,” *Proc. 23rd ACM Symposium on Operating Systems Principles*, October 2011, pp. 29-41.
- [7] J. Ousterhout, P. Agrawal, D. Erickson, C. Kozyrakis, J. Leverich, D., Mazières, S. Mitra, A. Narayanan, D. Ongaro, G. Parulkar, M. Rosenblum, S. Rumble, E. Stratmann, and R. Stutsman, “The Case for RAMCloud,” *Communications of the ACM*, Vol. 54, No. 7, July 2011, pp. 121-130.
- [8] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, “The Hadoop Distributed File System,” *Proc. IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*, 2010, pages 1–10.