Can We Make This Easier?

Ongaro, D., Ousterhout, J.
In search of an understandable consensus algorithm.
Proceedings of the Usenix Annual Technical Conference, 2014, 305–320.
https://www.usenix.org/conference/atc14/technical-sessions/presentation/ongaro

Finally we come to the question, have we built ourselves into unnecessary complexity by taking it on faith that Paxos and its close cousins are the only way to implement consensus? What if there was an algorithm that we could also show to be correct but was designed to be easier for people to comprehend and implement correctly?

Raft is a consensus algorithm written for managing a replicated log but designed with the goal of making the algorithm itself more understandable than Paxos. This is done both by decomposing the problem into pieces that can be implemented and understood independently and by reducing the number of states that are valid for the system to hold.

Consensus is decomposed into issues of leader election, log replication, and safety. Leader election uses randomized election timeouts to reduce the likelihood of two candidates for leader splitting the vote and requiring a new round of elections. It allows candidates for leader to be elected only if they have the most up-to-date logs. This prevents the need for transferring data from follower to leader upon election. If a follower's log does not match the expected state for a new entry, the leader will replay entries from earlier in its log until it reaches a point at which the logs match, thus correcting the follower. This also means that a history of changes is stored in the logs, providing a side value of letting clients read (some) historical entries, should they desire.

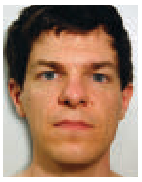
The authors then show that after teaching a set of students both Paxos and Raft, the students were quizzed on their understanding of each and scored meaningfully higher on the Raft quiz. Looking around the current state of consensus systems in industry, we can see this play out in another way: namely, several new consensus systems have been created since 2014 based on Raft, where previously there were very few reliable and successful open-source systems based on Paxos.

Bottlenecks, Single Points of Failure, and Consensus

Developers are often tempted to use a centralized consensus system to serve as the system of record for distributed coordination. Explicit coordination can make certain problems much easier to reason about and correct for; however, that puts the consensus system in the position of the bottleneck or critical point of failure for the other systems that rely on it to make progress. As we can see from these papers, making a centralized consensus system production-ready can come at the cost of adding optimizations and recovery mechanisms that were not dreamed of in the original Paxos literature.

What is the way forward? Arguably, writing systems that do not rely on centralized consensus brokers to operate safely would be the best option, but we are still in the early days of coordination-avoidance research and development. While we wait for more evolution on that front, Raft provides an interesting alternative, an algorithm designed for readability and general understanding. The impact of having an easier algorithm to implement is already being felt, as far more developers are embedding Raft within distributed systems and building specifically tailored Raft-based coordination brokers. Consensus remains a tricky problem—but one that is finally seeing a diversity of approaches to reaching a solution.

Camille Fournier is a writer, speaker, and entrepreneur. Formerly the CTO of Rent the Runway, she serves on the technical oversight committee for the Cloud Native Computing Foundation, as a Project Management Committee member of the Apache ZooKeeper project, and a project overseer of the Dropwizard Web framework.



Implications of NVM on Database Management Systems

By Joy Arulraj and Andrew Pavlo

The advent of non-volatile memory (NVM) will fundamentally change the dichotomy between memory and durable storage in a database management system (DBMS). NVM is a broad class of technologies—including phase-change memory, memristors, and STT-MRAM (spin-transfer torque-magnetoresistive random-access memory)—that provide low-latency reads and writes on the same order of magnitude as DRAM (dynamic random-access memory), but with persistent writes and large storage capacity like an SSD (solid-state drive). Unlike DRAM, writes to NVM are expected to be more expensive than reads. These devices also have limited write endurance, which necessitates fewer writes and wear-leveling to increase their lifetimes.

The first NVM devices released will have the same form factor and block-oriented access as today's SSDs. Thus, today's DBMSs will use this type of NVM as a faster drop-in replacement for their current storage hardware.

By the end of this decade, however, NVM devices will support byte-addressable access akin to DRAM. This will require additional CPU architecture and operating-system support for persistent memory. This also means that existing DBMSs are unable to take full advantage of NVM because their internal architectures are predicated on the assumption that memory is volatile. With NVM, many of the components of legacy DBMSs are unnecessary and will degrade the performance of data-intensive applications.

We have selected three papers that focus on how the emergence of byte-addressable NVM technologies will impact the design of DBMS architectures. The first two present new abstractions for performing durable atomic updates on an NVM-resident database and recovery protocols for an NVM DBMS. The third paper addresses the write-endurance limitations of NVM by introducing a collection of write-limited query-processing algorithms. Thus, this selection contains novel ideas that can help leverage the unique set of attributes of NVM devices for delivering the features required by modern data-management applications. The common theme for these papers is that you cannot just run an existing DBMS on NVM and expect it to leverage its unique set of properties. The only way to achieve that is to come up with novel architectures, protocols, and algorithms that are tailor-made for NVM.

ARIES Redesigned for NVM

Coburn, J., et al.
From ARIES to MARS: Transaction support for next-generation, solid-state drives.
Proceedings of the 24th ACM Symposium on Operating Systems Principles, 2013, 197–212.
http://queue.acm.org/rfp/vol14iss3.html

ARIES is considered the standard for recovery protocols in a transactional DBMS. It has two key goals: first, it provides an interface for supporting scalable ACID (atomicity, consistency, isolation, durability) transactions; second, it maximizes performance on disk-based storage systems. In this paper, the authors focus on how ARIES should be adapted for NVM-based storage.

Since random writes to the disk whenever a transaction updates the database obviously decrease performance, ARIES requires that the DBMS first record a log entry in the write-ahead log (a sequential write) before updating the database itself (a random write). It adopts a *no-force* policy wherein the updates are written to the database lazily after the transaction commits. Such a policy assumes that sequential writes to non-volatile storage are significantly faster than random writes. The authors, however, demonstrate that this is no longer the case with NVM.

The MARS protocol proposes a new hardware-assisted logging primitive that combines multiple writes to arbitrary storage locations into a single atomic operation. By leveraging this primitive, MARS eliminates the need for an ARIES-style undo log and relies on the NVM device to apply the redo log at commit time. We are particularly fond of this paper because it helps in better appreciating the intricacies involved in designing the recovery protocol in a DBMS for guarding against data loss.

Near-Instantaneous Recovery Protocols

Arulraj, J., Pavlo, A., Dulloor, S.R.
Let's talk about storage and recovery methods for non-volatile memory database systems.
Proceedings of the ACM SIGMOD International Conference on Management of Data, 2015, 707–722.
http://queue.acm.org/rfp/vol14iss3.html

This paper takes a different approach to performing durable atomic updates on an NVM-resident database than the previous paper. In ARIES, during recovery the DBMS first loads the most recent snapshot. It then replays the redo log to ensure that all the updates made by committed transactions are recovered. Finally, it uses the undo log to ensure that the changes made by incomplete transactions are not present in the database. This recovery process can take a lot of time, depending on the load on the system and the frequency with which snapshots are taken. Thus, this paper explores whether it is possible to leverage NVM's properties to speed up recovery from system failures.

The authors present a software-based primitive called *non-volatile pointer*. When a pointer points to data residing on NVM, and is itself stored on NVM, then it will remain valid even after the system recovers from a power failure. Using this primitive, the authors design a library of non-volatile data structures that support durable atomic updates. They propose a recovery protocol that, in contrast to MARS, obviates the need for an ARIES-style redo log. This enables the system to skip replaying the redo log, and thereby allows the NVM DBMS to recover the database almost instantaneously.

Both papers propose recovery protocols that target an NVM-only storage hierarchy. The generalization of these protocols to a multitier storage hierarchy with both DRAM and NVM is a hot topic in research today.

Trading Expensive Writes for Cheaper Reads

Viglas, S.D.
Write-limited sorts and joins for persistent memory.
Proceedings of the VLDB Endowment 7, 5 (2014), 413–424.
http://www.vldb.org/pvldb/vol7/p413-viglas.pdf

The third paper focuses on the higher write costs and limited write-endurance problems of NVM. For several decades algorithms have been designed for the random-access machine model where reads and writes have the same cost. The emergence of NVM devices, where writes are more expensive than reads, opens up the design space for new write-limiting algorithms. It will be fascinating to see researchers derive new bounds on the number of writes that different kinds of query-processing algorithms must perform.

Viglas presents a collection of novel query-processing algorithms that minimize I/O by trading off expensive NVM writes for cheaper reads. One such algorithm is the *segment sort*. The basic idea is to use a combination of two sorting algorithms—external merge sort and selection sort—that splits the input into two segments that are then processed using a different algorithm. The selection-sort algorithm uses extra reads, and writes out each element in the input only once at its final location. By using a combination of these two algorithms, the DBMS can optimize both the performance and the number of NVM writes.

Game Changer for DBMS Architectures

NVM is a definite game changer for future DBMS architectures. It will require system designers to rethink many of the core algorithms and techniques developed over the past 40 years. Using these new storage devices in the manner prescribed by these papers will allow DBMSs to achieve better performance than what is possible with today's hardware for write-heavy database applications. This is because these techniques are designed to exploit the low-latency read/writes of NVM to enable a DBMS to store less redundant data and incur fewer writes. Furthermore, we contend that existing in-memory DBMSs are better positioned to use NVM when it is finally available. This is because these systems are already designed for byte-addressable access methods, whereas legacy disk-oriented DBMSs will require laborious and costly overhauls in order to use NVM correctly, as described in these papers. Word is bond.

Joy Arulraj is a Ph.D. candidate at Carnegie Mellon University. He is interested in the design and implementation of next-generation database management systems.

Andy Pavlo is an assistant professor of databaseology in the Department of Computer Science at Carnegie Mellon University, Pittsburgh, PA.