Consensus algorithms

Aug 9, 2015

I'm going to start talking about data replication, and the first and probably most important topic is that of consensus algorithms. I don't want to regurgitate what others have written, so at first this is going to be a reading list. But, at some point, it will contain original bits, and then all of this will turn into a bibliography.

What went before

Here are a handful of the papers I think are important.

- Time, Clocks and the Ordering of Events in a Distributed System. 1978 - brilliant or obvious, or both, this provided a more formal description of partial orders (which is all we get in our universe where relativity applies).

Consensus algorithms

This sub-field was launched with a paper from Lamport titled The Byzantine Generals Problem.

- The Byzantine Generals Problem. 1982 - making progress in the presence of conflicting information.

Wikipedia has a decent article

- Consensus

Note that state machine replication and consensus algorithms go together, and we'll be diving into that as well, since that's the actual fun/hard work.

- State machine replication on Wikipedia.

More reading on consensus:

- Chapter 14: Consensus and Agreement, from the book Distributed Computing: Principles, Algorithms and Systems.
- Distributed Systems, Failures, and Consensus, from a class by Jeff Chase at Duke University on distributed systems.
- The Building Blocks of Consensus. 2008.
- Bounded Cost Algorithms for Multivalued Consensus Using Binary Consensus Instances.
- A Simple Proof of a Simple Consensus Algorithm. 1990?
- Randomized Protocols for Asynchronous Consensus. 2002.
- The correctness proof of Ben-Or's randomized consensus algorithm. 2011.

And Papers We Love has a whole section on distributed systems

papers-we-love/distributed_systems

2PC - two-phase commit

Conceptually, two-phase commit is straightforward; the commit-request phase (or call it the voting phase), and the commit phase, where the coordinator decides whether to commit or not, based on the information gathered in the voting phase.

This is a widely-used technique, although it has problems in the presence of failures. E.g. it assumes there is storage that can be trusted at each node, that no node crashes, and that nodes can communicate with each other. And it's a blocking protocol. Other than that, it's great :)

- Two-phase commit protocol from Wikipedia.
- Consensus Protocols: Two-Phase Commit. 2008.

3PC - three-phase commit

Fixes much of fragility of 2PC, although it's still vulnerable to partitions. E3PC tries to address the problem of partitions.

- Three-phase commit protocol from Wikipedia.
- A Formal Model of Crash Recovery in a Distributed System. 1983, Skeen and Stonebraker, introduced the three-phase commit protocol.
- Consensus Protocols: Three-phase Commit. 2008.
- Crash Recovery in a Distributed Data Storage System. 1979, Butler Lampson and

Howard Sturgis.
- Increasing the Resilience of Distributed and Replicated Data Systems. 1998, Idit Keidar and Danny Dolev, enhanced three-phase commit.

Paxos

This was developed by Leslie Lamport in the late 1980s, circulated informally in the community in the early 1990s, and finally formally published in Transactions on Computer Systems in 1998. It is an improvement over 2PC and 3PC (two-phase commit and three-phase commit).

Paxos has the reputation of being complex, and developers are often warned against implementing it themselves. This seems a little draconian or elitist to me.

These are in publication order, but are starred with a reading order for those wanting to understand Paxos.

- Paxos - Wikipedia. 2015 - recent version.

Papers

- The Part-Time Parliament. 1998 - the first Leslie Lamport paper on the Paxos algorithm.
- Paxos Made Simple. 2001 - another paper from Lamport explaining Paxos, presumably in a fashion easier for some people to understand.
- Cheap Paxos. 2004 - a version of Paxos that is cheaper (less work) when processors aren't constantly failing.
- Fast Paxos. 2005 - paper by Lamport proposing a faster version of Paxos that learns in two message delays instead of three (but error-intolerant).
- Generalized Consensus and Paxos. 2005 - allow parallel execution of non-interfering commands.
- Stoppable Paxos. 2008.
- Vertical Paxos and Primary-Backup Replication. 2009 - an exploration of the idea that valid master/slave replica systems are a variant of Paxos.
- Leaderless Byzantine Paxos. 2011.

Blog articles

- Consensus Protocols: Paxos. 2009 - A fairly clear article from the Paper Trail blog on Paxos.

- Understanding Paxos Part 1, Part 2. 2013 - pair of articles on Paxos.
- Neat Algorithms - Paxos. 2014 - Javascript implementation from Harry Brundage.
- Paxos Made Moderately Complex. 2015 - paper from Robbert van Renesse and Deniz Altinbuken talking about full reconfiguratble multidecree Paxos.

Uses of Paxos

- Google: the Chubby distributed lock service (BigTable uses Chubby).
- Yahoo and ZooKeeper.
- Heroku and Doozerd.
- Amazon Web Services.
- Cassandra and Nutanix.
- libpaxos-cpp

Raft

Raft was developed as an easier-to-understand consensus algorithm, easier that Paxos, that is.

- The Raft Consensus Algorithm. Home page on Github.
- Raft on Wikipedia.
- In Search of an Understandable Consensus Algorithm. 2014, Osterhout paper introducing Raft.
- The Secret Lives of Data
- Raft refloated - do we have consensus. 2015.
- Consensus Protocol. Consul uses Raft for its consensus protocol.
- In Search of an Understandable Consensus Algorithm video of presentation at Usenix 2014.
- ARC: Analysis of Raft Consensus. 2014, Heidi Howard.
- logcabin. LogCabin is a distributed storage system built on Raft; written in C++.

Facebook is using Raft in HydraBase, to replace HBase. CoreOS is using Raft in etcd.

- https://github.com/coreos/etcd

Ark

Introduced for TokuMX and MongoDB.

- Ark: A Real-World Consensus Implementation. 2014.
- Introducing Ark: A Consensus Algorithm for TokuMX and MongoDB.
- Explaining Ark Part 2: How Elections and Failover Currently Work
- Explaining Ark Part 3: Why Data May Be Lost on a Failover
- Explaining Ark Part 4: Fixing Majority Write Concern

Names in consensus algorithms

Major names

- Leslie Lamport. Also wrote TLA+, and LaTeX.
- Jim Gray. Two-tier transaction commit semantics, described ACID semantics.
- Butler Lampson. One of the key names for Xerox PARC and the Alto.
- Barbara Liskov.

Implementors

- Harry Brundage
- Henry Robinson LinkedIn, Paper Trail, henryr on Github - Cloudera, ZooKeeper.

Some Things Are Obvious

Some Things Are Obvious
neurocline@gmail.com

neurocline
neurocline

Answers to some of the world's questions