Research Proposal: Consensus Shootout by Simulation

Benjamin Bengfort and Pete Keleher

Department of Computer Science
University of Maryland
{bengfort,keleher}@cs.umd.edu

April 29, 2016

Abstract

We propose to leverage our simulation framework created for the purpose of studying federated consistency, to review and evaluate multiple popular consensus algorithms. In our review, we will briefly describe each algorithm along with a message flow diagram outlining how many steps are required for consensus decisions. We will compare and contrast the algorithms experimentally in simulation by measuring messages per decision, latency, and correctness. Finally we will visualize the behavior of each algorithm using an interactive JavaScript web application. Through this work we hope not only to compare consensus algorithms but also create reference implementations as well as a teaching tool for use at the University of Maryland.

Introduction

There are a myriad of algorithms designed to coordinate the replication of a linearizable sequence of events, generally referred to as distributed consensus. Many of these algorithms extend the Paxos protocol [9], notorious for its difficulty in implementation [3] and understandability [18], to provide faster or more efficient consensus decisions. Although Paxos has been implemented in real systems [1, 2], many of its descendants are either not publicly available or provide reference implementations that are open to interpretation.

Further complicating matters is the general difficulty reproducing results for systems; not only do the algorithms have to be implemented from their descriptions in papers, so to does the experimental setup have to be replicated. This challenges reviewers in both time and expense; as well as limits pedagogical opportunities. In 2014, Collberg et al. studied the reproducibility of 613 papers from eight ACM conferences and determined that 308 (50%) of papers were not reproducible after searching for GitHub, Sourceforge, or BitBucket repositories and then emailing the authors for more information [4]. Efforts to encourage reproducibility through community exhortation [8] or even by grand challenges [19] have fallen short.

The implementation of the Raft algorithm [18], however, may have changed things for distributed consensus. Raft became very popular after its publication, and in fact there are now at least 76 reference implementations in a variety of programming languages. In terms of reproducibility, Howard et al. used a discrete event simulation to easily create an environment with which to test the experiments designed by Ongaro and Ousterhout [5]. In terms of pedagogy, a number of tools exist for explaining the Raft algorithm using interactive JavaScript visualizations [7, 17].

We propose to extend the work done by Howard et al. by similarly creating a general consensus simulator, which we will make open source and available for consensus experiments. The simulator will contain two parts: an event based simulator that can exercise trace data into a consensus algorithm, measure its performance, and verify its correctness; as well as a visual front-end that will demonstrate how the consensus algorithm responds to reads and writes in the system. We will then utilize the simulator to study several distributed consensus algorithms, verify their results and compare them in a controlled environment.

Proposal

In this section we will describe the two distinct parts of our proposal: the study we propose to undertake, and the simulator we hope to extend for use in our study. The goals for both the study and the simulator are as follows:

1. Survey current distributed consensus algorithms.
2. Measure, validate, and compare performance and correctness for those algorithms.
3. Produce a tool that can be used to reproduce and review distributed consensus.
4. Produce visualizations and products that can be used to teach distributed consensus.

Because work has already been started on the simulator (for a different study), we believe that the primary challenge in this proposal will be the development of any algorithm involved in the study (besides Raft, which has already been implemented).

Study

We propose to survey at least four distributed consensus algorithms, implement them in simulation, and visualize their behavior. This survey would study the differences in distributed consensus algorithms as well as their relative merits and performance gains. In particular, we propose to look at Multi-Paxos [3], ePaxos [14], Raft [18], Warranties [11], and Mencius [12]. If possible, we also propose the study of Fast Paxos [10], Chubby [2], ZooKeeper [6], and Stellar [13] as stretch goals.

In the study, we will provide a general description of distributed consensus, including outlining specific assumptions and environments that we will simulate. We will discuss each protocol as concisely as possible, but will particularly include message flow diagrams of each that show how many messages or steps are required for consensus. We will conduct experiments using a simulated network environment and compare and contrast the protocols. Finally, we will use the simulator's front-end JavaScript visualization to demonstrate how each protocol works.

For the experimental study, we will simulate a fixed set of access events (reads/writes) to the cluster. We will measure the total time required to complete the simulation, the number of messages required, the commit and visibility latency, the number of timeouts, and the amount of downtime. We will also verify correctness, in that all replicated nodes maintain the same, consistent ordering at the conclusion of the simulation.

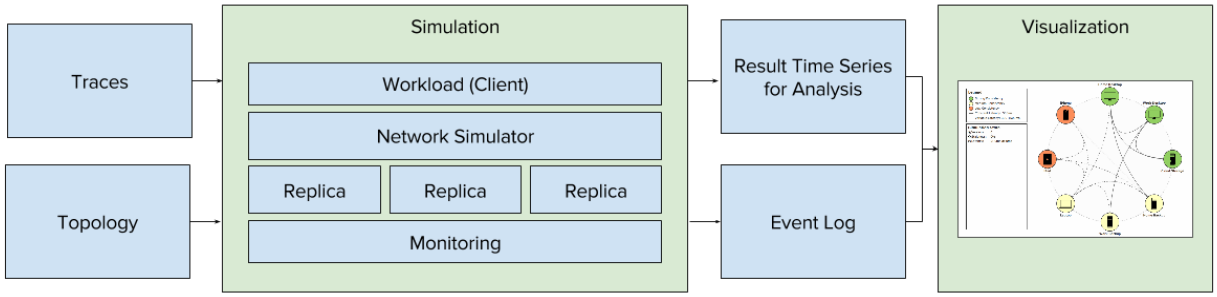

Figure 1: Component architecture of the distributed consensus simulator that has already been developed.

Simulator

The simulator for this study has already been developed to study federated consensus, and although some work still needs to be done (for example the JavaScript visualization is only 50% complete) it is in a very good place to undertake this effort. The simulator uses discrete event simulation implemented by the SimPy framework in Python [15, 16], however real time simulation is also possible.

As shown in Figure 1, the simulator accepts as input a *topology*, specified as a JSON file that determines what nodes are in the system, what consistency levels they're at, and the network latencies between their connections. Additionally the simulator can accept *traces*, a TSV file of fixed event accesses made to each replica in the simulator. The output is a results file formatted as JSON data that includes time series measurements made by the simulation as well as meta data about the configuration of the nodes. Optionally the simulator can also output an event trace which is used to demonstrate behavior through an animation.

Extending the current simulation to include new replica types that implement the various consensus protocol would require subclassing the `Replica` object and writing event handlers for the various message types and RPCs that need to be implemented.

Conclusion

We have proposed a study of multiple distributed consensus protocols that utilizes a currently available simulator to compare and contrast performance. We believe that this survey will be valuable to the systems community both as reference implementations and also as a mechanism to reproduce results. By making the simulator open, we hope to encourage community participation, thereby creating a robust analysis mechanism that is currently unavailable in the field. Moreover, we believe that both the simulator, the survey, and the visualizations will serve as invaluable teaching tools for distributed systems courses.

References

[1] William J. Bolosky, Dexter Bradshaw, Randolph B. Haagens, Norbert P. Kusters, and Peng Li. Paxos replicated state machines as the basis of a high-performance data store. In *Symposium*

on Networked Systems Design and Implementation (NSDI), pages 141–154, 2011.

[2] Mike Burrows. The Chubby lock service for loosely-coupled distributed systems. In *Proceedings of the 7th symposium on Operating systems design and implementation*, pages 335–350. USENIX Association, 2006.

[3] Tushar D. Chandra, Robert Griesemer, and Joshua Redstone. Paxos made live: an engineering perspective. In *Proceedings of the twenty-sixth annual ACM symposium on Principles of distributed computing*, pages 398–407. ACM, 2007.

[4] Christian Collberg, Todd Proebsting, Gina Moraila, Akash Shankaran, Zuoming Shi, and Alex M. Warren. Measuring reproducibility in computer systems research. *Department of Computer Science, University of Arizona, Tech. Rep*, 2014.

[5] Heidi Howard, Malte Schwarzkopf, Anil Madhavapeddy, and Jon Crowcroft. Raft refloated: do we have consensus? *ACM SIGOPS Operating Systems Review*, 49(1):12–21, 2015.

[6] Patrick Hunt, Mahadev Konar, Flavio Paiva Junqueira, and Benjamin Reed. ZooKeeper: Wait-free Coordination for Internet-scale Systems. In *USENIX Annual Technical Conference*, volume 8, page 9, 2010.

[7] Ben Johnson. The Secret Lives of Data. http://thesecretlivesofdata.com/, October 2014.

[8] Jelena Kovacevic. How to encourage and publish reproducible research. In *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, volume 4, pages IV–1273. IEEE, 2007.

[9] Leslie Lamport. Paxos made simple. *ACM Sigact News*, 32(4):18–25, 2001.

[10] Leslie Lamport. Fast paxos. *Distributed Computing*, 19(2):79–103, 2006.

[11] Jed Liu, Tom Magrino, Owen Arden, Michael D. George, and Andrew C. Myers. Warranties for faster strong consistency. *Proc. of the 11th USENIX NSDI*, 2014.

[12] Yanhua Mao, Flavio Paiva Junqueira, and Keith Marzullo. Mencius: building efficient replicated state machines for WANs. In *OSDI*, volume 8, pages 369–384, 2008.

[13] David Mazieres. The stellar consensus protocol: A federated model for internet-level consensus. *Draft, Stellar Development Foundation, 15th May, available at: https://www. stellar. org/papers/stellarconsensus-protocol. pdf (accessed 23rd May, 2015)*, 2015.

[14] Iulian Moraru, David G. Andersen, and Michael Kaminsky. There is more consensus in egalitarian parliaments. In *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, pages 358–372. ACM, 2013.

[15] Klaus Muller and Tony Vignaux. Simpy: Simulating Systems in Python, February 2003.

[16] Klaus G. Müller, Tony Vignaux, Ontje Lünsdorf, Stefan Scherfke, and Karen Turner. SimPy, June 2015.

[17] Diego Ongaro. Raft Scope. https://raft.github.io/raftscope-replay/, August 2015.

[18] Diego Ongaro and John Ousterhout. In search of an understandable consensus algorithm. In *2014 USENIX Annual Technical Conference (USENIX ATC 14)*, pages 305–319, 2014.

[19] Pieter Van Gorp and Steffen Mazanek. SHARE: a web portal for creating and sharing executable research papers. In *Procedia Computer Science*, volume 4, pages 589–597, 2011.