Key Characteristics and Evaluation of Paxos

Sverre Coucheron
Computer science
University of Tromsø
Tromsø, Norway
Email: sverre.coucheron@uit.no

***Abstract*—Consensus algorithms are a known problem in computer science. When talking about them, Paxos is often the first one mentioned. This paper will analyze and evaluate the key characteristics of Paxos with a different number of concurrent clients. Fault tolerance is already proven byLeslie Lamport [3], so the focus will be on throughput of accepted requests.**

I. Introduction

Paxos is a fairly simple protocol, but it has a reputation on itself to be hard to understand even for a seasoned professional. It is an algorithm created by Leslie Lamport, and is considered by many to be the . This paper will discuss the performance of an implementation done by Robbert Van Renesse and Deniz Altinbken [1]

II. Evaluation

This section will discuss and evaluate the performance of the given Paxos implementation. All testing has been done on the hardware in the table below.

CPU	i5-2500k
RAM (GB)	8 (1600Mhz)
Network	Local (Realtek 8812AU)
Number of runs	Five per number of concurrent client

When testing the performance and throughput of the Paxos network it is done by using 5 acceptors, from 0 to 20 concurrent clients with each client sending 100 requests to the Paxos cluster. Each test is done five times to get more correct statistics, meanwhile the lowest and highest values are removed incase of extreme values.

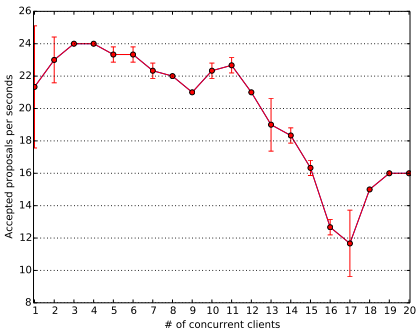

Fig. 1. Simulation results for the network using a python implementation [2]

As one can see from the graph the number of accepted requests done to the network goes down the higher the amount of concurrent clients there are in the network. Since each client does a hundred requests to the cluster, the amount of total messages to the replica will be a lot higher when adding another client. This means that a lot of the request will not be accepted on the first try and have to be resent at a later time. Since paxos guarantees that at some point the request will go through, the wait time will be longer the more you spam the network. The chance of a client getting a request in is slimmer. Together with this the rest of the requests has tries at the same time, which means that the whole network has to halt for a while to discard of these and send a reply so they can be brought back at a later stage.

Even though some safety measures are taken to avoid extreme values in the calculations, when having 17 concurrent clients one can see in the graph that the standardeviation is around 16%. This may have been a hickup in the network traffic where three values spent extreme times to get accepted. Since only the lowest and highest values are removed, there is a possibility that the next highest or lowest value was an extreme and therefore made a huge difference the result.

It is also important to mention that the results for each level are written to disk, which may have lead to a decrease in the performance.

III. Conclusion

As the graph shows the Paxos algorithms throughput in accepted messages per second goes down with more clients. This was expected since more traffic usually means a longer queue to get chosen. It is also worth mentioning that since there was only done five runs per concurrency-level, this may lead to spikes and anomalies in the results, so for a more thorough evaluation it may be wise to run with a couple of hundred test per concurrency-level. One should also try to avoid to write to disk, since this consumes a lot of time.

References

[1] Robert van Renesse, *Paxos Made Moderately Complex*, ACM Computing Surveys (CSUR), 47(3):42, 2015.
[2] http://paxos.systems/index.html
[3] Leslie Lamport et al. *Paxos made simple*, ACM Sigact News, 32(4):18-25, 2001