CSC 466 Project Proposal: Distributed Consensus Under Multicasting

Blake Smith: v00850827, Feb 7th 2022

The Problem

The scale of many modern systems has grown past the capabilities of a single computer. Often-times the amount of data being stored, accessed, and analysed exceeds the maximum potential storage capacity of a single machine. It is also common that there are vast numbers of concurrent users in modern systems, that cannot be adequately served by a single computer processor. As such, large-scale systems are distributed across many computers. The use of distributed systems has many benefits such as availability, data redundancy, geographical distribution, and (horizontal) scalability. However, running systems across multiple computers introduces significant challenges. Core to these challenges is the issue of **consensus**. Computers in a distributed system must agree on the current state of the system as a whole. Each machine has it's own local state which must be reconciled with the state of all other participants in the distributed system. If such agreement is not reached then a system is bound to show unexpected behaviour, critically affecting the effectiveness of the system.

There are proven algorithms which solve the consensus problem such as RAFT and Paxos. These algorithms work by keeping a synchronised log of system events, where an event does not officially occur (is not responded to) until it has been recognised by all nodes in the system. Consensus algorithms are successful in achieving strong consistency in distributed system, allowing distributed systems to behave like classical systems from the user's perspective. Necessarily, distributed consensus comes at a significant cost as it requires many messages to be passed between the system's participants. In the current internet, this requires many open TCP connections which presents a serious overhead. Many modern distributed systems, such as Kafka, forgo the strong consistency offered by RAFT and Paxos in favour of eventual consistency. In eventually consistent distributed systems it is guaranteed that all participants will eventually be updated about all events across the system. This is a weaker guarantee than is provided by full distributed consensus, but comes at a reduced cost in comparison.

New internet proposals, such as MobilityFirst and Internet Indirection Infrastructure, offer new capabilities of Any-casting and Multi-casting. These new internet features have the potential to drastically improve the real-time performance of distributed consensus algorithms.

Importance

In modern industry, it is common to elect for eventual consistency instead of strong consistency. The real-world overhead of full consensus reduces system performance beyond what is acceptable for many applications. Programmers utilising distributed databases and other systems must be aware when working with eventually consistent systems. Failing to recognise their limitations may lead to byzantine issues which are difficult to reproduce, and thus nearly impossible to debug. In contrast, strongly consistent systems more successfully mimic non-distributed systems from the programmers perspective. This leads to fewer expensive, byzantine, bugs and thus more reliable systems.

If a new internet architecture can reduce the cost, in time, of full distributed consensus it may become feasible to deploy strong consistency across a wider variety of services. A move to high performance strongly consistent distributed systems

could have a massive effect in the reliability of commonly used internet services, and thus an improvement in user experience as a whole across the internet.

This Work

In this project, I will evaluate the potential performance of distributed consensus algorithms under a new internet architecture which includes multi-casting as a core feature. Further I will investigate how distributed consensus algorithms might be changed when multi-casting, and other new internet features, are available.

Timeline

- Feb 7th - Feb 14th
    - Write clear explanation of RAFT's complexity in terms of network costs (current internet)
    - Research further into the new internet proposals discussed in class and multi-casting
        - How multi-casting can be implemented in a simulated network
        - Select a simulation solution to evaluate performance of RAFT with multi-casting
- Feb 14th - Feb 28th
    - Simulate RAFT algorithm (in current internet, without multi-casting)
    - Establish baseline performance metrics
    - Update project report with explanation of the simulation
        - Set up environment for simulation with multi-casting
- March 1st - March 30th
    - Prepare midterm presentation
    - Simulate RAFT with multi-casting
    - Evaluate performance against standard RAFT algorithm
    - Include results in project report
- April 1st - End of term
    - Continue project report
    - Prepare final presentation

Deliverables

Task	Tentative Due Date
Simulation environment ready	Feb 18th
Implemented RAFT Simulation (without multi-casting)	Feb 28th
Midterm Presentation	TBD
Implemented RAFT Simulation (with multi-casting)	March 20th
RAFT performance (with multi-casting) evaluated	March 30th
Final Presentation	TBD
Final Report Submission	TBD

References

- https://www.inf.ed.ac.uk/teaching/courses/ds/slides1516/agreement.pdf
- https://raft.github.io/raft.pdf
- https://lamport.azurewebsites.net/pubs/paxos-simple.pdf
- ripublication.com/ijcir17/ijcirv13n10_17.pdf
- Efficient multicasting agreement protocol - ScienceDirect
- ir.lib.cyut.edu.tw:8080/bitstream/310901800/6455/1/Generalized+Agreement+Underlying+a+Multicasting+Environment.pdf