Distributed Systems

CSE 5306-002 PROJECT REPORT

Multi-Paxos protocol

Submitted by

Shubash Muniyappa (1001915563)

Sai Rohit Kalyan Gandham (1002070724)

Instructor: Jia Rao

Academic Integrity

I have neither given nor received unauthorized assistance on this work

Signed: [illegible] Date: 12/07/2022

Signed [illegible] Date: 12/07/2022

<u>Assignment 1</u>

Introduction

Let's say you have a group of computers, and you want them to all concur on something. This is what consensus, which is just agreement, is all about.

In the design of distributed systems, consensus is a common topic. We need it for a number of reasons, including mutual exclusion to decide who has access to a resource, elections to decide who will rule, and common event ordering among a group of computers (e.g., what action to take next, or state machine replication).

Maybe the most typical application of consensus is replication. Using the Paxos algorithm, a dispersed group of computers that communicate via an asynchronous network can come to an agreement. We have consensus when the majority of the systems running Paxos agree on one of the offered values, which can be suggested by one or more clients. Since Paxos was the first consensus algorithm to be conclusively shown to be valid, it has received widespread use and is renowned in the field of computer science.

Paxos merely chooses a single value from one or more values that are suggested to it and announces that value to everyone. The Paxos protocol is executed, and a single proposed value is chosen. If you need to generate a duplicated log (for a replicated state machine, for instance) then, Paxos should run repeatedly which is known as multi-paxos.

A distributed system can reach consensus using the multi-Paxos protocol, which is a distributed consensus protocol. It is a Paxos protocol variation made to function in an asynchronous setting.

1. **How does the multi-paxos protocol ensure multiple nodes to agree on a consistent ordering of a sequence of values and how is it different from running the basic Paxos protocol for multiple rounds?**

   A quorum-based commit procedure is employed by the multi-Paxos protocol to guarantee that a majority of nodes concur on the sequencing of a set of values. This quorum-based commit procedure makes sure that in order to commit a value to the sequence, a node must first deliver a prepared message to the system's majority of nodes. A node may send a commit message to all other nodes in the system if the majority of responding nodes provide a promising response.

   Assume, for instance, that a system has three nodes: A, B, and C. To add a value to the sequence, Node A desires to commit. A prepared message is sent to nodes B and C. Node A may transmit a commit message to every node in the system, including node B, if node B replies with a promising reply.

   The main theoretical distinction is that while Paxos utilizes just one modified instance, Multi-Paxos uses an array of Paxos instances. And there is only one prepare phase and one commit phase for each value in the multi-Paxos protocol. The prepare phase of the multi-Paxos protocol uses a quorum-based commit mechanism, which requires a majority of promising responses before a node can send a commit message. In contrast to paxos, the commit phase of the multi-Paxos protocol sends a commit message to every node in the system, not simply to the majority nodes. Multi-Paxos, will allows each Paxos instance to commit its values out of order, whereas Paxos commits values in the order in which they are proposed or processed (though concurrent processing is still possible).

2. What are the performance and scalability issues of the multi-paxos protocol?

A distributed state machine must receive a constant stream of agreed-upon values acting as instructions in a typical Paxos deployment. A large amount of overhead would result if each command was the result of a single instance of the Basic Paxos protocol.

Phase 1 is not required if the leader is largely steady. Therefore, phase 1 may be skipped in subsequent instances of the protocol with the same leader.

In order to do this, each value that is increased in each round by the same Leader is given, along with the round number I. The failure-free message delay (proposal to learning) is decreased by multi-Paxos.

Even though multi-paxos protocol is prominent, there are few problems with performance and scalability.

> The multi-Paxos protocol may need up to n rounds of Paxos for agreement if there are n nodes in the system. This is due to the possibility that each node has to communicate with each other node before committing. Each node must keep a track of all the values that all other nodes propose which results in linear explosion of logs size depending on how many Paxos rounds are necessary to reach agreement. This also results in communication cost to increase linearly as the systems nodes increases.

Assignment 2

2 Phase commit

The Two-phase commit protocol (2PC) is a type of atomic commitment protocol (ACP). A distributed method coordinates whether to commit or abort (roll back) a distributed atomic transaction between all the processes involved. This protocol, a specialized form of consensus protocol, is extensively used since it accomplishes its objective even in many instances of transient system failure (including either process, network node, communication, etc. problems).

Implementation

The 2-phase commit as the name implies have 2 phases a prepare phase and the commit phase.

Prepare phase:

1. After completing its transaction locally, each nodes sends a vote a message. Once the coordinator has received this message from all of the slaves, it sends a "PREPARE" message to all of the slaves.
2. The coordinator receives a "Yes" message from each slave in response to the "PREPARE" message.
3. If a slave responds with a "No" message or does not respond at all, the coordinator sends a global "ABORT" message to all the other slaves. Until all of the slaves agree that the transaction has been abandoned, the coordinator does not consider it to be complete.

Commit phase:

1. When the transaction coordinator has received the "READY" message from all of the slaves, the "COMMIT" message—which contains the information about the transaction that must be saved in the databases—is issued to each slave.
2. A "DONE" acknowledgment message is sent to the coordinator once each slave has finished
3. The coordinator considers the entire transaction to be complete if it receives a "DONE" message from each slave.

Challenges

Major challenge was to understand 2 PC commit phases and scenarios theoretically.

Observations

We observed how the 2-pc protocol helps the system in different scenarios of crashes.

Contributions

- Coordinator in assignment 2 is implemented by Shubash.
- Nodes in assignment 2 is implemented by Rohit.
- Report for assignment 2 is created by Rohit.
- Introduction and question 1 in Assignment 1 created by Shubash.
- Question 2 in assignment 1 is answered by Rohit.

References

https://www.researchgate.net/publication/315526399_Low-Overhead_Paxos_Replication

https://en.wikipedia.org/wiki/Paxos_(computer_science)

https://lamport.azurewebsites.net/pubs/paxos-simple.pdf

https://www.researchgate.net/publication/225201460_Revisiting_the_PAXOS_algorithm

https://www.researchgate.net/publication/266659113_Bringing_Paxos_Consensus_in_Multi-agent_Systems

https://people.cs.rutgers.edu/~pxk/417/notes/paxos.html

https://www.researchgate.net/publication/330919099_An_Optimized_Multi-Paxos_Protocol_with_Centralized_Failover_Mechanism_for_Cloud_Storage_Applications_Methods_and_Protocols

https://www.researchgate.net/figure/Normal-operation-of-Multi-Paxos-in-a-client-server-system-with-3-server-replicas-and-a_fig2_271910927

https://dl.acm.org/doi/10.1145/3127479.3128609

https://cs.uwaterloo.ca/~bernard/Canopus.pdf

http://charap.co/scalable-but-wasteful-or-why-fast-replication-protocols-are-actually-slow/

https://www.researchgate.net/publication/236693891_Multi-Ring_Paxos