Decentralized state machine using Nakamoto (probabilistic) consensus

Research Project 2022

Nicolas COURAUD (nicolas.couraud@etu.emse.fr)
École Nationale Supérieure des Mines de Saint-Étienne

October 31, 2022

Abstract

As defined by Schneider [Sch90], the state machine approach is a general method for implementing a fault-tolerant service by replicating servers and coordinating client interactions with server replicas.

In this article, we look at state machines as a way to hold in multiple servers copies of the same record. That record, in our implementation, is a standard key-value data structure (a Go map).

We show how one can build a decentralized state machine using Nakamoto Consensus (in our implementation we used the Snowball consensus algorithm), and we study the advantages and drawbacks of such an approach, compared to classical consensus algorithms like Raft [OO13].

The implementation of the project is available at github.com/Nicolascrd/distributed-state-machine

Contents

1 Traditional state machine replication

1.1 Introduction

Introduction Introduction Introduction Introduction Introduction Introduction Introduction

1.2 Byzantine fault-tolerance with classical consensus

Byzantine Byzantine Byzantine Byzantine Byzantine Byzantine Byzantine Byzantine

2 Nakamoto (probabilistic) state machine replication

2.1 The Nakamoto Consensus

The Naka consensus The Naka consensus The Naka consensus The Naka consensus The Naka consensus The Naka consensus The Naka consensus

2.2 My Nakamoto state machine implementation

Nakamoto state machine Nakamoto state machine Nakamoto state machine Nakamoto state machine Nakamoto state machine Nakamoto state machine Nakamoto state machine Nakamoto state machine Nakamoto state machine Nakamoto state machine Nakamoto state machine Nakamoto state machine Nakamoto state machine Nakamoto state machine Nakamoto state machine

3 Performance Evaluation

3.1 Probability of not reaching consensus in Nakamoto Consensus

Markov Markov Markov Markov Markov Markov Markov Markov Markov Markov

3.2 Number of requests required in regular functioning

Number of requests Number of requests Number of requests Number of requests Number of requests Number of requests Number of requests Number of requests Number of requests Number of requests Number of requests Number of requests Number of requests

3.3 In case of a failure

Failure ? Failure ? Failure ? Failure ? Failure ? Failure ? Failure ? Failure ? Failure ? Failure ?

4 Conclusion

Conclusion Conclusion Conclusion Conclusion Conclusion Conclusion Conclusion Conclusion Conclusion

References

[OO13] D Ongaro and J Ousterhout. In search of an understandable consensus algorithm. 2013. `https://raft.github.io/raft.pdf`.

[Sch90] F B. Schneider. Implementing fault-tolerant services using the state machine approach : a tutorial. 1990. https://www.cs.cornell.edu/fbs/publications/SMSurvey.pdf.