Verification and Visualization of a Consensus Algorithm using TLA$^+$

Afonso das Neves Fernandes

Supervisor: Rolando da Silva Martins

Computer Science Department
Faculty of Science, University of Porto

Motivation

- Initial thesis theme: "Byzantine Distributed Storage".
- Research phase: Distributed systems and Ceph.
  - Consensus algorithm based on Paxos.
  - Mention of some deviation from the original algorithm.
  - Main documentation of the algorithm in source code comments.
- Research phase: Distributed consensus.
  - The TLA$^+$ Video Course.
- "Formal Verification of the Ceph Consensus Algorithm".

Objectives

- Better documentation of how the algorithm worked.
- Safety verification of the consensus algorithm.
- New way of testing other versions of the algorithm.

Table of Contents

1. Background
2. Ceph Consensus Algorithm Specification
3. Visualization Tool
4. Results
5. Future Work

Table of Contents

1. Background
2. Ceph Consensus Algorithm Specification
3. Visualization Tool
4. Results
5. Future Work

Consensus in Distributed Systems

- The problem of multiple nodes agreeing on a sequence of values.
- Single-value consensus and general consensus.

Important properties of single-value consensus

- The chosen value must be a value from the set of proposed values.
- Only a single value must be chosen.
- A correct node will only learn that a value was chosen if it actually has been chosen.

Paxos

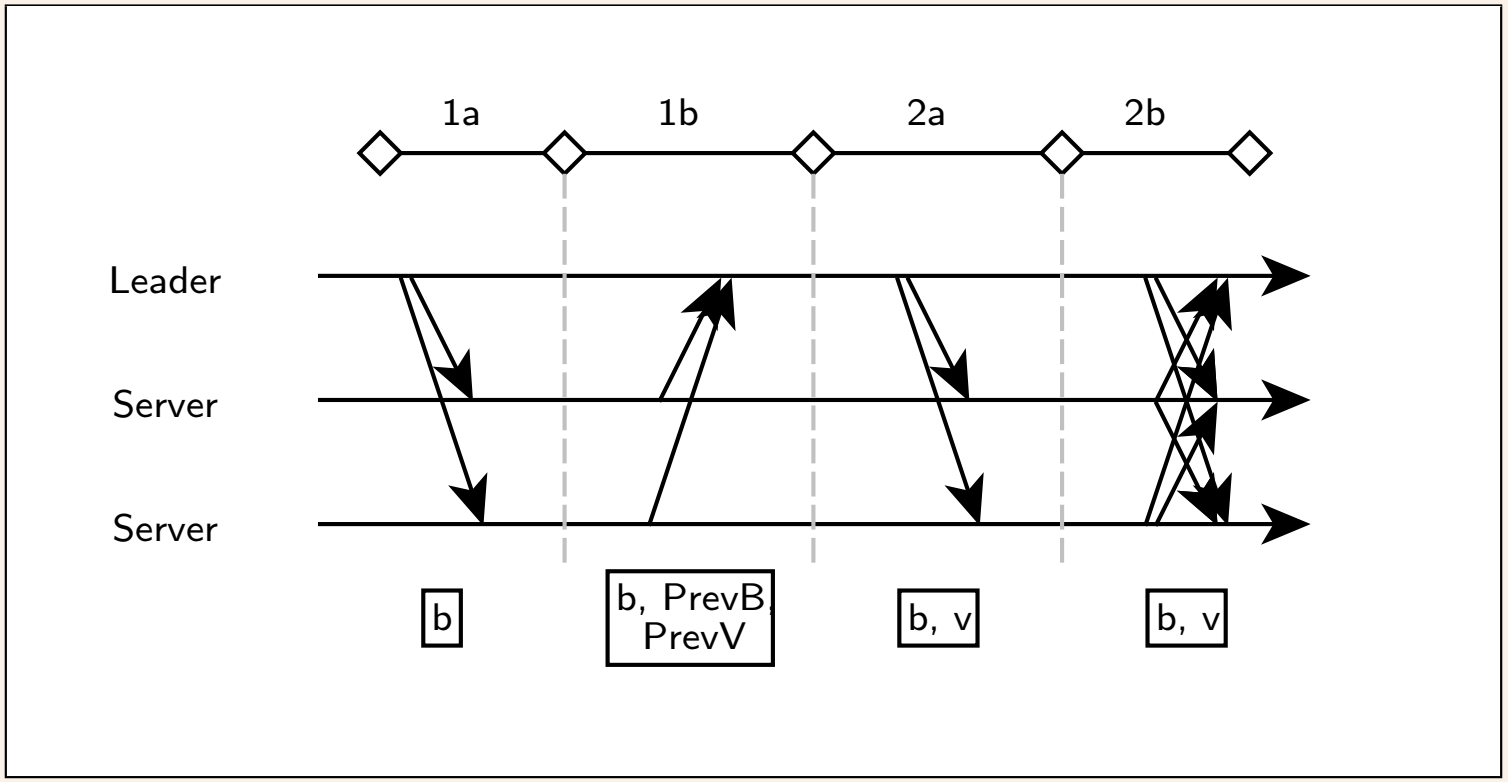

Figure: Paxos consensus message diagram. Each server is both an acceptor and a learner. One of the servers, the Leader, is also a proposer. In the diagram, b is a ballot number, v a value, $PrevB$ a previous voted ballot number and $prevV$ a previous voted value.

Multi-Paxos

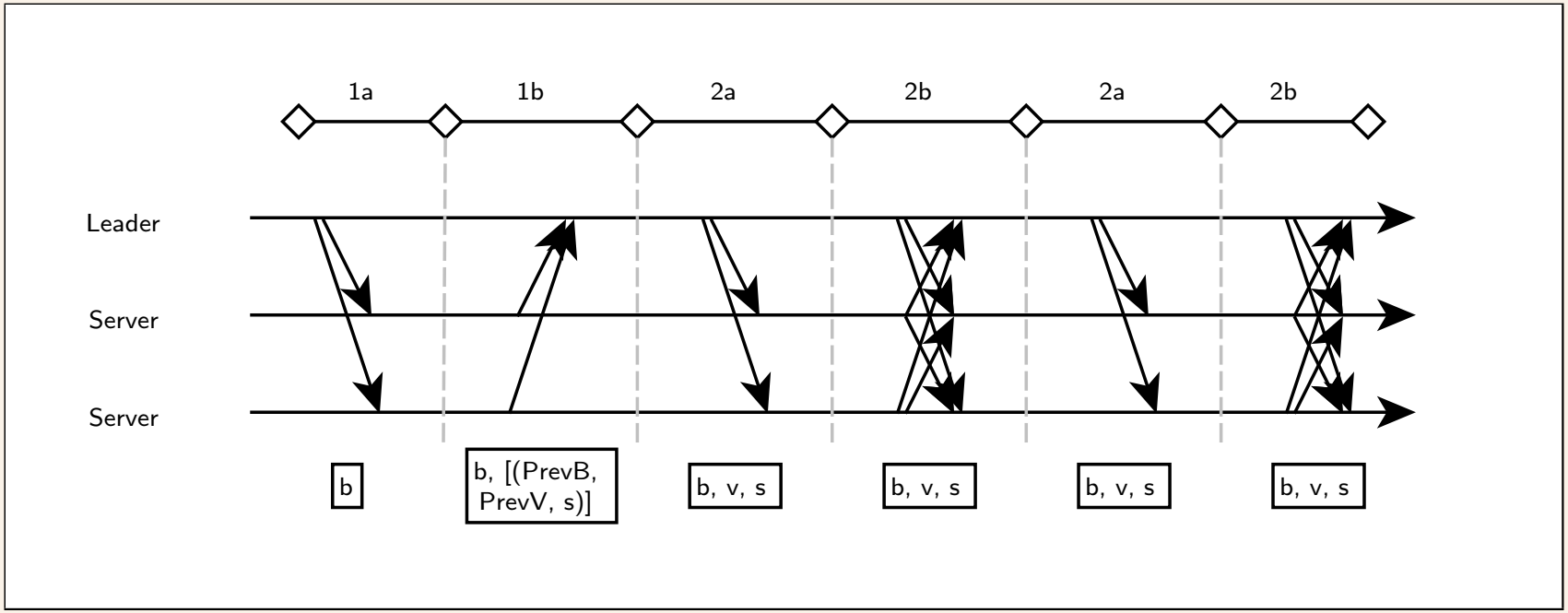

Figure: Multi-Paxos consensus message diagram. Each server is both an acceptor and a learner. One of the servers, the Leader, is also a proposer. In the diagram, b is a ballot number, v a value, s a slot, $PrevB$ a previous voted ballot number and $prevV$ a previous voted value.

Table of Contents

1. Background
2. Ceph Consensus Algorithm Specification
3. Visualization Tool
4. Results
5. Future Work

Introduction to Ceph

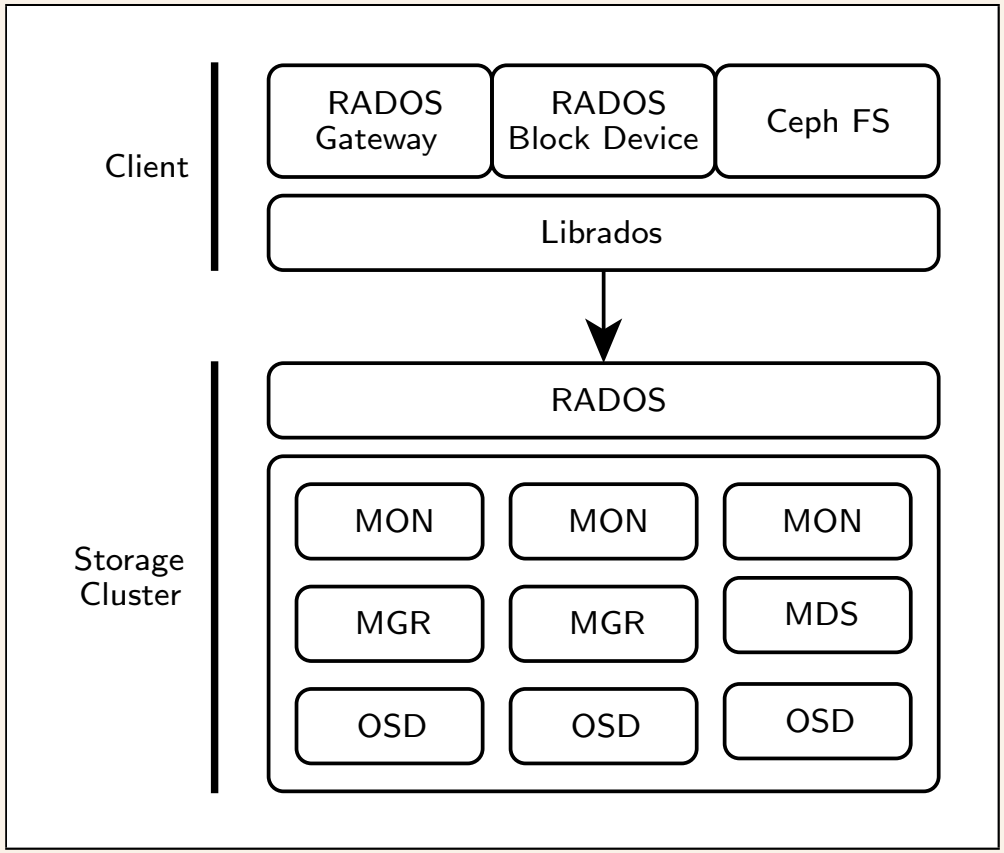

Figure: Diagram of the Ceph architecture. MON stands for monitor servers, MGR stands for manager servers, MDS stands for metadata servers, and OSD stands for object storage devices.

Introduction to Ceph

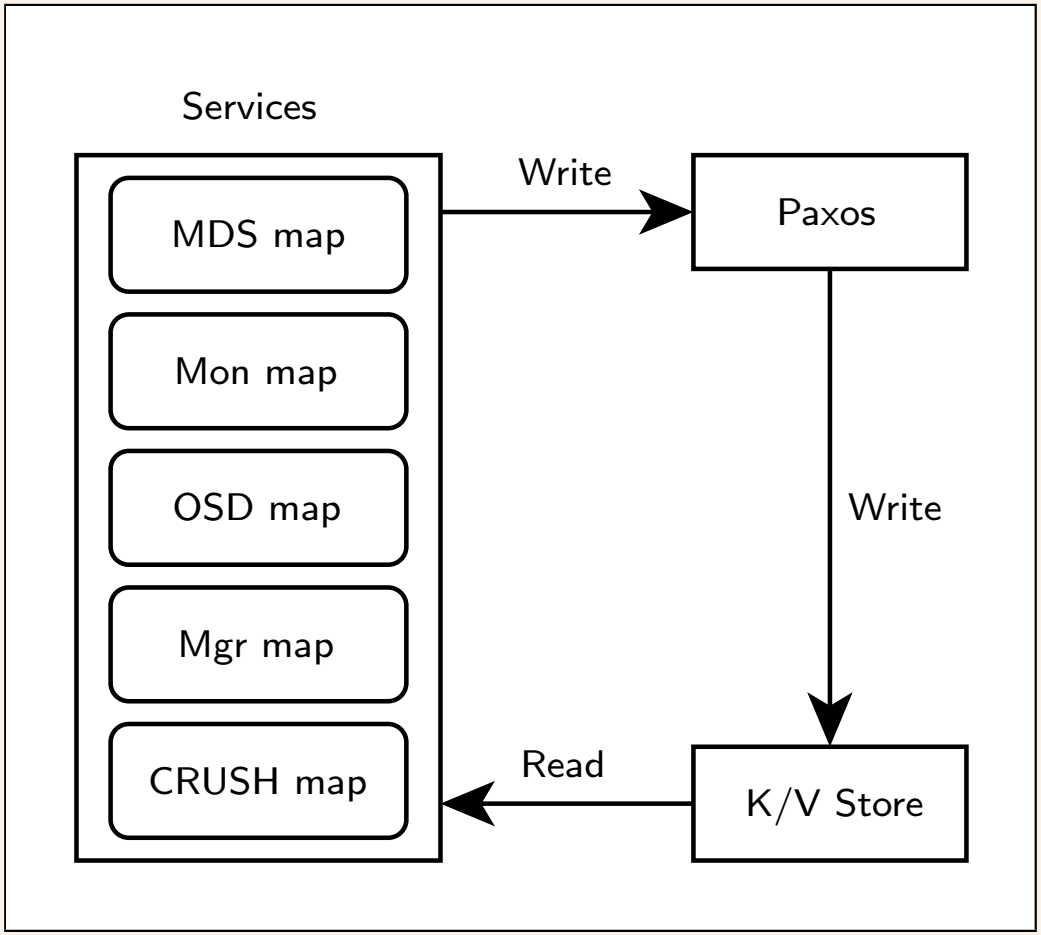

Figure: Diagram of a Ceph monitor.

Ceph Consensus Algorithm

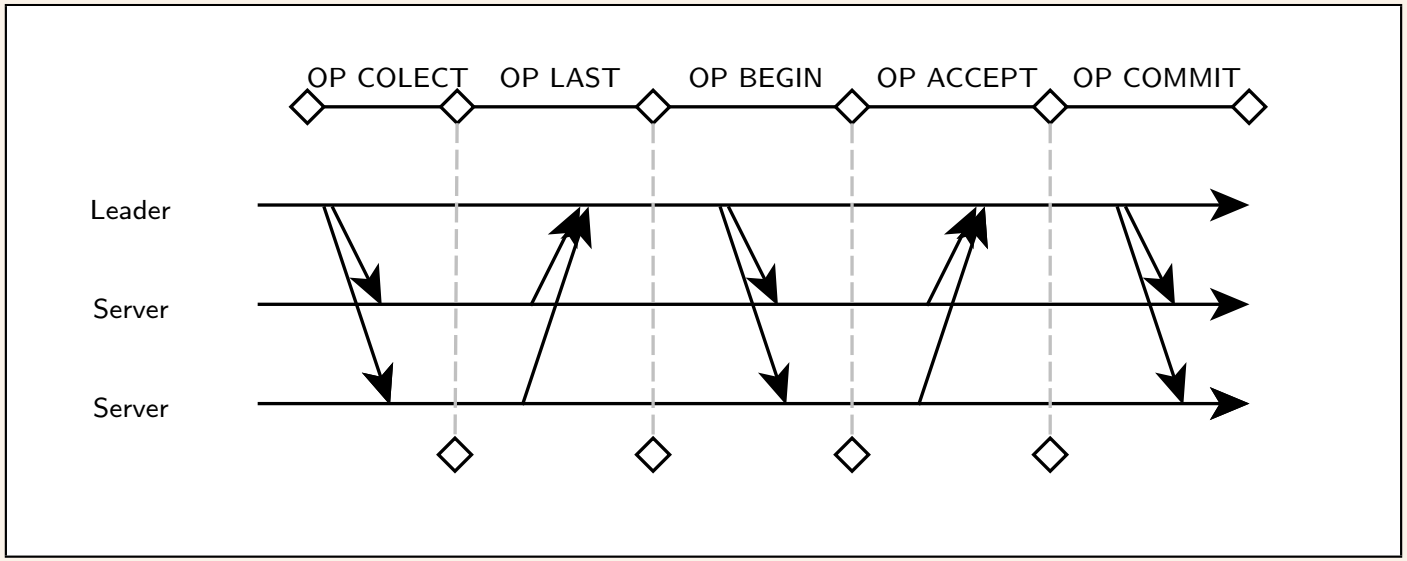

Figure: Diagram of the Ceph consensus algorithm.

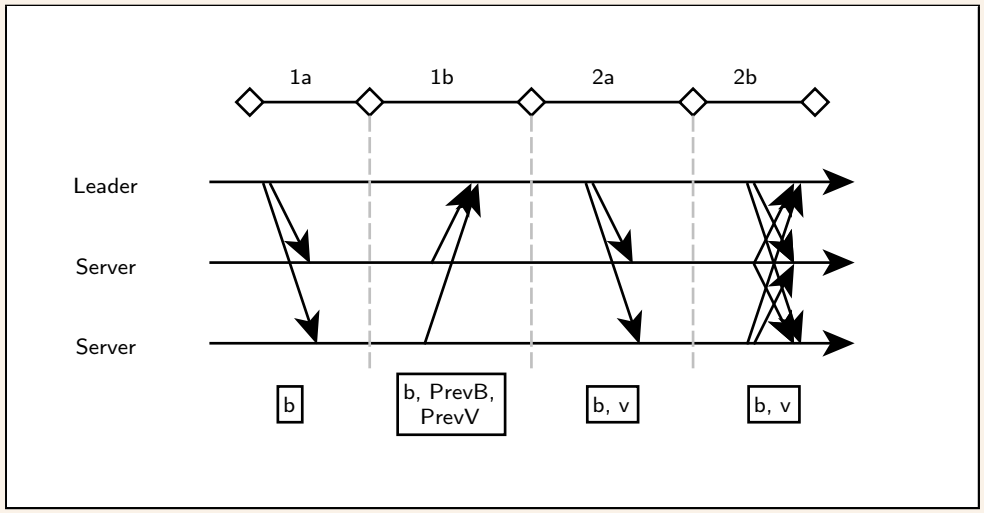

Figure: Diagram of the Paxos consensus algorithm.

Ceph Specification - Design Choices

- Communication based on TCP protocol.
  - No message duplication.
  - No message delivered out of order.
  - Message lost when a new epoch starts.
- Election logic.
  - Only one leader elected per epoch.
  - Quorum formed with all available monitors.
- Failure model.
  - Variable that tracks which monitors are down.
  - A monitor crash/recover triggers an election.
- Transactions (values committed).
  - A transaction corresponds to a change of a single value.
  - Values are proposed to the leader of the current epoch.

Ceph Specification - Overview

- Declaration of constants and variables.
- Initial predicates (variables initialization).
- Message manipulation predicates.
- Helper predicates.
- Main algorithm:
  - Lease phase predicates.
  - Commit phase predicates.
  - Client Request.
  - Collect phase predicates.
- Leader election.
- Crash and recover predicates.
- Dispatchers and next statement.
- Invariants.

Ceph Specification

`https://github.com/afonsonf/ceph-consensus-spec`

Verifying the Safety of the Algorithm

```
same_monitor_store ==
  \A mon1, mon2 \in Monitors:
    /\ state[mon1] = STATE_ACTIVE
    /\ state[mon2] = STATE_ACTIVE
    => monitor_store[mon1] = monitor_store[
       mon2]

same_monitor_values ==
  \A version \in 1..Max({last_committed[mon
     ]: mon \in Monitors}):
    \E val \in Value_set:
      \A mon \in Monitors:
        \/ last_committed[mon] < version
        \/ values[mon][version] = val
```

Verifying the Specification

```
In the send_collect collect operator:
[-] /\ uncommitted_pn' = [uncommitted_pn
   EXCEPT ![mon] = pending_pn[mon]]

[+] /\ uncommitted_pn' = [uncommitted_pn
   EXCEPT ![mon] = accepted_pn[mon]]


In the hanle_collect operator:
[-] uncommitted_pn  |-> pending_pn[mon],

[+] uncommitted_pn  |-> accepted_pn[mon],
```

Verifying the Specification

TLA Toolbox Demo

Table of Contents

1. Background
2. Ceph Consensus Algorithm Specification
3. Visualization Tool
4. Results
5. Future Work

Algorithm Visualization

- Motivation: search and understand example behaviours that break symmetry.
- Web application tool to explore and animate TLA$^+$ specifications.
- Can be used with state graphs (dot format) and trace errors (tlc tool format).
- State representations can be personalized using JavaScript, HTML and CSS.
- State parser that converts the state variables into JavaScript objects.

Algorithm Visualization

https://github.com/afonsonf/tlaplus-graph-explorer

Table of Contents

1. Background
2. Ceph Consensus Algorithm Specification
3. Visualization Tool
4. Results
5. Future Work

Performance of the Model

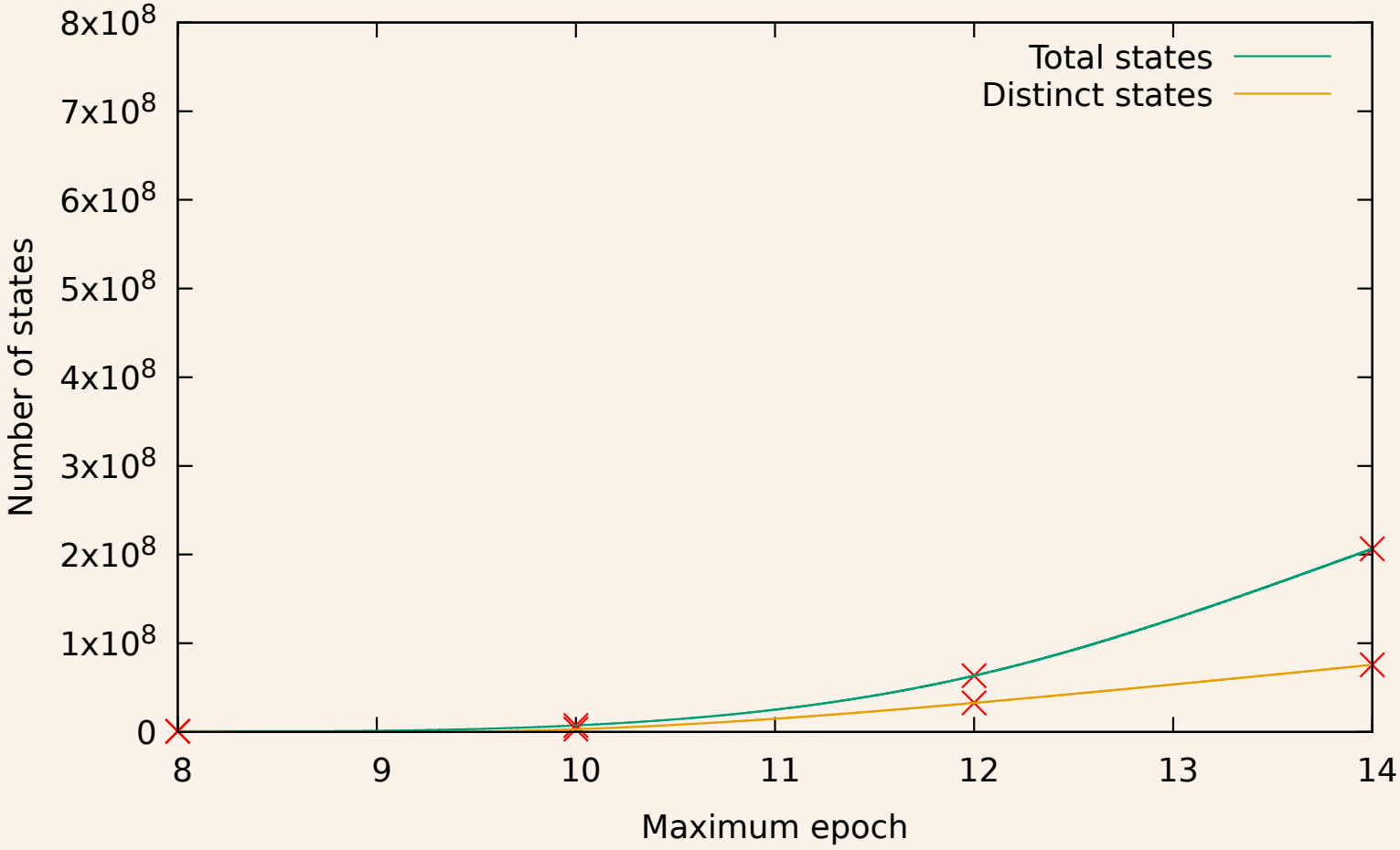

Figure: Graphic on the number of states generated (total and distinct) depending on the maximum possible epoch. Model with 3 monitors, a set of 2 possible values, a proposal number limit of 400, and limit on the number of committed values of 2.

Performance of the Model

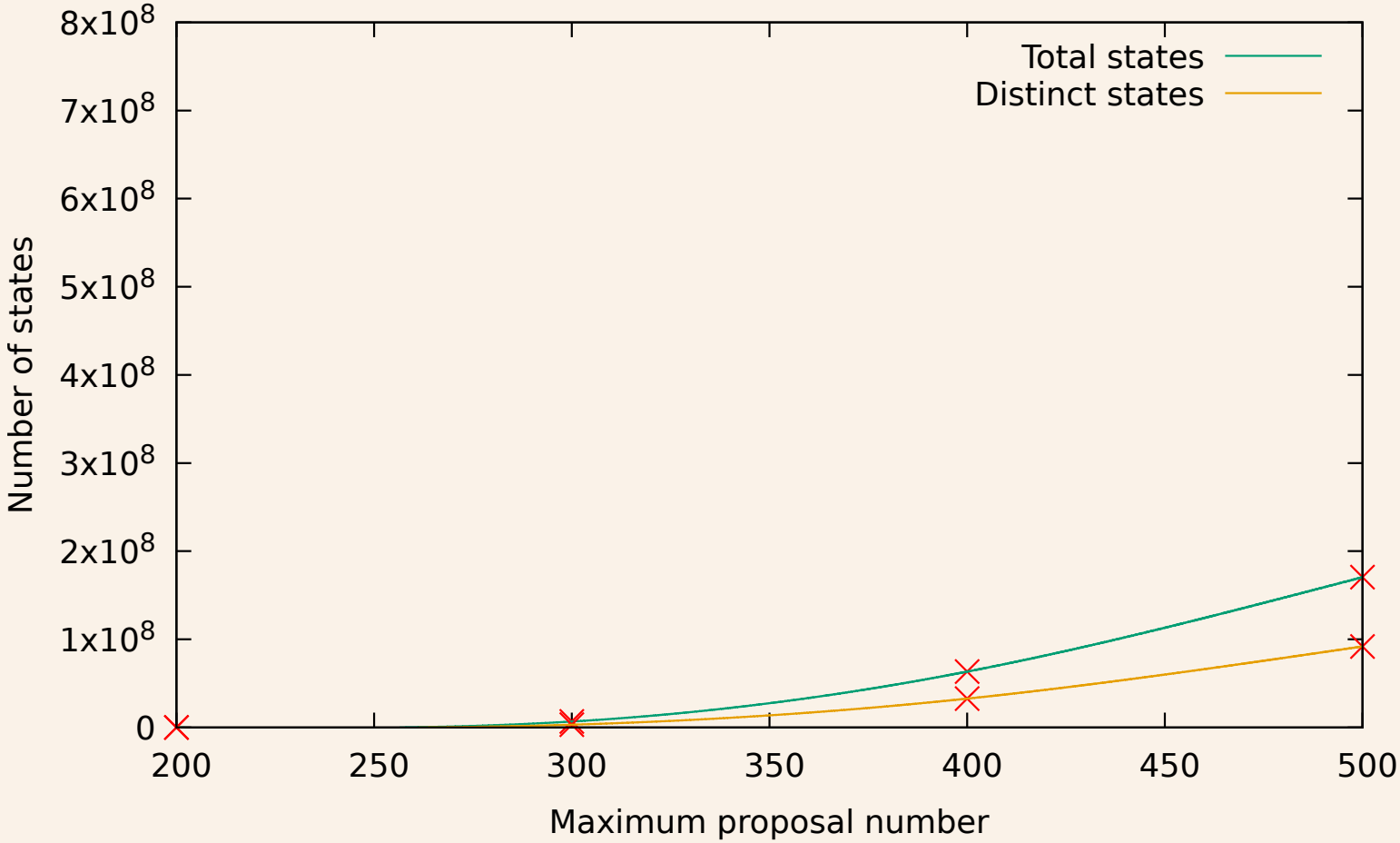

Figure: Graphic on the number of states generated (total and distinct) depending on the maximum possible proposal number. Model with 3 monitors, a set of 2 possible values, a epoch limit of 12, and limit on the number of committed values of 2.

Performance of the Model

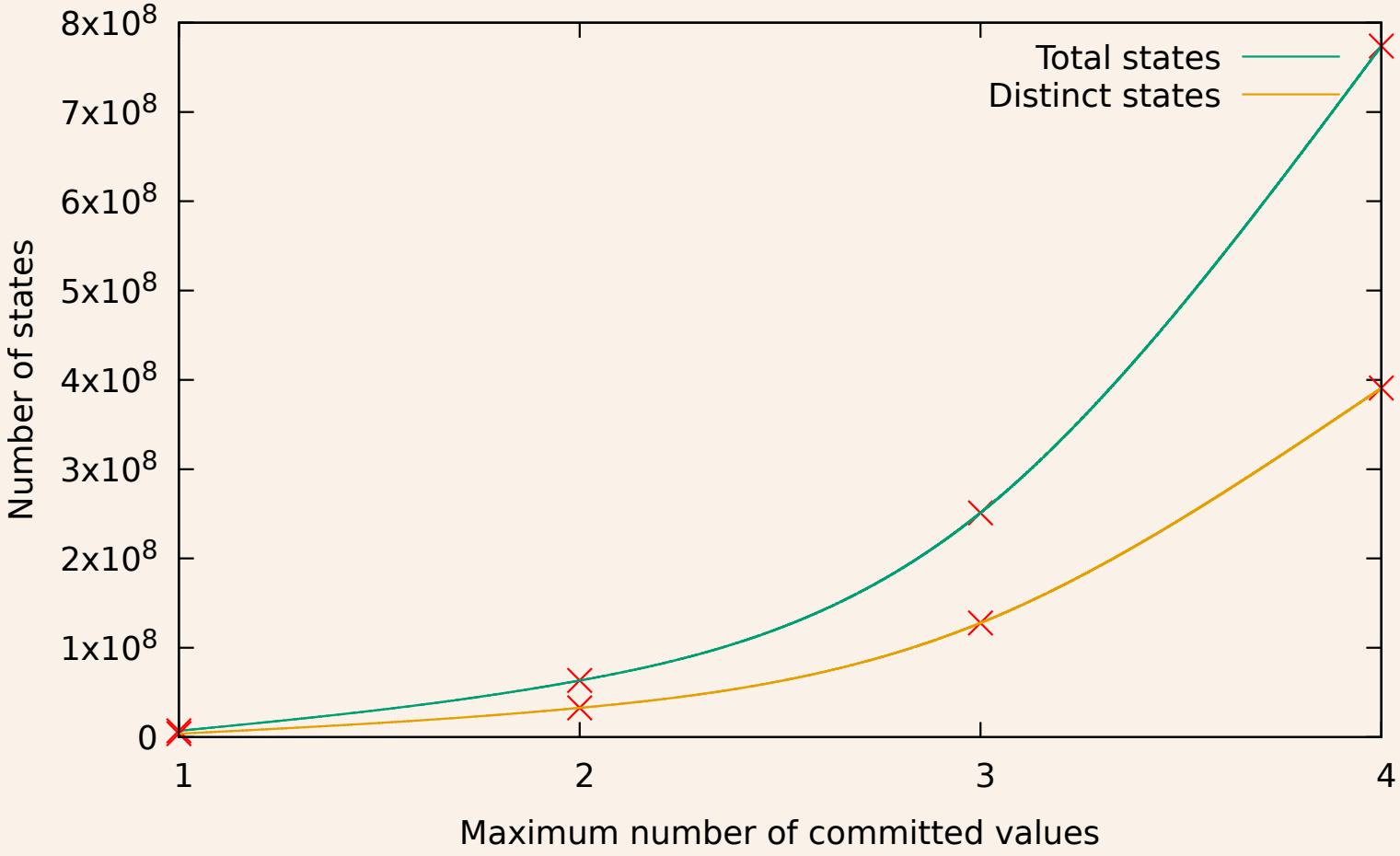

Figure: Graphic on the number of states generated (total and distinct) depending on the maximum number of committed values. Model with 3 monitors, a set of 2 possible values, a epoch limit of 12, and a proposal number limit of 400.

Performance of the Model

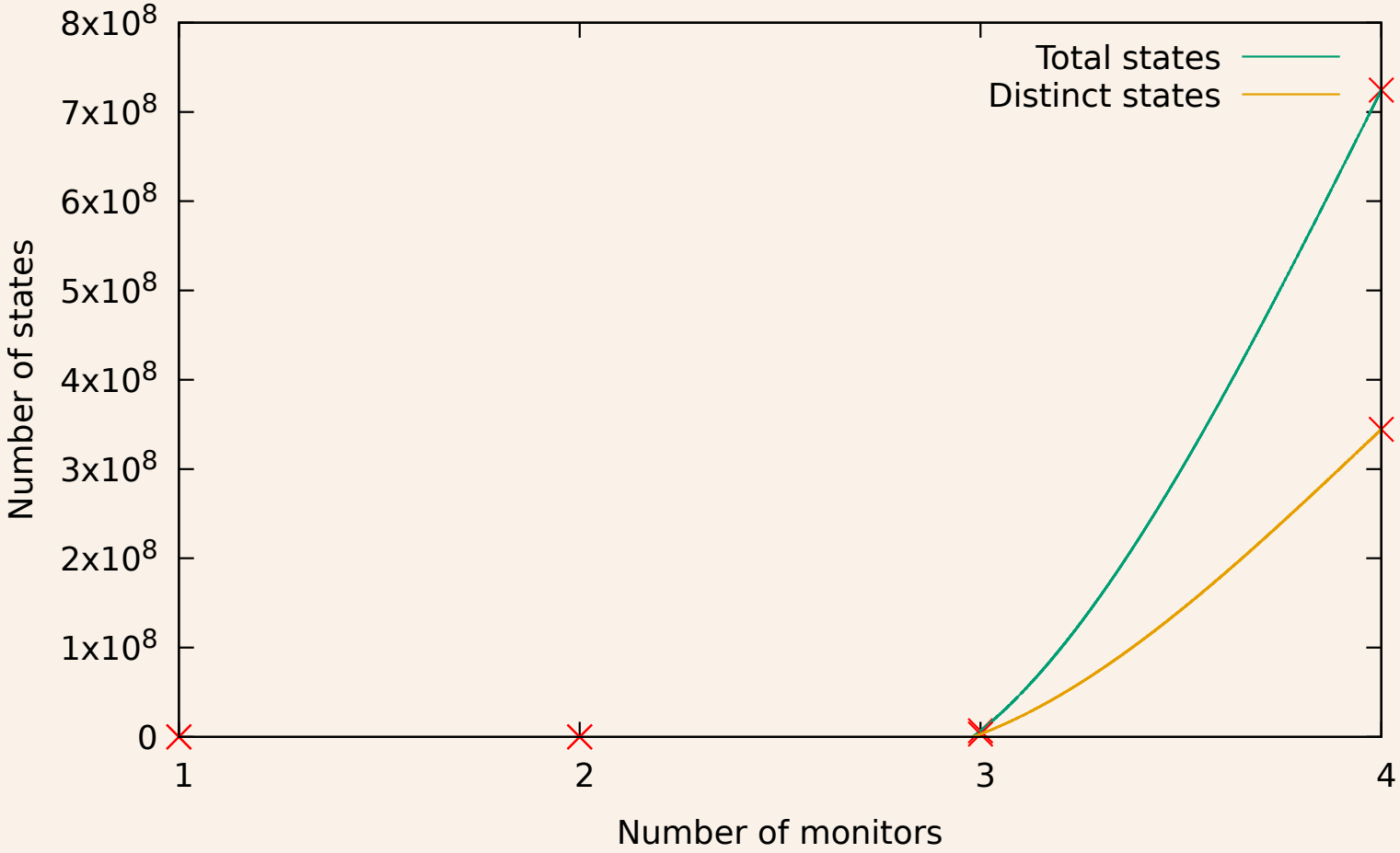

Figure: Graphic on the number of states generated (total and distinct) depending on the number of monitors in the system. Model a set of 2 possible values, a epoch limit of 12, a proposal number limit of 300, and limit on the number of committed values of 2.

Performance of the Visualization Tool

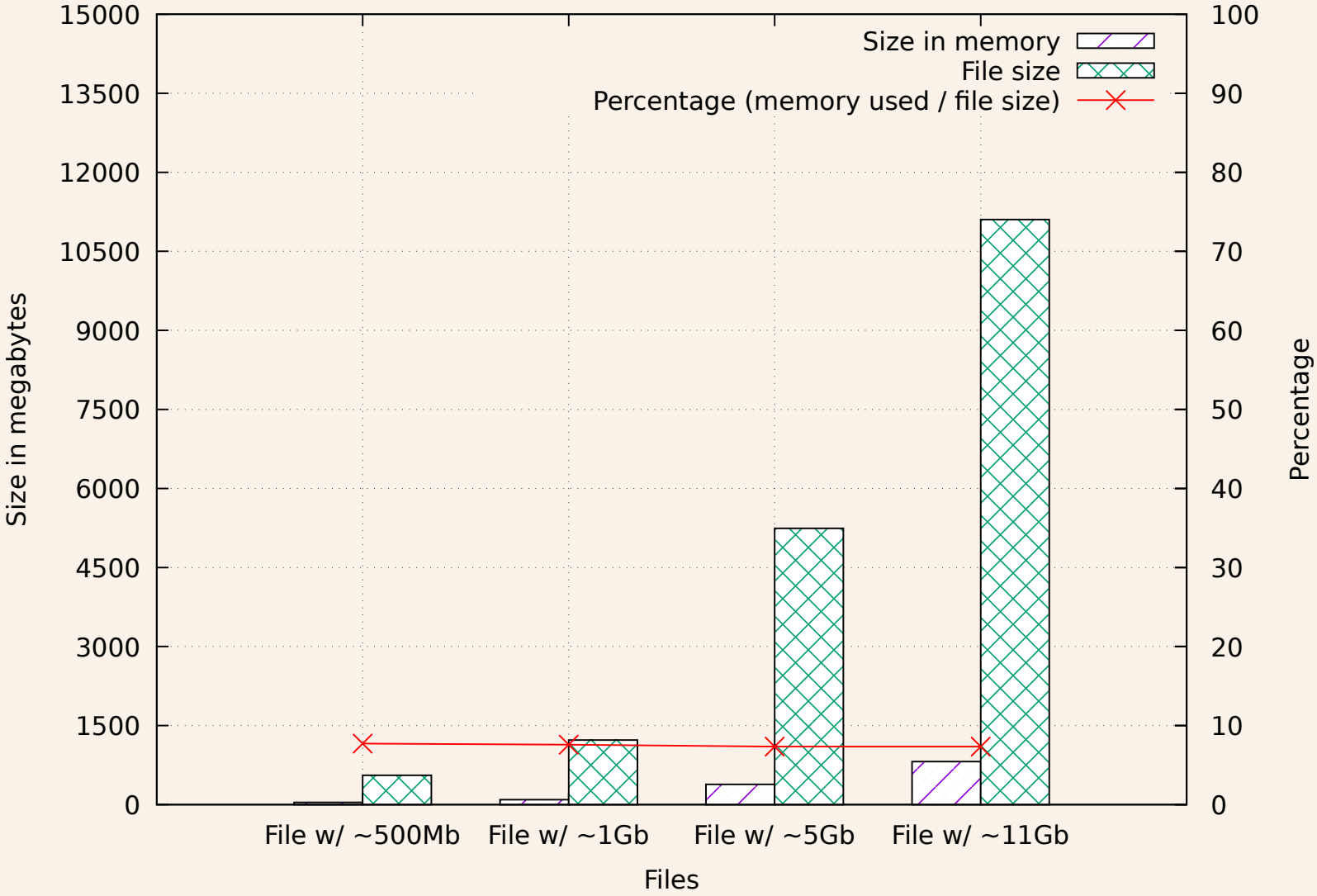

Figure: Graphic on the memory used by the visualization tool depending on the file size. Results obtained using the Firefox browser (Version 89.0.1) for Ubuntu 20.04.

Table of Contents

1. Background
2. Ceph Consensus Algorithm Specification
3. Visualization Tool
4. Results
5. Future Work

Future Work

- Specify and test other versions of consensus algorithm for Ceph.
- Rewrite the explorer tool in TLA$^+$.
- State print module: Function that takes the state variables and a template as arguments, and then prints the state to a file.
- Do the exploration using the prints from the previous module.
- Module with tools to (interactively?) explore a specification in runtime.

Contacts

linkedin.com/in/afonsonf

github.com/afonsonf