Brief Announcement: Design and Verification of a Logless Dynamic Reconfiguration Protocol in MongoDB Replication

William Schultz ✉
Northeastern University, Boston, MA, USA

Siyuan Zhou ✉
MongoDB, New York, NY, USA

Stavros Tripakis ✉
Northeastern University, Boston, MA, USA

Abstract

We introduce a novel dynamic reconfiguration protocol for the MongoDB replication system that extends and generalizes the single server reconfiguration protocol of the Raft consensus algorithm. Our protocol decouples the processing of configuration changes from the main database operation log, which allows reconfigurations to proceed in cases when the main log is prevented from processing new operations. Additionally, this decoupling allows for configuration state to be managed by a *logless* replicated state machine, storing only the latest version of the configuration and avoiding the complexities of a log-based protocol. We present a formal specification of the protocol in TLA+, initial verification results of model checking its safety properties, and an experimental evaluation of how reconfigurations are able to quickly restore a system to healthy operation when node failures have stalled the main operation log. This announcement is a short version and the full paper is available at [16].

2012 ACM Subject Classification Information systems → Parallel and distributed DBMSs; Software and its engineering → Software verification

Keywords and phrases Reconfiguration, Consensus, State Machine Replication

Digital Object Identifier 10.4230/LIPIcs.DISC.2021.61

Related Version *Full Version*: `https://arxiv.org/abs/2102.11960`

1 Introduction

Distributed replication systems based on the replicated state machine model [13] have become ubiquitous as the foundation of modern, fault-tolerant data storage systems. In order for these systems to ensure availability in the presence of faults, they must be able to dynamically replace failed nodes with healthy ones, a process known as dynamic reconfiguration. The protocols for building distributed replication systems have been well studied and implemented in a variety of systems [2, 20, 4, 18]. Paxos [5] and, more recently, Raft [11], have served as the logical basis for building provably correct distributed replication systems. Dynamic reconfiguration, however, is an additionally challenging and subtle problem [1] that has not been explored as extensively as the foundational consensus protocols underlying these systems. The Raft protocol, originally published in 2014, provided a dynamic reconfiguration algorithm in its initial publication, but did not include a precise discussion of its correctness or include a formal specification or proof. A critical safety bug [10] in one of its reconfiguration protocols was found after initial publication, which has since been fixed. The discovery of bugs like these, however, demonstrates that the design and verification of a safe dynamic reconfiguration protocol is a non-trivial task.

Since its inception, MongoDB [9], a general purpose distributed database, included a replication system [15] with a mechanism for clients to dynamically reconfigure replica membership. This legacy reconfiguration protocol was, however, unsafe in certain cases. In recent versions of MongoDB, reconfiguration has become a more common operation, necessitating the need for a redesigned, safe reconfiguration protocol with provable correctness guarantees. It was also desirable that this new protocol minimize changes to the existing, legacy protocol, which did not use a log-based approach to managing configurations. In this brief announcement we propose *MongoRaftReconfig*, a redesigned, safe reconfiguration protocol with provable correctness guarantees that minimizes changes to the existing, legacy protocol where possible. *MongoRaftReconfig* provides *logless* dynamic reconfiguration by decoupling the processing of configuration changes from the main database operation log, and improves upon and generalizes the single server reconfiguration protocol of standard Raft. This announcement is a short version of the full paper available at [16].

2 *MongoRaftReconfig*: A Logless Dynamic Reconfiguration Protocol

Many consensus based replication protocols [17, 7, 11] utilize the main operation log (referred to as the *oplog*, in MongoDB) to manage configuration changes by writing special reconfiguration log entries. *MongoRaftReconfig* instead decouples configuration updates from the main operation log by managing the configuration state of a replica set in a separate replicated state machine, which we refer to as the *config log*. The config log is maintained alongside the oplog, and manages the configuration state used by the overall protocol.

Decoupling these two conceptually distinct logs, the oplog and the config log, enables certain optimizations and simplifications in *MongoRaftReconfig* that would not be possible in a protocol where both logs are interleaved with each other. First, it allows for a simplification of the config log structure by observing that configuration changes are an "update only" operation. This obviates the need to store the entire log history, allowing the config log to operate as a *logless* replicated state machine, storing only the latest version of the configuration state. Second, it prevents the dynamics of either log negatively impacting the other unnecessarily. For example, it is possible to commit database writes in either log independently, without requiring previous writes in the other log to also become committed. This can allow the config log to bypass the oplog, allowing for reconfigurations in cases where a slow or stalled oplog replication channel would otherwise prevent reconfigurations from proceeding. For a detailed description of the protocol and its behaviors see [16].

3 Formal Specification and Model Checking

To formally describe and model check safety properties of *MongoRaftReconfig*, we use the TLA+ language [6]. TLA+ is a formal specification language for describing distributed and concurrent systems that is based on first order and temporal logic [12]. The full TLA+ specification of *MongoRaftReconfig* can be found at [14]. Note that the TLA+ language does not impose an underlying system or communication model (e.g. message passing, shared memory, etc.), which allows one to write specifications at a wide range of abstraction levels. Our specifications are written at a deliberately high level of abstraction, ignoring some lower level details of the protocol and system model. In practice, we have found the abstraction level of our specifications most useful for understanding and communicating the essential behaviors and safety characteristics of the protocol, while also serving to make automated verification feasible.

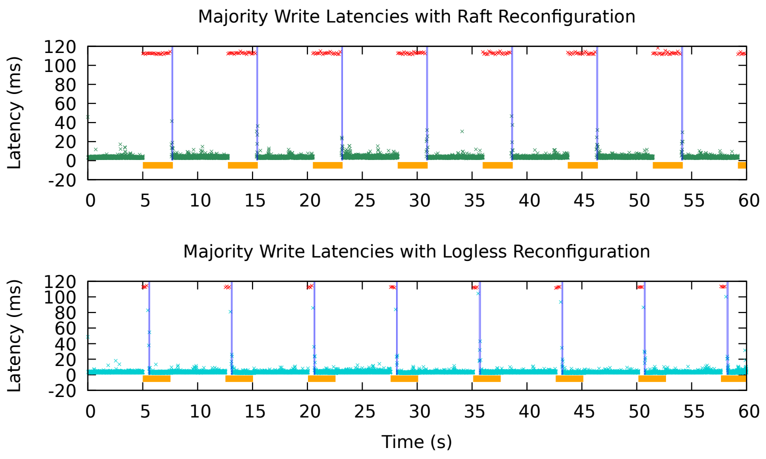

Figure 1 Latency of majority writes in the face of node degradation and reconfiguration to recover. Red points indicate writes that timed out. Orange bars indicate intervals of time where system entered a *degraded* mode. Vertical blue bars indicate completion of reconfiguration events.

For initial verification, we undertook an automated approach using TLC [19], an explicit state model checker for TLA+ specifications. We verified finite instances of the protocol to provide a sound guarantee of protocol correctness for fixed, finite parameters. It has been observed elsewhere [8] that relatively small, finite instances of distributed protocols are often sufficient to exhibit behaviors that are generalizable to larger (potentially infinite) instances, which helps to provide initial confidence in protocol correctness. This verification approach is, however, incomplete, in the sense that it does not establish correctness of the protocol for an unbounded number of servers. A goal for future work is to develop a general safety proof using the TLA+ proof system [3].

4 Experimental Evaluation

To demonstrate the benefits of *MongoRaftReconfig*, we designed an experiment to measure how quickly a replica set can reconfigure in new nodes to restore write availability when it faces periodic phases of degradation. For comparison, we implemented a simulated version of the Raft reconfiguration algorithm in MongoDB by having reconfigurations write a no-op oplog entry and requiring it to become committed before the reconfiguration can complete. Our experiment initiates a 5 node replica set and we periodically simulate a failure of two nodes, reconfigure them out of the set, and add in two new, healthy nodes. The ability of *MongoRaftReconfig* to quickly reconfigure in this scenario is seen in the results of Figure 1. Full details on the experimental setup are presented in [16].

References

1 Marcos Aguilera, Idit Keidar, Dahlia Malkhi, Jean-Philippe Martin, and Alexander Shraer. Reconfiguring Replicated Atomic Storage: A Tutorial. *Bulletin of the European Association for Theoretical Computer Science EATCS*, 2010.

2 Tushar D Chandra, Robert Griesemer, and Joshua Redstone. Paxos Made Live: An Engineering Perspective. In *Proceedings of the Twenty-Sixth Annual ACM Symposium on Principles of Distributed Computing*, PODC '07, pages 398–407, New York, NY, USA, 2007. Association for Computing Machinery. `doi:10.1145/1281100.1281103`.

3 Kaustuv Chaudhuri, Damien Doligez, Leslie Lamport, and Stephan Merz. Verifying safety properties with the tla+ proof system. In *International Joint Conference on Automated Reasoning*, pages 142–148. Springer, 2010.

4 Dongxu Huang, Qi Liu, Qiu Cui, Zhuhe Fang, Xiaoyu Ma, Fei Xu, Li Shen, Liu Tang, Yuxing Zhou, Menglong Huang, Wan Wei, Cong Liu, Jian Zhang, Jianjun Li, Xuelian Wu, Lingyu Song, Ruoxi Sun, Shuaipeng Yu, Lei Zhao, Nicholas Cameron, Liquan Pei, and Xin Tang. TiDB: a Raft-based HTAP database. *Proceedings of the VLDB Endowment*, 2020. doi:10.14778/3415478.3415535.

5 Leslie Lamport. The Part-Time Parliament. *ACM Transactions on Computer Systems*, 1998. doi:10.1145/279227.279229.

6 Leslie Lamport. *Specifying Systems: The TLA+ Language and Tools for Hardware and Software Engineers*. Addison-Wesley, June 2002.

7 Leslie Lamport, Dahlia Malkhi, and Lidong Zhou. Stoppable paxos. *TechReport, Microsoft Research*, 2008.

8 Haojun Ma, Aman Goel, Jean Baptiste Jeannin, Manos Kapritsos, Baris Kasikci, and Karem A. Sakallah. I4: Incremental inference of inductive invariants for verification of distributed protocols. In *SOSP 2019 - Proceedings of the 27th ACM Symposium on Operating Systems Principles*, 2019. doi:10.1145/3341301.3359651.

9 MongoDB Github Project, 2021. URL: https://github.com/mongodb/mongo.

10 Diego Ongaro. Bug in single-server membership changes, July 2015. URL: https://groups.google.com/g/raft-dev/c/t4xj6dJTP6E/m/d2D9LrWRza8J.

11 Diego Ongaro and John Ousterhout. In Search of an Understandable Consensus Algorithm. In *Proceedings of the 2014 USENIX Conference on USENIX Annual Technical Conference*, USENIX ATC'14, pages 305–320, USA, 2014. USENIX Association.

12 Amir Pnueli. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*, pages 46–57. IEEE, 1977.

13 Fred B. Schneider. Implementing Fault-Tolerant Services Using the State Machine Approach: A Tutorial. *ACM Computing Surveys (CSUR)*, 1990. doi:10.1145/98163.98167.

14 William Schultz. MongoRaftReconfig TLA+ Specification, 2021. URL: https://github.com/will62794/logless-reconfig/blob/3d6b378eae7dabe3a35a0b4042bfc55fd178cb21/specs/MongoRaftReconfig.tla.

15 William Schultz, Tess Avitabile, and Alyson Cabral. Tunable consistency in mongodb. *Proceedings of the VLDB Endowment*, 12(12):2071–2081, 2019.

16 William Schultz, Siyuan Zhou, and Stavros Tripakis. Design and verification of a logless dynamic reconfiguration protocol in mongodb replication. *arXiv preprint*, 2021. arXiv:2102.11960.

17 Alexander Shraer, Benjamin Reed, Dahlia Malkhi, and Flavio Junqueira. Dynamic reconfiguration of primary/backup clusters. In *Proceedings of the 2012 USENIX Annual Technical Conference, USENIX ATC 2012*, 2019.

18 Rebecca Taft, Irfan Sharif, Andrei Matei, Nathan VanBenschoten, Jordan Lewis, Tobias Grieger, Kai Niemi, Andy Woods, Anne Birzin, Raphael Poss, Paul Bardea, Amruta Ranade, Ben Darnell, Bram Gruneir, Justin Jaffray, Lucy Zhang, and Peter Mattis. CockroachDB: The Resilient Geo-Distributed SQL Database. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, SIGMOD '20, pages 1493–1509, New York, NY, USA, 2020. Association for Computing Machinery. doi:10.1145/3318464.3386134.

19 Yuan Yu, Panagiotis Manolios, and Leslie Lamport. Model checking tla+ specifications. In *Advanced Research Working Conference on Correct Hardware Design and Verification Methods*, pages 54–66. Springer, 1999.

20 Jianjun Zheng, Qian Lin, Jiatao Xu, Cheng Wei, Chuwei Zeng, Pingan Yang, and Yunfan Zhang. PaxosStore: High-availability storage made practical in WeChat. In *Proceedings of the VLDB Endowment*, 2017. doi:10.14778/3137765.3137778.