Replicated state machines without replicated execution

Jonathan Lee Kirill Nikitin* Srinath Setty
Microsoft Research **EPFL*

Abstract

This paper introduces a new approach to reduce end-to-end costs in large-scale replicated systems built under a Byzantine fault model. Specifically, our approach transforms a given replicated state machine (RSM) to another RSM where nodes incur lower costs by *delegating* state machine execution: an untrusted prover produces succinct cryptographic proofs of correct state transitions along with state changes, which nodes in the transformed RSM verify and apply respectively.

To realize our approach, we build *Piperine*, a system that makes the proof machinery profitable in the context of RSMs. Specifically, Piperine reduces the costs of both proving and verifying the correctness of state machine execution while retaining liveness—a distinctive requirement in the context of RSMs. Our experimental evaluation demonstrates that, for a payment service, employing Piperine is more profitable than naive reexecution of transactions as long as there are $> 10^4$ nodes. When we apply Piperine to ERC-20 transactions in Ethereum (a real-world RSM with up to 10^5 nodes), it reduces per-transaction costs by 5.4× and network costs by 2.7×.

1 Introduction

A modern example of a large-scale replicated system is a blockchain network [64, 86], which employs replication to enable mutually-distrusting entities to transact *without* relying on trusted authorities. Specifically, blockchains instantiate *replicated state machines* (RSMs) [71] under a Byzantine fault model in an open, permissionless network where each node executes and validates every transaction. Unfortunately, the most popular blockchains achieve a throughput of only a handful of transactions per second. This has motivated research to improve throughput and to reduce costs, for example, by changing the underlying consensus protocol used to realize RSMs [41, 46, 50]. These proposals, however, introduce additional assumptions for safety and/or liveness (§7).

We consider a different approach, one that applies to any existing replicated state machine in a Byzantine fault model (including blockchains) *without* any changes to the underlying consensus protocol. Naturally, it does not introduce any strong assumptions for safety or liveness. In fact, this approach is complementary to aforementioned advances [41, 46, 50] and can be used in conjunction with those proposals. Our approach is based on work in the area of *proof-based verifiable computation* (see [83] for a survey), which has developed a powerful primitive called *verifiable state machines* [24, 73]: for a state machine S and a batch of transactions x, an untrusted *prover* can produce outputs y and a short proof π such that a *verifier* can check if y is the correct output of S with x as input (using π)—*without* reexecuting the state transitions. Furthermore, the cost of verifying such a proof is less than reexecuting the corresponding state transitions and the size of the proof is far less than the size of the original batch of transactions. Thus, nodes (in an RSM) that replicate a state machine S can delegate S to an untrusted prover and then replicate the verifier at each node to verify the prover's proofs. Naturally, if the end-to-end resource costs of the transformed RSM (CPU, storage, network, etc.) is cheaper than the original RSM, verifiable delegation leads to lower costs.

In theory, the above picture is straightforward and offers a principled solution to reduce end-to-end costs of a replicated system. However, in practice, the above approach is completely impractical. Specifically, even with state-of-the-art systems for verifiable outsourcing, the verifier is more resource-efficient compared to reexecution only under narrow regimes [80, 81, 83]. Furthermore, in the context of RSMs, the verifier running at each node must have a copy of the delegated state machine's state, otherwise liveness of the transformed RSM hinges on the liveness of the prover (relying on the prover for liveness introduces attack vectors for mounting denial of service). Finally, the prover's cost to produce a proof is 10^4–$10^7\times$ higher than natively executing the corresponding state transition (the overheads depends on whether the outsourced computation is efficiently representable in the computational model of the proof machinery) [73, 81].

The primary contribution of this paper is a set of techniques to reduce the costs of verifiable state machines in the context of RSMs and to ensure liveness without increasing the costs of the prover. To demonstrate the benefits of these techniques, we build a system called *Piperine*. When we apply Piperine to a popular type of state machine on Ethereum's blockchain, Piperine's proofs act as compressed information (e.g., there is no need to transmit digital signatures or the raw transactions over the blockchain), which allows Piperine to transparently reduce per-transaction network costs by 2.7× and per-transaction end-to-end costs by 5.4×. Beyond cost reductions, Piperine resolves an open question in the context of replicated systems: Piperine offers the first approach to build RSMs with concurrent transaction processing in a permissionless model. Note that prior works that achieve concurrent transaction processing in RSMs [9, 49] require substantial changes to the underlying consensus protocol and apply only to a permissioned membership model.

Reducing costs. To tame costs imposed by the proof machinery, Piperine leverages the following observations: (1) in our target state machines, the primary computational bottleneck of a state transition is authenticating a transaction by verify-

DOI 10.1109/SP40000.2020.00068