BChain: Byzantine Replication with High Throughput and Embedded Reconfiguration

Sisi Duan[1], Hein Meling[2], Sean Peisert[1], and Haibin Zhang[1]

[1] University of California, Davis, USA
{sduan,speisert,hbzhang}@ucdavis.edu
[2] University of Stavanger, Norway
hein.meling@uis.no

Abstract. In this paper, we describe the design and implementation of BChain, a Byzantine fault-tolerant state machine replication protocol, which performs comparably to other modern protocols in fault-free cases, but in the face of failures can also quickly recover its steady state performance. Building on chain replication, BChain achieves high throughput and low latency under high client load. At the core of BChain is an efficient Byzantine failure detection mechanism called *re-chaining*, where faulty replicas are placed out of harm's way at the end of the chain, until they can be replaced. Our experimental evaluation confirms our performance expectations for both fault-free and failure scenarios. We also use BChain to implement an NFS service, and show that its performance overhead, with and without failures, is low, both compared to unreplicated NFS and other BFT implementations.

1 Introduction

Building online services that are both highly available and correct is challenging. Byzantine fault tolerance (BFT), a technique based on state machine replication [25,31], is the only known *general* technique that can mask *arbitrary* failures, including crashes, malicious attacks, and software errors. Thus, the behavior of a service employing BFT is indistinguishable from a service running on a correct server.

There are two broad classes of BFT protocols that have evolved in the past decade: broadcast-based [5,24,1,12] and chain-based protocols [18,34]. The main difference between these two classes is their performance characteristics. Chain-based protocols aim at achieving high throughput, at the expense of higher latency. However, as the number of concurrent client requests grows, it turns out that chain-based protocols can actually achieve lower latency than broadcast-based protocols. The downside however, is that chain-based protocols are less resilient to failures, and typically relegate to broadcasting when failures are present. This results in a significant performance degradation.

In this paper we propose *BChain*, a fully-fledged BFT protocol addressing the performance issues observed when a BFT service experiences failures. Our evaluation shows that BChain can quickly recover its steady-state performance, while Aliph-Chain [18] and Zyzzyva [24] experience significantly reduced performance, when subjected to a simple crash failure. At the same time, the steady-state performance of BChain is comparable to Aliph-Chain, the state-of-the-art, chain-based BFT protocol. BChain also

Table 1. Characteristics of state-of-the-art BFT protocols tolerating f failures with batch size b. Bold entries mark the protocol with the lowest cost. The critical path denotes the number of one-way message delays. *Two message delays is only achievable with no concurrency.

	PBFT	Q/U	HQ	Zyzzyva	Aliph	Shuttle	BChain-3	BChain-5
Total replicas	$3f+1$	$5f+1$	$3f+1$	$3f+1$	$3f+1$	$\mathbf{2f+1}$	$3f+1$	$5f+1$
Crypto ops	$2+\frac{8f+1}{b}$	$2+8f$	$4+4f$	$2+\frac{3f}{b}$	$1+\frac{f+1}{b}$	$2+\frac{2f}{b}$	$\mathbf{1+\frac{3f+2}{b}}$	$1+\frac{4f+2}{b}$
Critical path	4	$\mathbf{2}^*$	4	3	$3f+2$	$2f+2$	$2f+2$	$3f+2$
Additional Requirements	None	None	None	Correct Clients	Protocol Switch	Olympus; Reconfig.	Reconfig.	None

outperforms broadcast-based protocols PBFT [5] and Zyzzyva with a throughput improvement of up to 50% and 25%, respectively. We have used BChain to implement a BFT-based NFS service, and our evaluation shows that it is only marginally slower (1%) than a standard NFS implementation.

BChain in a Nutshell. BChain is a self-recovering, chain-based BFT protocol, where the replicas are organized in a chain. In common case executions, clients send their requests to the head of the chain, which orders the requests. The ordered requests are forwarded along the chain and executed by the replicas. Once a request reaches a replica that we call the *proxy tail* , a reply is sent to the client.

When a BFT service experiences failures or asynchrony, BChain employs a novel approach that we call *re-chaining*. In this approach, the head reorders the chain when a replica is suspected to be faulty, so that a fault cannot affect the critical path.

To facilitate re-chaining, BChain makes use of a novel failure detection mechanism, where any replica can suspect its successor and only its successor. A replica does this by sending a signed suspicion message up the chain. No proof that the suspected replica has misbehaved is required. Upon receiving a suspicion, the head issues a new chain ordering where the accused replica is moved out of the critical path, and the accuser is moved to a position in which it cannot continue to accuse others. In this way, correct replicas help BChain make progress by suspecting faulty replicas, yet malicious replicas cannot *constantly* accuse correct replicas of being faulty.

Our re-chaining approach is inexpensive; a single re-chaining request corresponds to processing a single client request. Thus, the steady-state performance of BChain has minimal disruption. The latency reduction caused by re-chaining is dominated by the failure detection timeout.

Our Contributions in Context. We consider two variants of BChain—BChain-3 and BChain-5, both tolerating f failures. BChain-3 requires $3f + 1$ replicas and a reconfiguration mechanism coupled with our detection and re-chaining algorithms, while BChain-5 requires $5f + 1$ replicas, but can operate without the reconfiguration mechanism. We compare BChain-3 and BChain-5 with state-of-the-art BFT protocols in Table 1. All protocols use MACs for authentication and request batching with batch size b. The number of MAC operations for BChain at the bottleneck server tends to one for gracious executions. While this is also the case for Aliph-Chain [18], Aliph requires that clients take responsibility for switching to another slower BFT protocol in the