ResilientDB — Global Scale Resilient Blockchain Fabric

Presented by Timothee Duthoit [40120801]

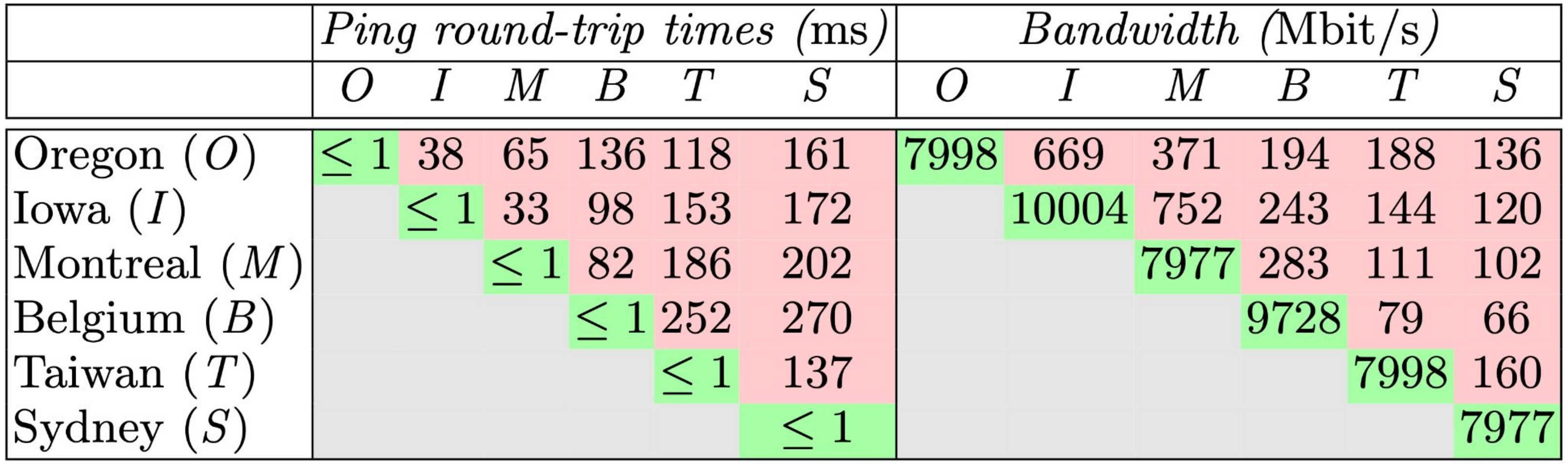
Table of Contents

Background and Motivations

Issues with current applications

- Blockchain applications rely on Byzantine Fault-Tolerant (BFT) consensus
- Current BFT protocols have high latency and communication costs when nodes are spread across large areas
- For centralized decision making, the bandwidth of the primary tends to be a bottleneck
- Global scale blockchain applications could benefit from a topology-aware system which optimizes global consensus

Real World Cluster-to-Cluster Communication Costs



	Ping round-trip times (ms)						*Bandwidth* (Mbit/s)					
	O	I	M	B	T	S	O	I	M	B	T	S
Oregon (O)	≤ 1	38	65	136	118	161	7998	669	371	194	188	136
Iowa (I)		≤ 1	33	98	153	172		10004	752	243	144	120
Montreal (M)			≤ 1	82	186	202			7977	283	111	102
Belgium (B)				≤ 1	252	270				9728	79	66
Taiwan (T)					≤ 1	137					7998	160
Sydney (S)						≤ 1						7977

Benefits of Topology-Aware Consensus

Allow parallel consensus execution

Minimize communication between distant nodes

Group nearby nodes/replicas into clusters

Minimize required coordination between clusters

Design of GeoBFT

Main Components of GeoBFT’s Design

- Aware of the **network topology**
- Nearby replicas are grouped together into **clusters**
- Makes use of **rounds** in which clusters commit one transaction each
- Each cluster achieves consensus independently using **PBFT**
- A novel **global sharing protocol** for inter-cluster communication
- Clusters share **commit certificates** with each other
- A novel **remote view change** protocol is used to handle failures
- Guarantees **safety** and **liveness**

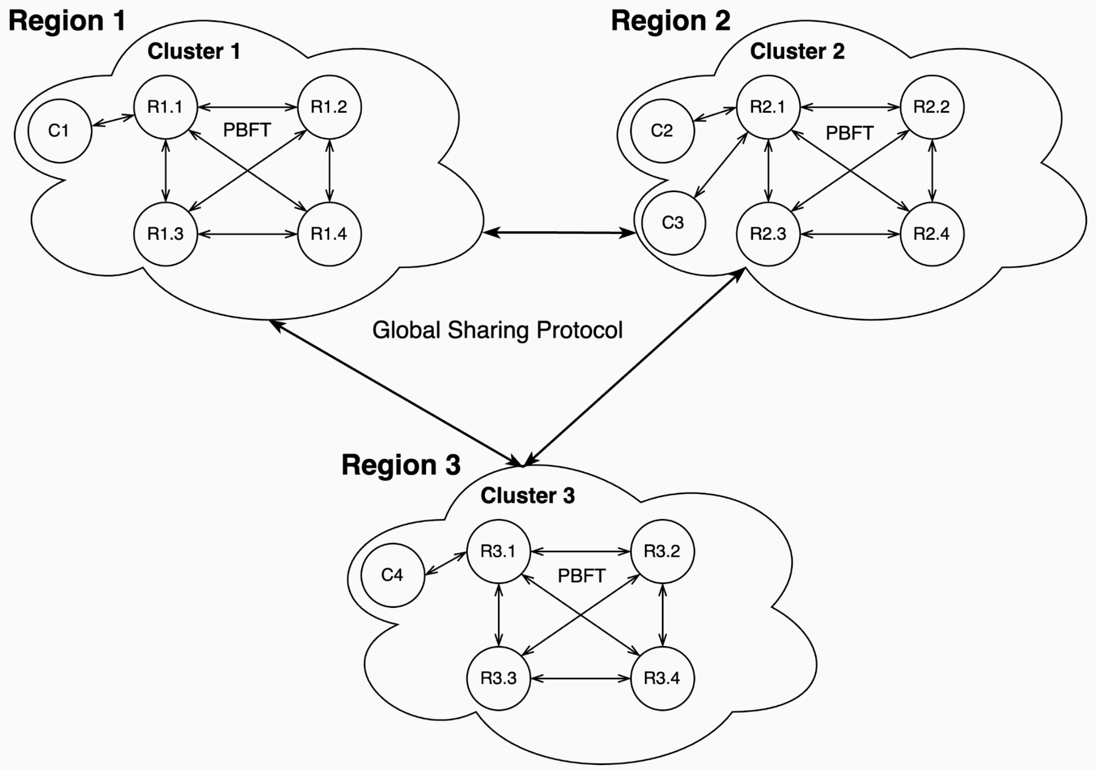
GeoBFT — Relation Between Clusters, Replicas and Failures

For **z** clusters of **n** replicas each and at most **f** failing replicas in each cluster we have:

- Number of reliable replicas per cluster = **n − f**
- Each cluster must have **n > 3f** or **n ≥ 3f + 1**
- Maximum number of failures tolerated is **fz** with no more than **f** failures in each cluster

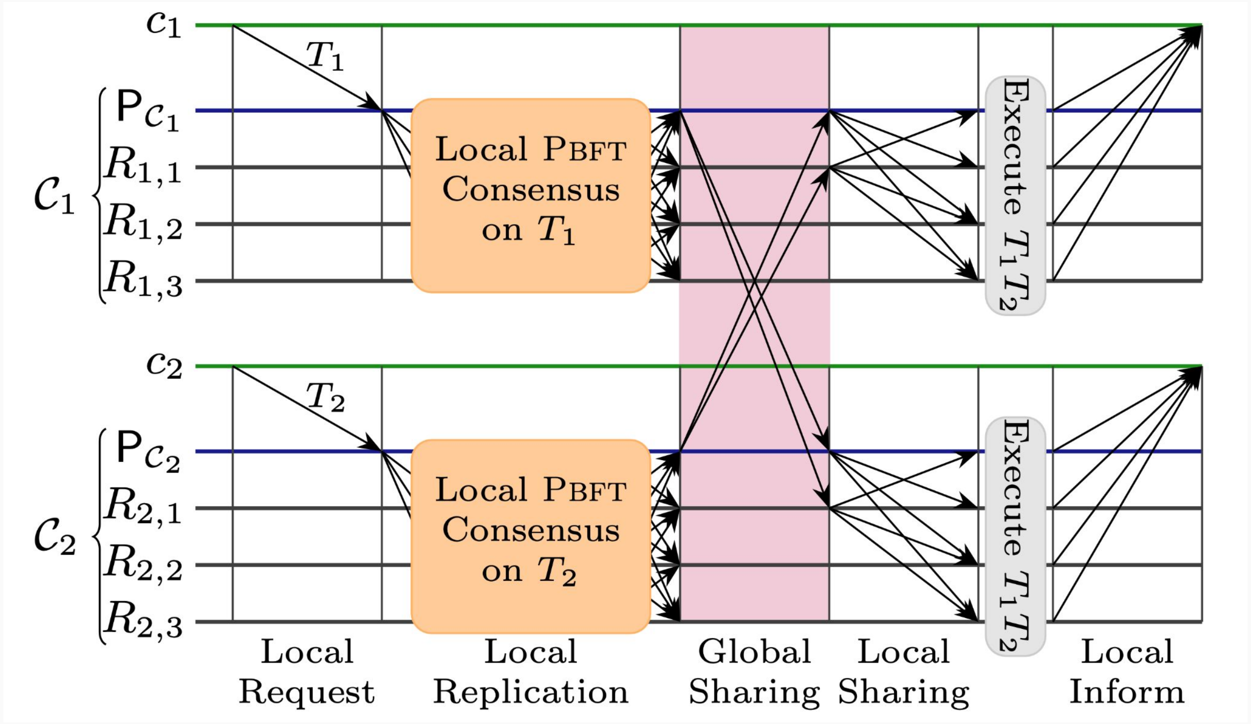
Example: For 3 clusters, **z = 3**, of 4 replicas each **n = 4** → **f = 1** and the total number of failures is **fz = 3**

Network Overview

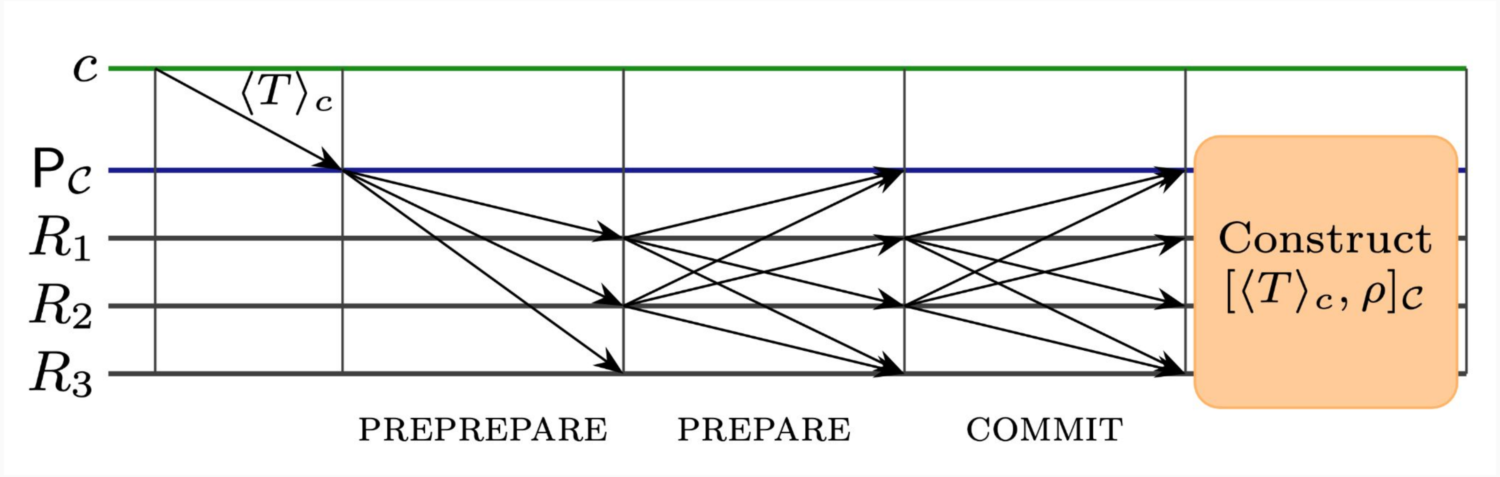


GeoBFT — Normal Case Scenario

1. Client sends Transaction **T$_1$** to the nearest cluster's Primary **P$_{C1}$**
2. **P$_{C1}$** sends PREPREPARE message to all the replicas in the cluster
3. The cluster achieves local consensus using PBFT
4. **P$_{C1}$** sends **T$_1$** along with its commit certificate to other clusters
5. **P$_{C1}$** receives transactions with commit certificates from other clusters and broadcasts them
6. Replicas in all clusters execute/commit the transactions
7. All replicas inform their local clients of the executed transactions



GeoBFT — Local PBFT Consensus

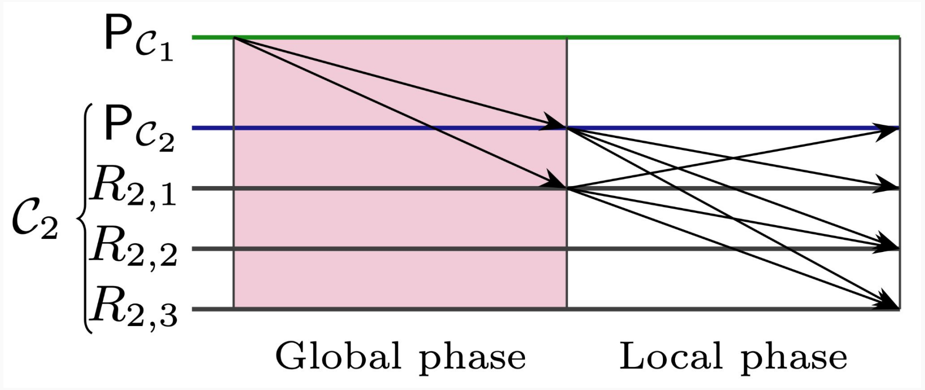


Global Sharing Protocol — Sending

$\mathbf{P_{C1}}$ send the transaction along with its commit certificate to $\mathbf{P_{C2}}$ and **f** other replicas in $\mathbf{C_2}$.

So each primary sends **f + 1** messages.

Sending **f + 1** messages allows the protocol to handle primary failures from the other end.

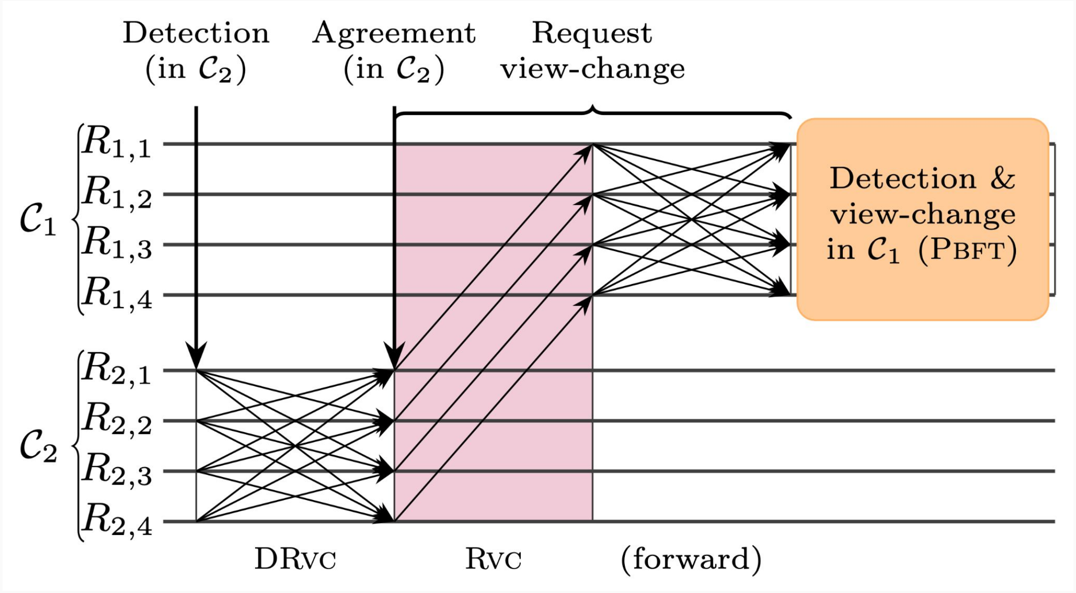


Global Sharing Protocol — Remote View Change

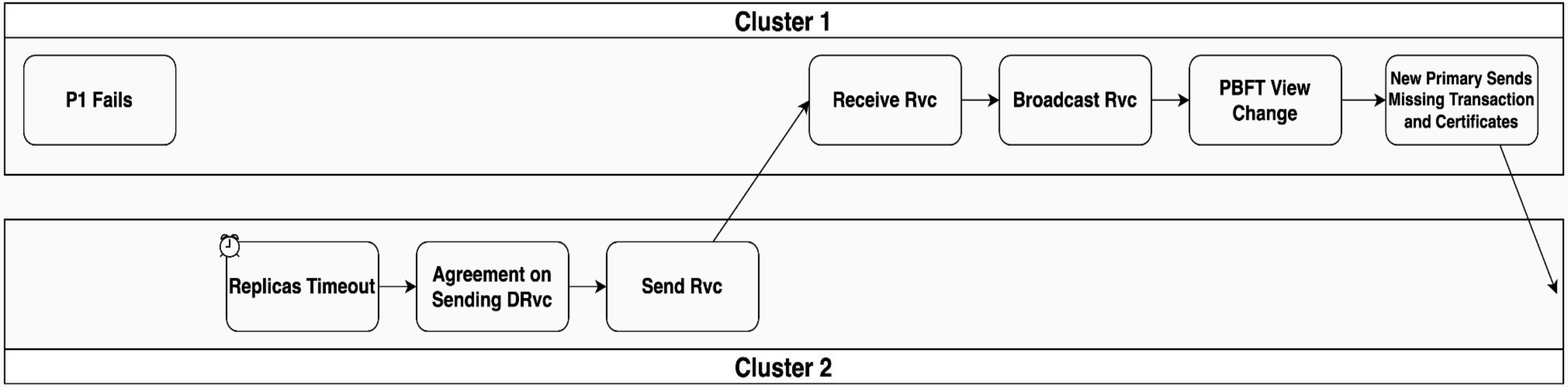
In the event where a primary fails, other clusters won't receive the transaction and certificate from C_1 for round ρ.

Using a timer, nodes from other clusters detect failure and initiate a view change to replace C_1's primary.

At most one view change can happen per round.

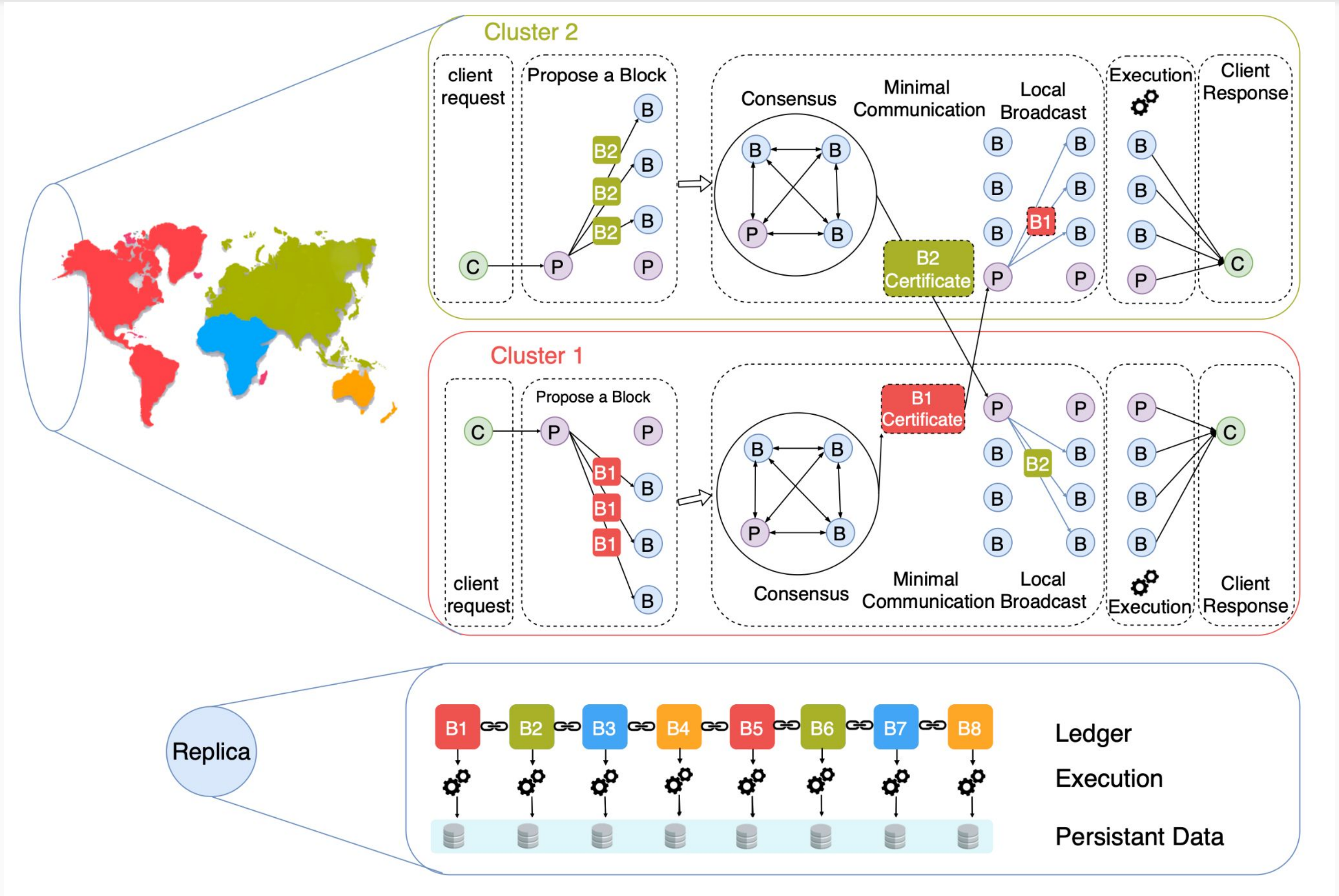


Global Sharing Protocol — Remote View Change



Implementation of GeoBFT in ResilientDB

Architecture of ResilientDB Fabric



ResilientDB Fabric — The Ledger

➔ The ledger is an **immutable, append-only blockchain**.

➔ The ledger records the **ordered sequence** of accepted client requests.

➔ Each block **represents a single executed client request.**

➔ In each **round**, for **z clusters**, every replica executes **z requests** adding **z blocks.**

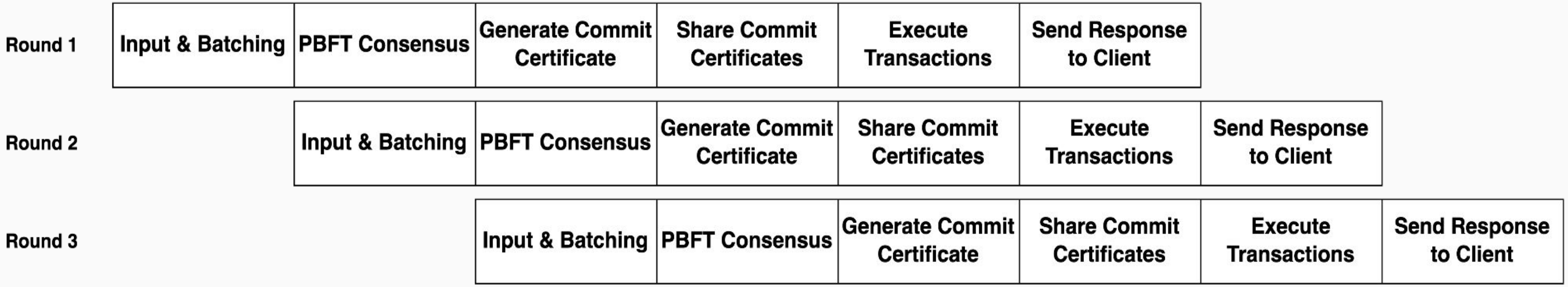
➔ The **z requests** added each belong to a different cluster C_i ($1 \leq i \leq z$).

ResilientDB Fabric — Cryptography

- ResilientDB uses **strong cryptographic primitives** to ensure secure communication and integrity.
- It **follows NIST recommendations** for security standards.
- **Key cryptographic mechanisms used**
  - **ED25519-based digital signatures** for signing messages.
  - **AES-CMAC** for authenticated communication.
  - **SHA256** for generating **collision-resistant message digests.**

ResilientDB Fabric — Pipelined Consensus

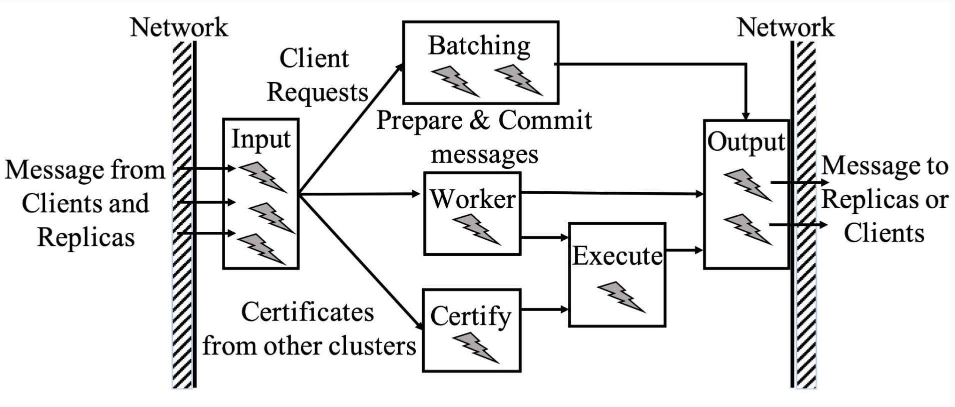
ResilientDB uses a **multi-threaded pipelined architecture** to optimize performance.



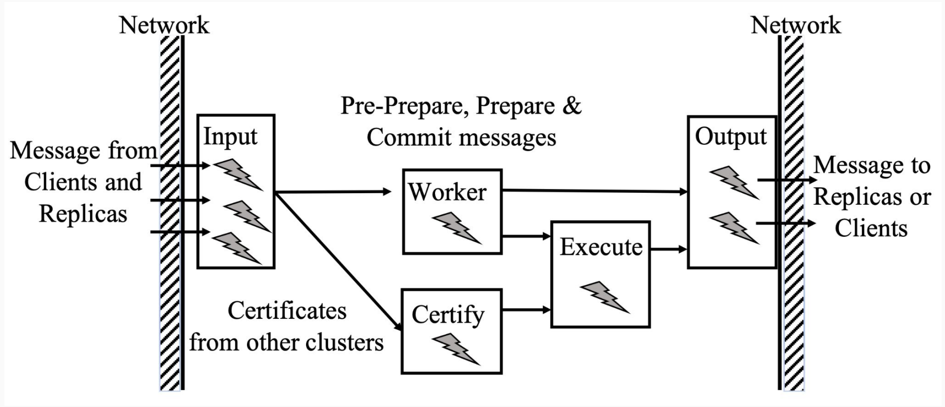
ResilientDB Fabric — Request Batching

- Request batching helps optimize consensus performance by grouping multiple client requests into a single batch.
- Clients can:
    - Send batches of requests to their local cluster.
    - Local primaries can aggregate multiple client requests into a single batch.
- Consensus processing optimizations:
    - Instead of handling each request individually, GeoBFT processes a batch as a single request.
    - This reduces the overhead and shares consensus costs among multiple requests.

Multithreaded Implementation of GeoBFT


Network
Client Requests
Batching
Prepare & Commit messages
Network
Message from Clients and Replicas
Input
Worker
Output
Message to Replicas or Clients
Execute
Certificates from other clusters
Certify

Implementation for local primaries


Network
Network
Pre-Prepare, Prepare & Commit messages
Message from Clients and Replicas
Input
Worker
Output
Message to Replicas or Clients
Execute
Certificates from other clusters
Certify

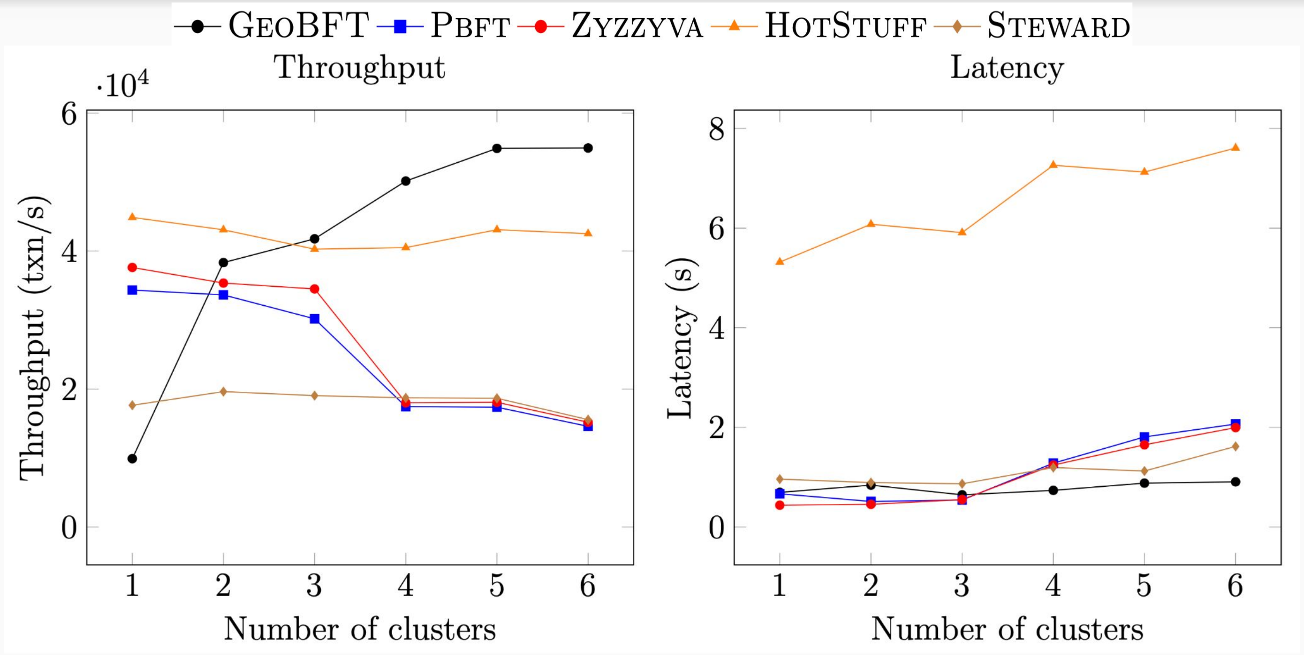
Implementation for other replicas

Evaluation of GeoBFT

Impact of the Number of Clusters

Increasing the number of clusters improves GeoBFT's throughput significantly.

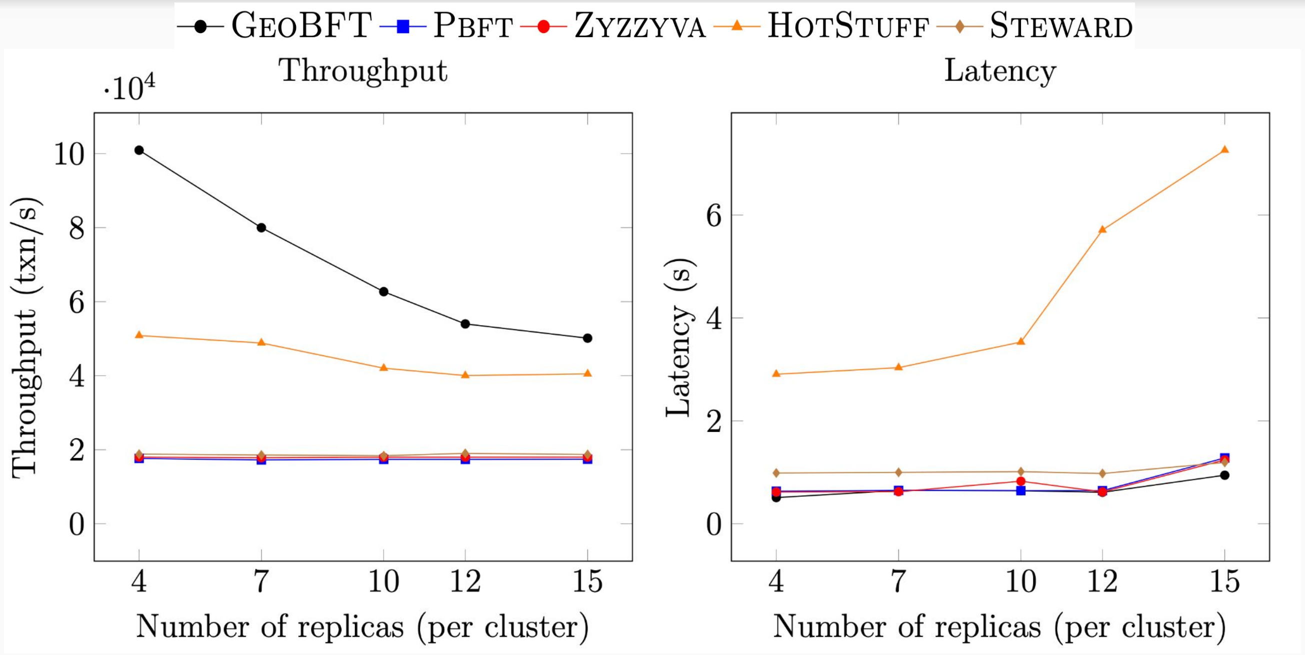
GeoBFT scales rather well with more clusters, while its parallel consensus execution reduces bottlenecks.



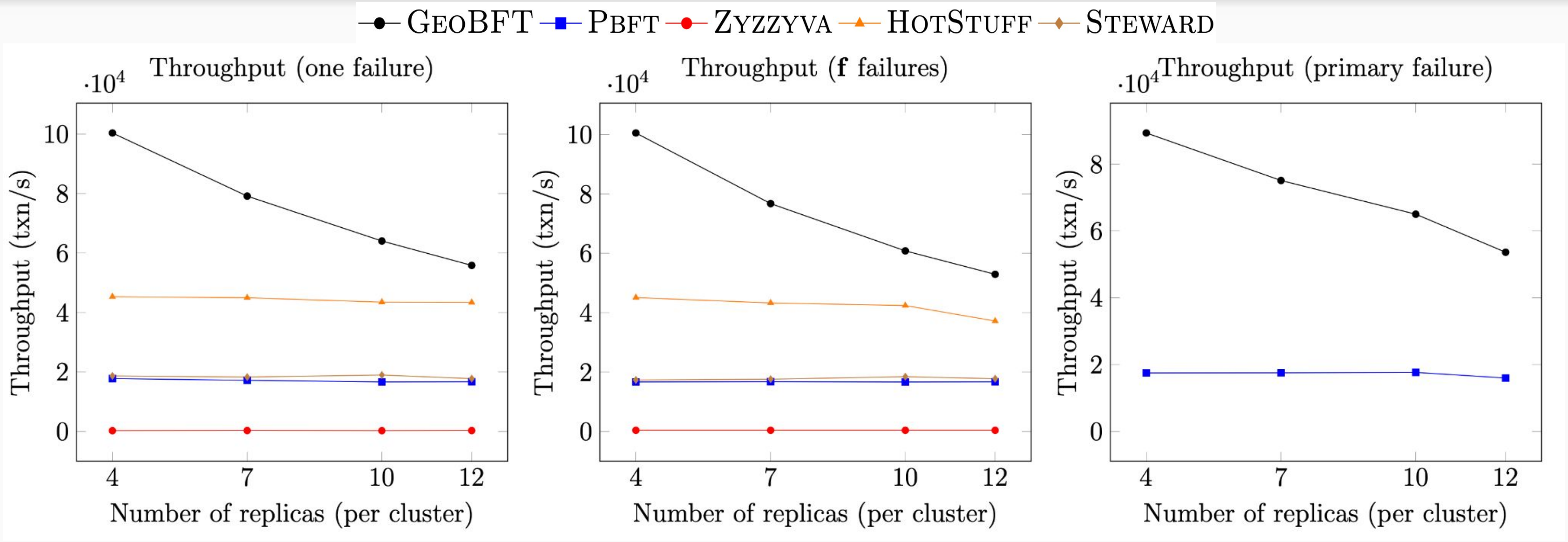
Impact of the Number of Replicas per Cluster

Increasing the number replicas per cluster decreases GeoBFT's throughput significantly.

GeoBFT is sensitive to the cluster size. Adding too many replicas per cluster recreates the same issues as PBFT. The bandwidth of the primary becomes the bottleneck



Impact of Failures

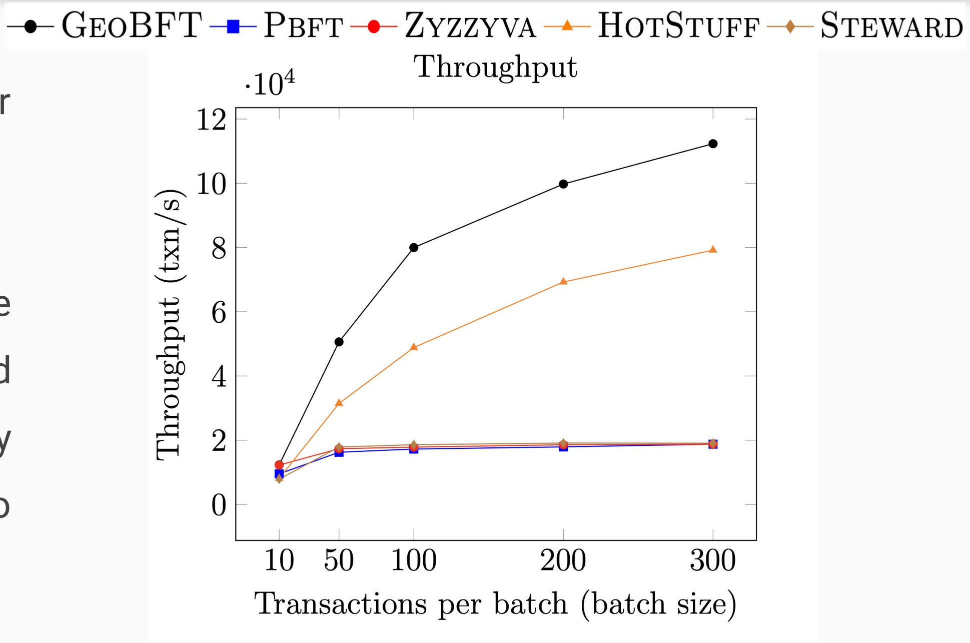


GeoBFT is resilient to moderate failures but supports fewer failures than other BFT protocols with the same number of replicas.

Impact of Increasing Batch Size

Larger batch sizes result in a larger throughput a certain point.

The appropriate batch size can be selected depending on the desired efficiency vs throughput/latency tradeoffs the user is willing to make.



Critical Analysis — Strong Points

- ➔ Outperforms all other protocols on a global scale.
- ➔ Decentralized → Global consensus shared between primaries.
- ➔ Easy to scale → New replicas are added to an existing/new cluster.
- ➔ Reduced global communication overhead.

Critical Analysis — Weak Points

- GeoBFT supports fewer failures than other protocols with the same number of replicas.
- More complex → GeoBFT requires the implementation of a global communication protocol.
- Sensitive to cluster size → Throughput decreases as the number of replicas per cluster increases.

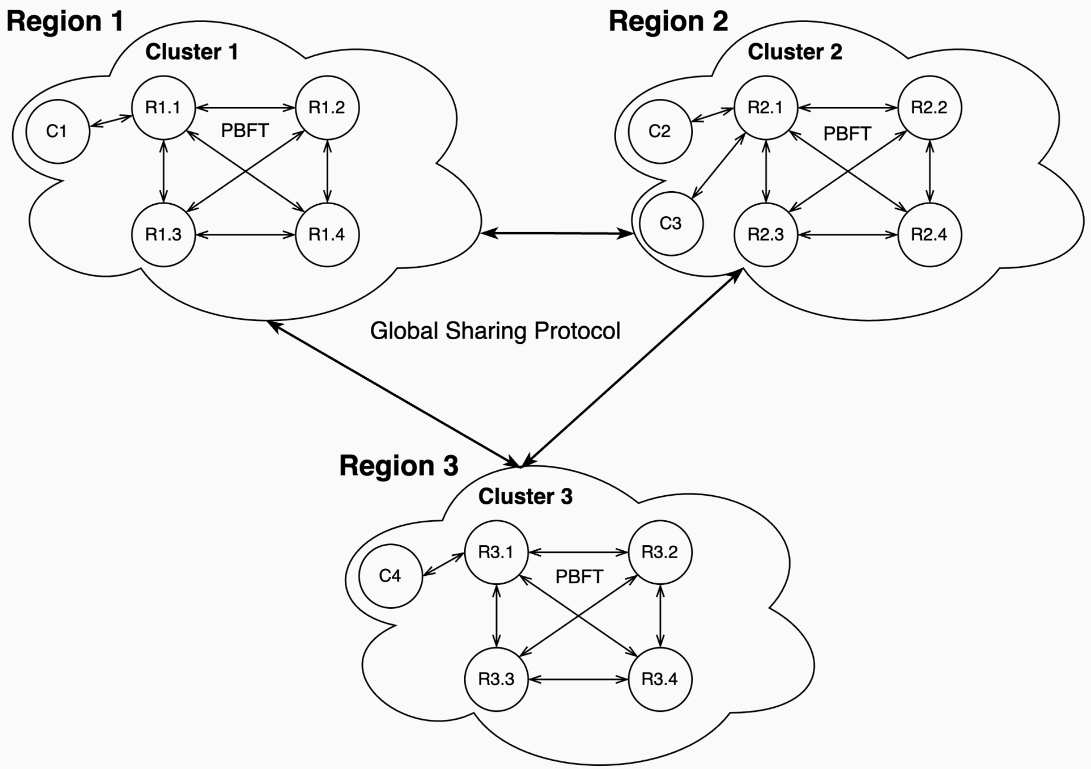
Summary

GeoBFT is a scalable, cluster-based Byzantine Fault Tolerant (BFT) consensus protocol designed for geo-distributed blockchains. It improves efficiency, scalability, and resilience over traditional BFT protocols by:

- Grouping replicas into clusters to minimize global communication.
- Introducing an optimistic global sharing protocol to reduce overhead.
- Adding a remote view-change mechanism to ensure fault tolerance.

Thank you!

Any questions?



References

Information and figures taken from

S. Gupta, S. Rahnama, J. Hellings, and M. Sadoghi, “ResilientDB: Global scale resilient blockchain fabric,” Proc. VLDB Endow., vol. 13, no. 6, pp. 868–883, Feb. 2020, doi: 10.14778/3380750.3380757.