ResilientDB: Global Scale Resilient Blockchain Fabric*

Suyash Gupta

Sajjad Rahnama

Jelle Hellings

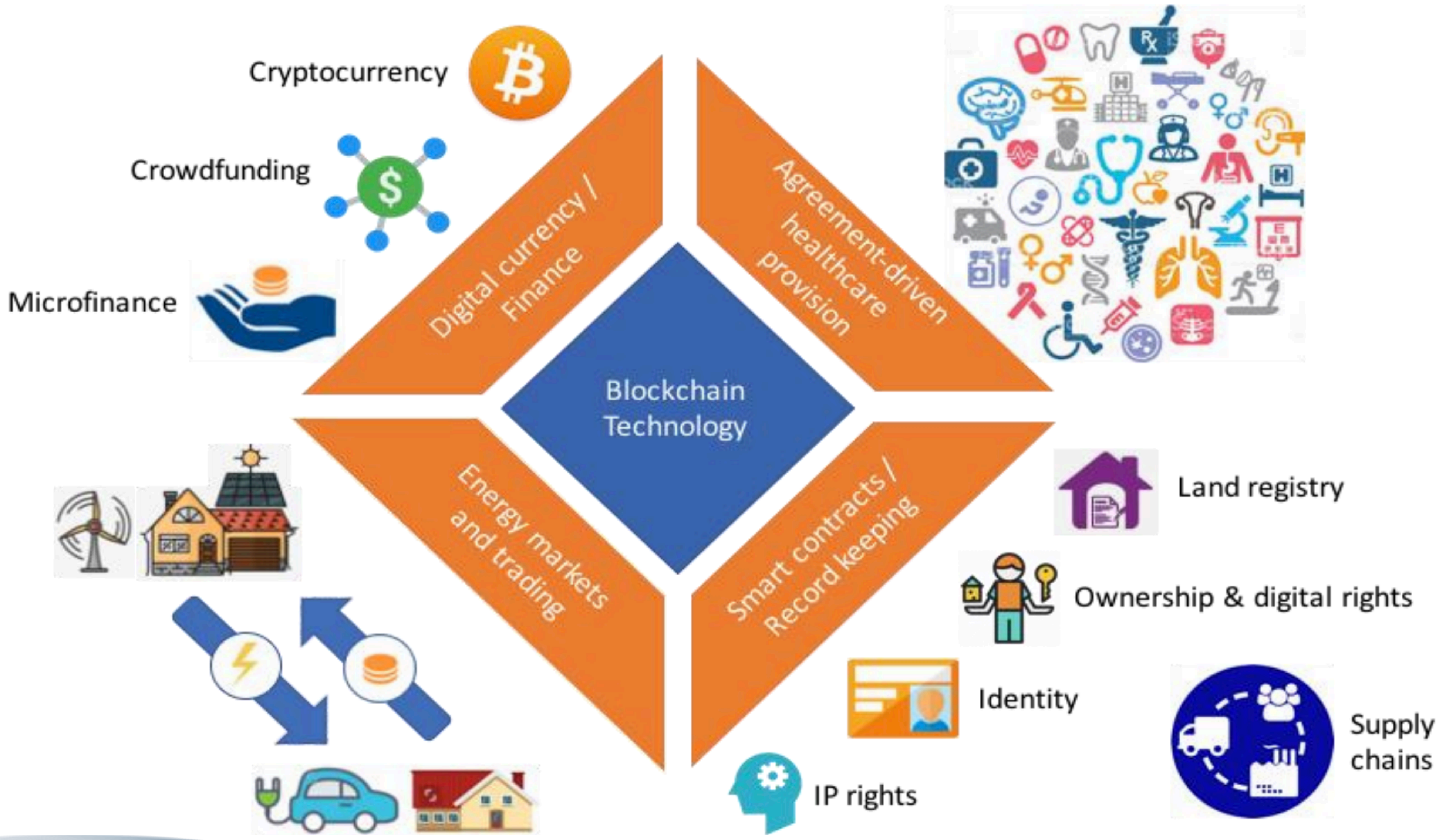
Mohammad Sadoghi

Exploratory Systems Lab

University of California Davis

ExpoLab Creativity Unfolded ResilientDB UCDAVIS UNIVERSITY OF CALIFORNIA

*Proceedings of the VLDB Endowment, Vol. 13, No. 6.



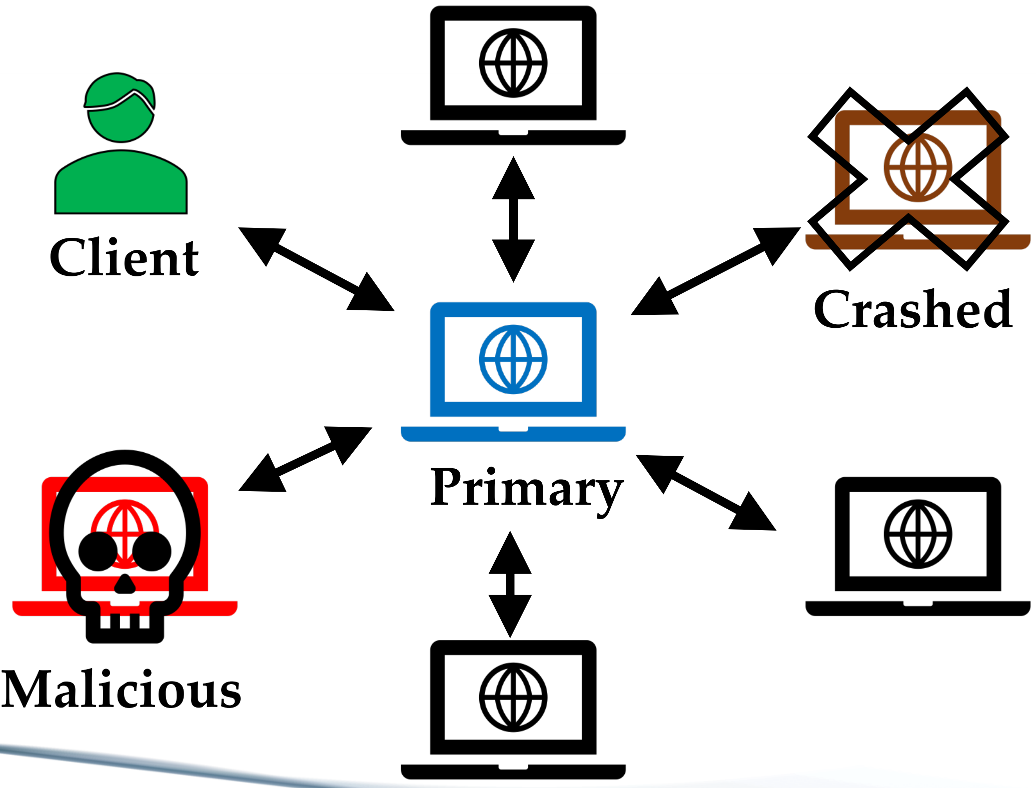
- Jacobsen et al., Blockchain Landscape and AI Renaissance: The Bright Path Forward. Middleware (Tutorials) 2018: 2:1

Types of Blockchain Systems

- **Permissionless → Open Access**
  - Anyone can participate.
  - Identities of the replicas unknown.
  - Face blockchain *forks*.
- **Permissioned (Our focus) → Restricted Access**
  - Only a selected group of replicas, although untrusted can participate.
  - Identities of the replica known a priori.

ResilientDB ExpoLab Creativity Unfolded

At the core of *any* Blockchain application is a Byzantine Fault-Tolerant (BFT) consensus protocol.



Challenges For Geo-Scale Blockchains

	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

Real-world inter- and intra-cluster communication costs in terms of the ping round-trip times (which determines latency) and bandwidth (which determines throughput). Measurements taken on Google Cloud using clusters of n1 machines (replicas) that are deployed in six different regions.

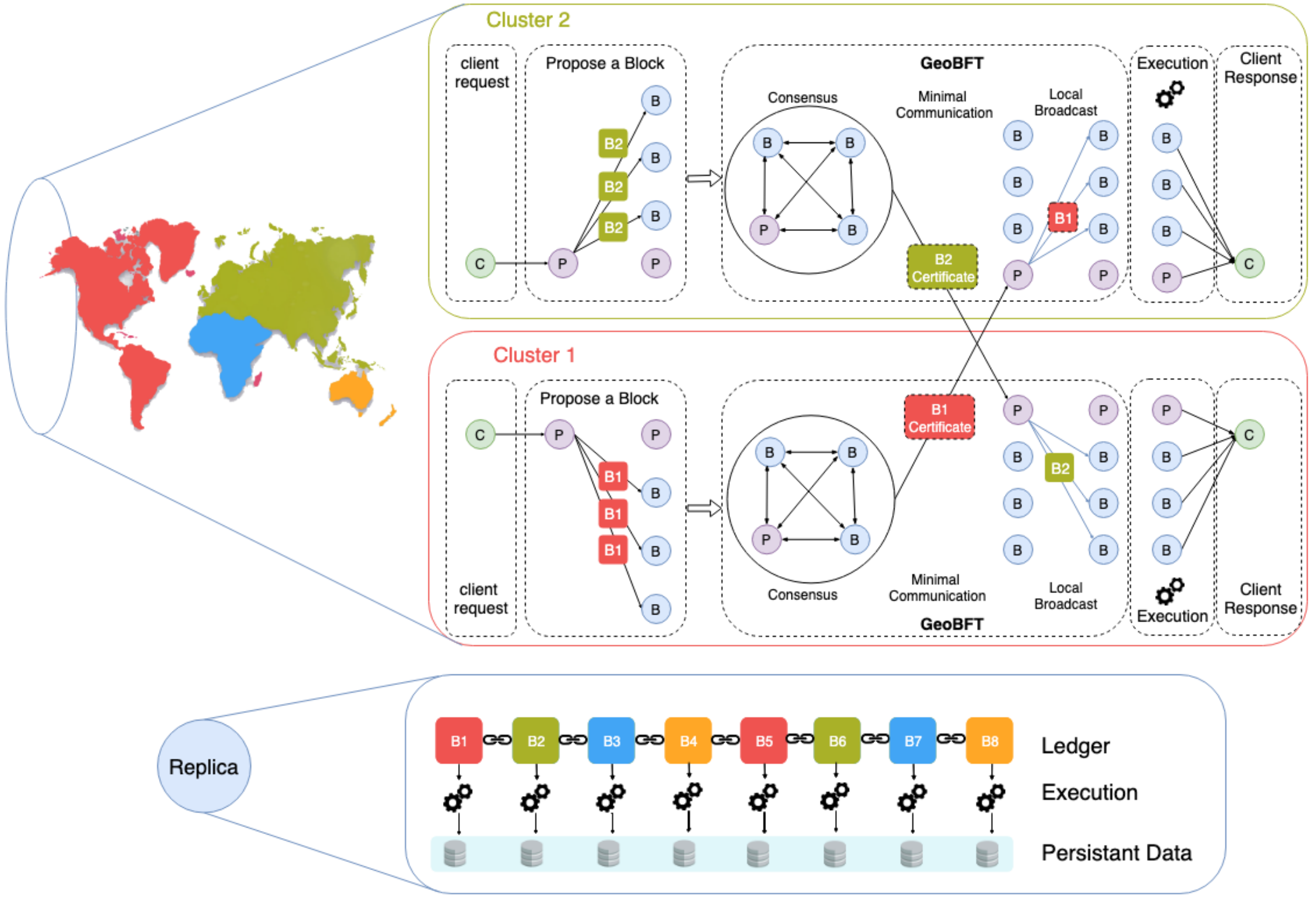
ResilientDB ExpoLab Creativity Unfolded

Limitations of Existing Consensus Protocols

Protocol	Decisions	Communication (Local)	Communication (Global)	Centralized
GeoBFT (our paper)	$\mathbf{z}$	$\mathcal{O}(2\mathbf{zn}^2)$	$\mathcal{O}(\mathbf{fz}^2)$	No
↳ *single decision*	1	$\mathcal{O}(4\mathbf{n}^2)$	$\mathcal{O}(\mathbf{fz})$	No
Steward	1	$\mathcal{O}(2\mathbf{zn}^2)$	$\mathcal{O}(\mathbf{z}^2)$	Yes
Zyzzyva	1	$\mathcal{O}(\mathbf{zn})$		Yes
Pbft	1	$\mathcal{O}(2(\mathbf{zn})^2)$		Yes
PoE	1	$\mathcal{O}((\mathbf{zn})^2)$		Yes
HotStuff	1	$\mathcal{O}(8(\mathbf{zn}))$		Partly

The normal-case metrics of BFT consensus protocols in a system with z clusters, each with n replicas of which at most f , n > 3f , are Byzantine. GeoBFT provides the lowest global communication cost per consensus decision and operates decentralized.

Vision Geo-Scale Byzantine Fault-Tolerance



GeoBFT Protocol

GeoBFT is a topology-aware protocol, which groups replicas into clusters. Each cluster runs the PBFT consensus protocol, in parallel and independently.

Local Replication

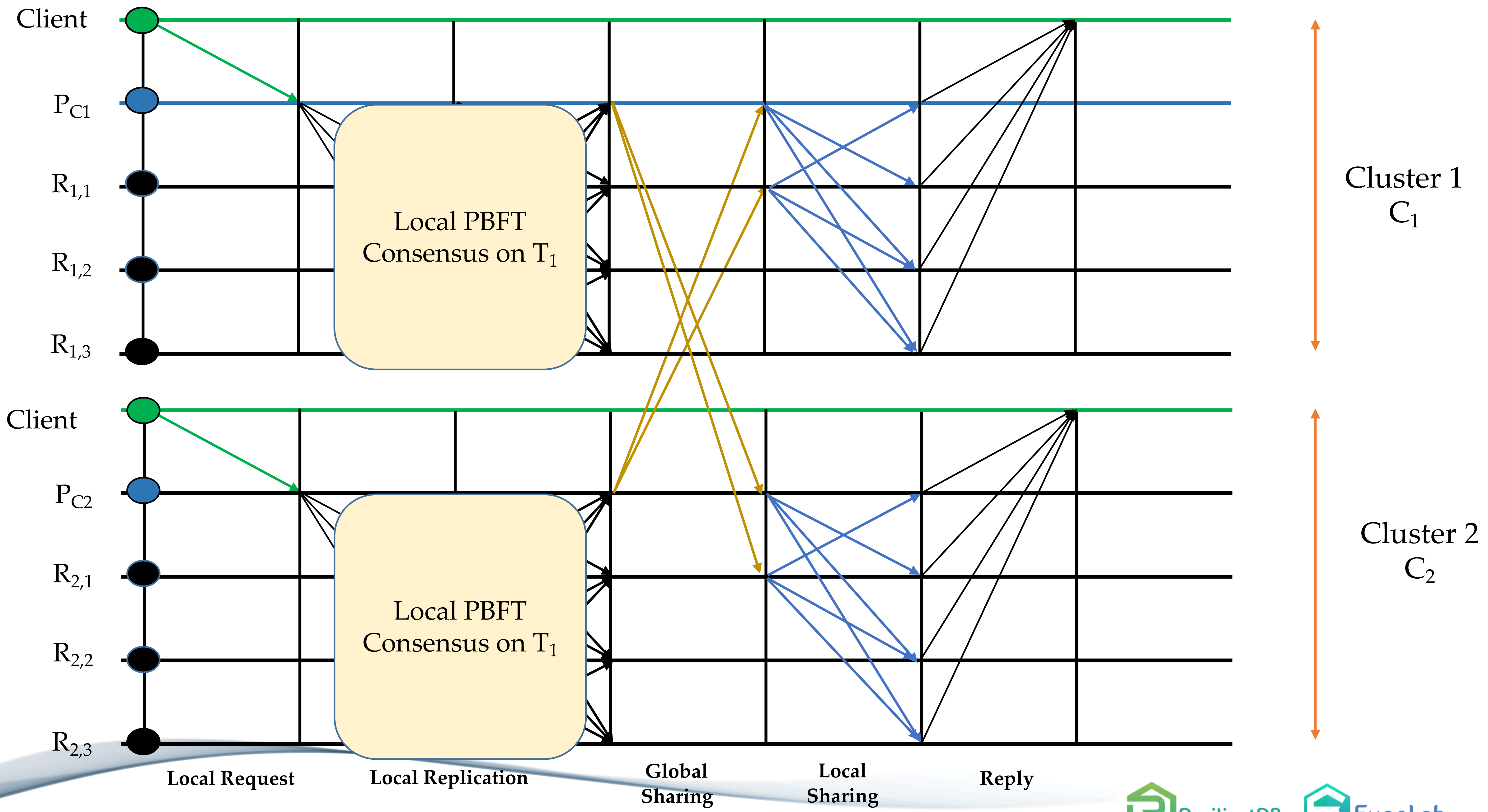
Each cluster runs PBFT to select, locally replicate, and certify a client request.

Inter-cluster Sharing

Primary at each cluster shares the certified client request with other clusters.

Ordering and Execution

Order the certified requests, execute them, and inform local clients.

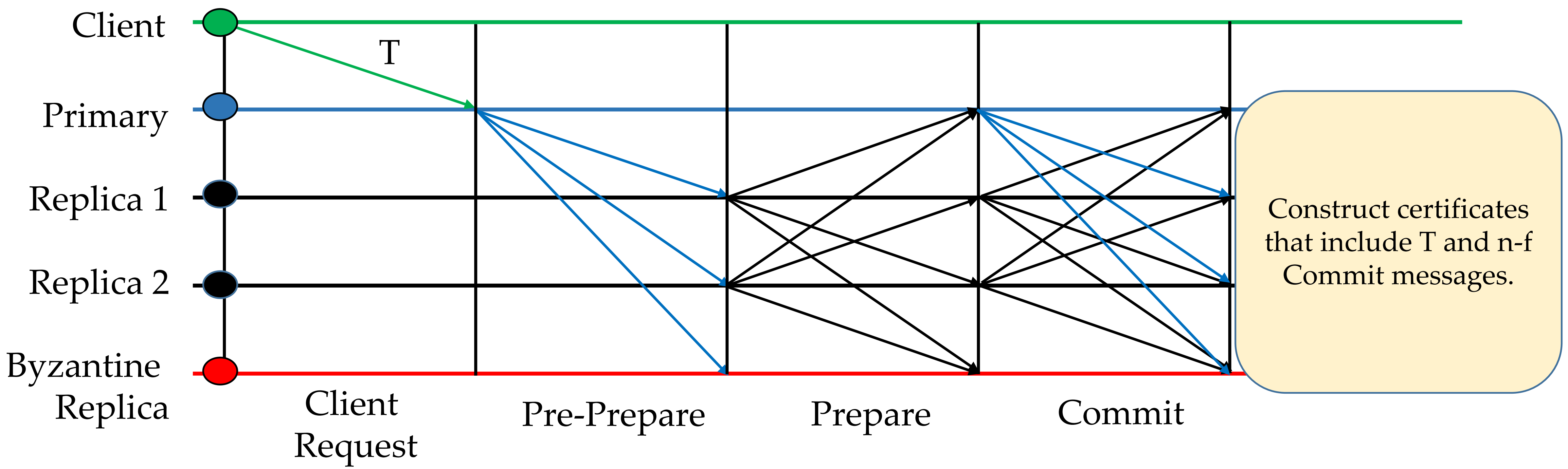


ResilientDB ExpoLab Creativity Unfolded

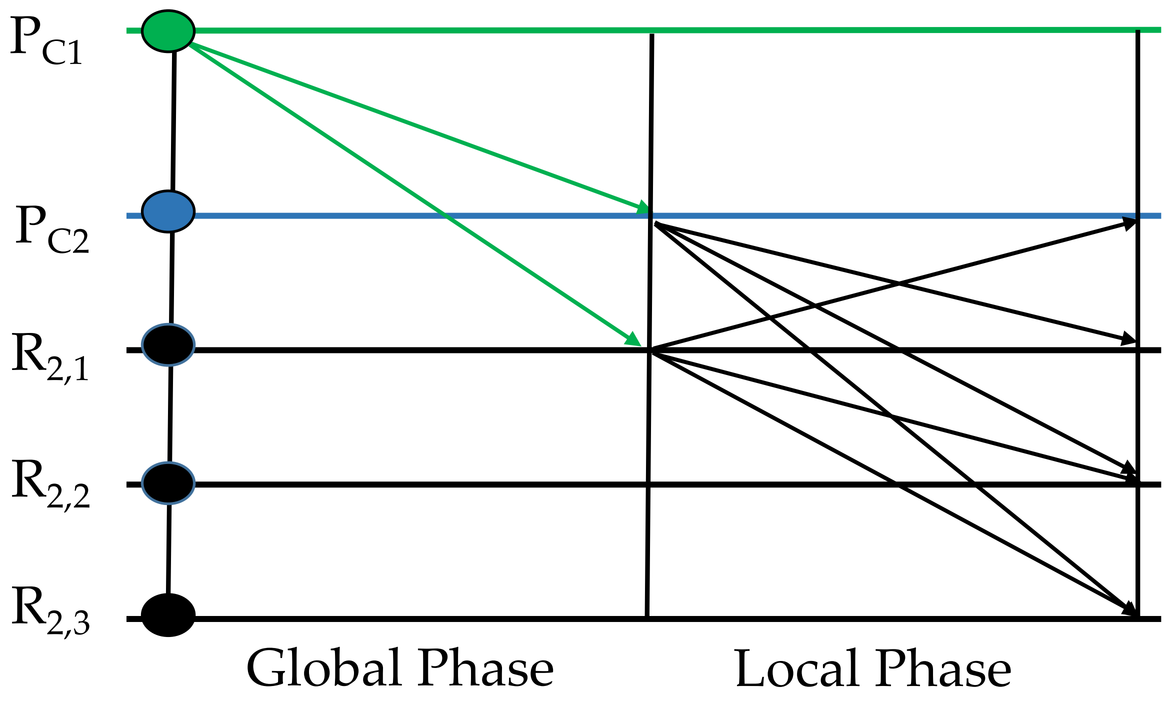
Local Replication (PBFT)

- First practical Byzantine Fault Tolerant Protocol.
- Tolerates up to **f** failure out of 3f+1 replicas
- Three phases of which two require quadratic communication complexity.
- Safety is always guaranteed and Liveness is guaranteed in periods of partial synchrony.
- View-Change protocol for replacing malicious primary

PBFT Civil Execution



Inter-Cluster Sharing

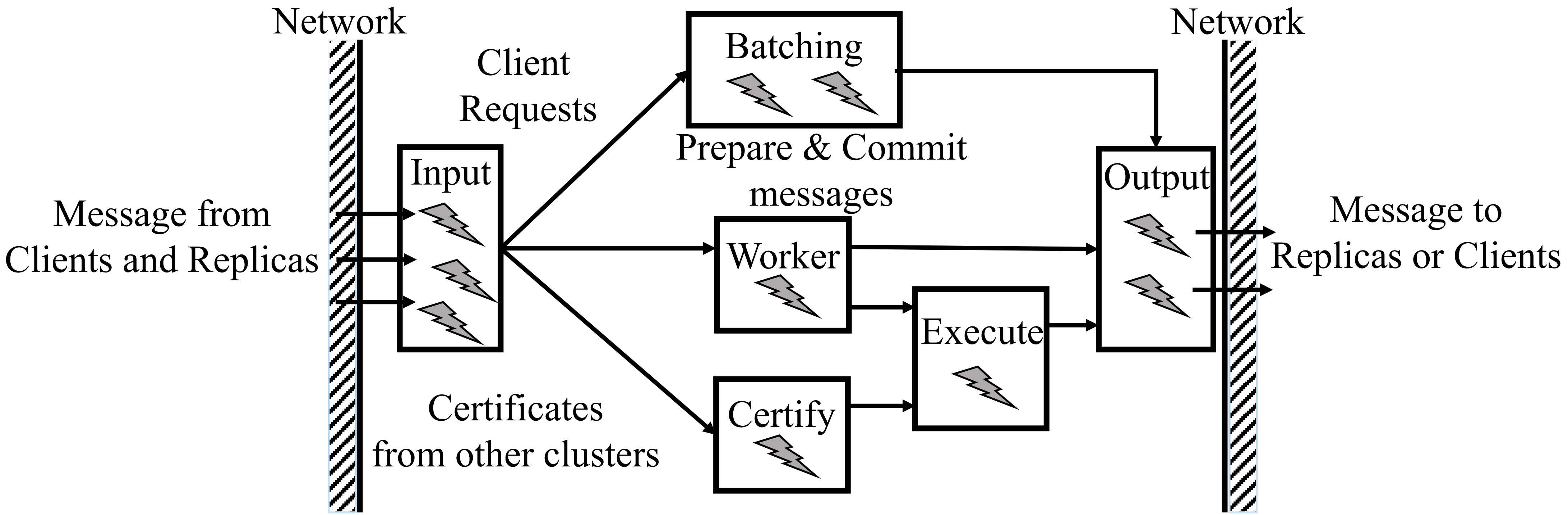


The Primary P_{C1} sends a certificate that includes the client request and commit messages from n-f replicas of Cluster C_1 .

Ordering and Execution

- GeoBFT orders requests deterministically.
- For $i < j$, requests of Cluster C_i are executed before requests of cluster C_j.
- For example: requests of C_1 are executed before C_2 .

Implementation on ResilientDB



ResilientDB associates a multi-threaded deep-pipelined architecture with each replica.

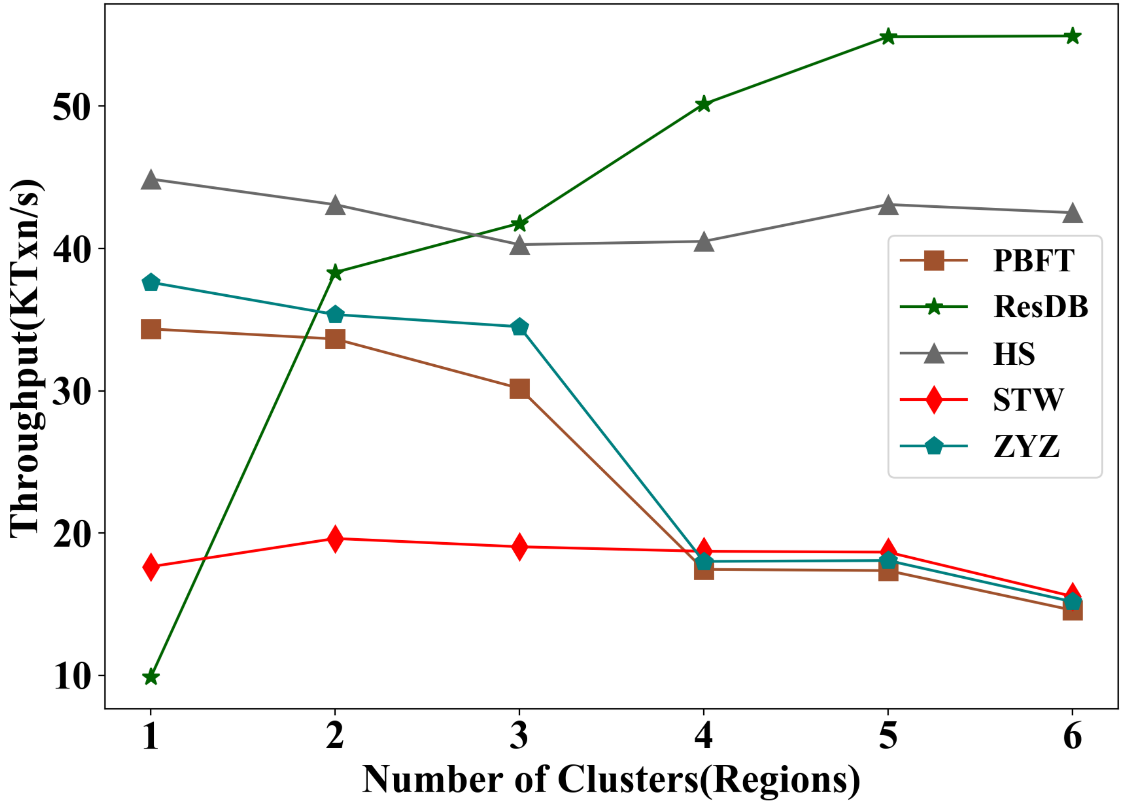
Ledger (Blockchain) Management

- In ResilientDB, i^{th} block in the ledger contains the i^{th} executed request.
- In each round of GeoBFT, each replica executes z requests, each belonging to a different cluster C_i , $1 <= i <= z$.
- Hence, in each round, each replica creates z blocks.
- To ensure immutability, each block includes both client requests and exchanged certificates.

Evaluation on ResilientDB

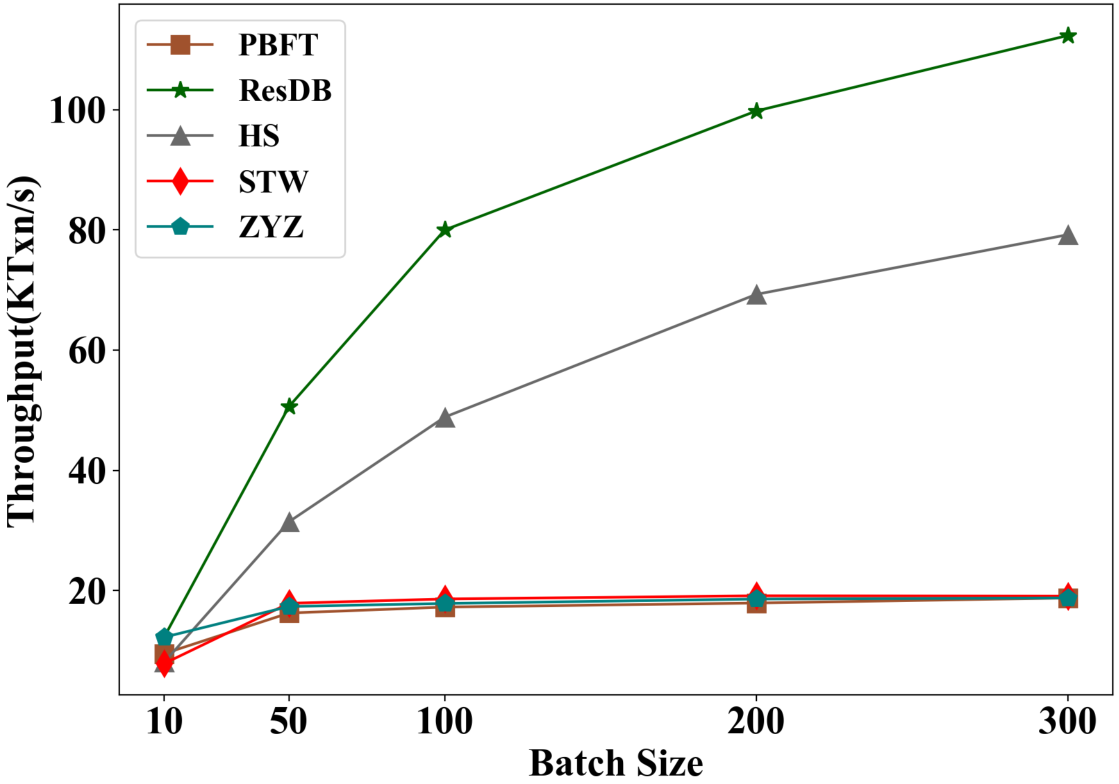
- Google cloud used for deploying replicas and clients.
- Each replica used 8-core Intel Skylake CPUs and had access to 16 GB memory.
- Total 160K clients deployed on eight 4-core machines.
- Workload provided by Yahoo Cloud Serving Benchmark (YCSB).
- Replicas deployed across six different regions: Oregon, Iowa, Montreal, Belgium, Taiwan and Sydney.
- Primaries for centralized protocol placed at Oregon (highest bandwidth).

Impact of Geo-Scale Deployments



Throughput as a function of the number of clusters; zn = 60 replicas.

Impact of Request Batching



Throughput as a function of the batch size; z = 4 and n = 7.

Conclusions and Final Remarks

- For achieving faster local replication, other efficient BFT protocols, such as PoE, can be employed.
- Modern cryptographic techniques such as Threshold signatures can be used in place of sending n-f Commit messages.
- If a cluster does not have a request, it can send “no-op” messages.
- GeoBFT optimizes consensus by reducing global communication costs.
- Parallel local replication helps to increase system throughput.
- GeoBFT is a topology-aware protocol.

References

1. S. Gupta, S. Rahnama, J. Hellings, and M. Sadoghi. ResilientDB: Global Scale Resilient Blockchain Fabric. Proc. VLDB Endow., 13(6):868–883, Feb. 2020.
2. Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance. In Proceedings of the Third Symposium on Operating Systems Design and Implementation, pages 173–186. USENIX Association, 1999.
3. Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance and proactive recovery. ACM Transactions on Computer Systems, 20(4):398–461, 2002. doi:10.1145/571637.571640.
4. M. Yin, D. Malkhi, M. K. Reiter, G. G. Gueta, and I. Abraham. HotStuff: BFT consensus with linearity and responsiveness. In Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, PODC, pages 347–356. ACM, 2019.
5. R. Kotla, L. Alvisi, M. Dahlin, A. Clement, and E. Wong. Zyzzyva: Speculative byzantine fault tolerance. In Proceedings of Twenty-first ACM SIGOPS Symposium on Operating Systems Principles, SOSP, pages 45–58. ACM, 2007.
6. Yair Amir, Claudiu Danilov, Danny Dolev, Jonathan Kirsch, John Lane, Cristina Nita-Rotaru, Josh Olsen, and David Zage. Steward: Scaling byzantine fault-tolerant replication to wide area networks. IEEE Transactions on Dependable and Secure Computing, 7(1):80–93, 2010. doi:10.1109/TDSC.2008.53.
7. S. Gupta, J. Hellings, S. Rahnama, and M. Sadoghi, Proof-of-Execution: Reaching Consensus through Fault-Tolerant Speculation, CoRR, vol. abs/1911.00838, 2019.