Ziziphus: Scalable Data Management Across Byzantine Edge Servers

Mohammad Javad Amiri[1] Daniel Shu[2] Sujaya Maiyya[3] Divy Agrawal[2] Amr El Abbadi[2]

[1]University of Pennsylvania, [2]University of California Santa Barbara, [3]University of Waterloo

ICDE2023
Anaheim, CA April 3 – 7, 2023

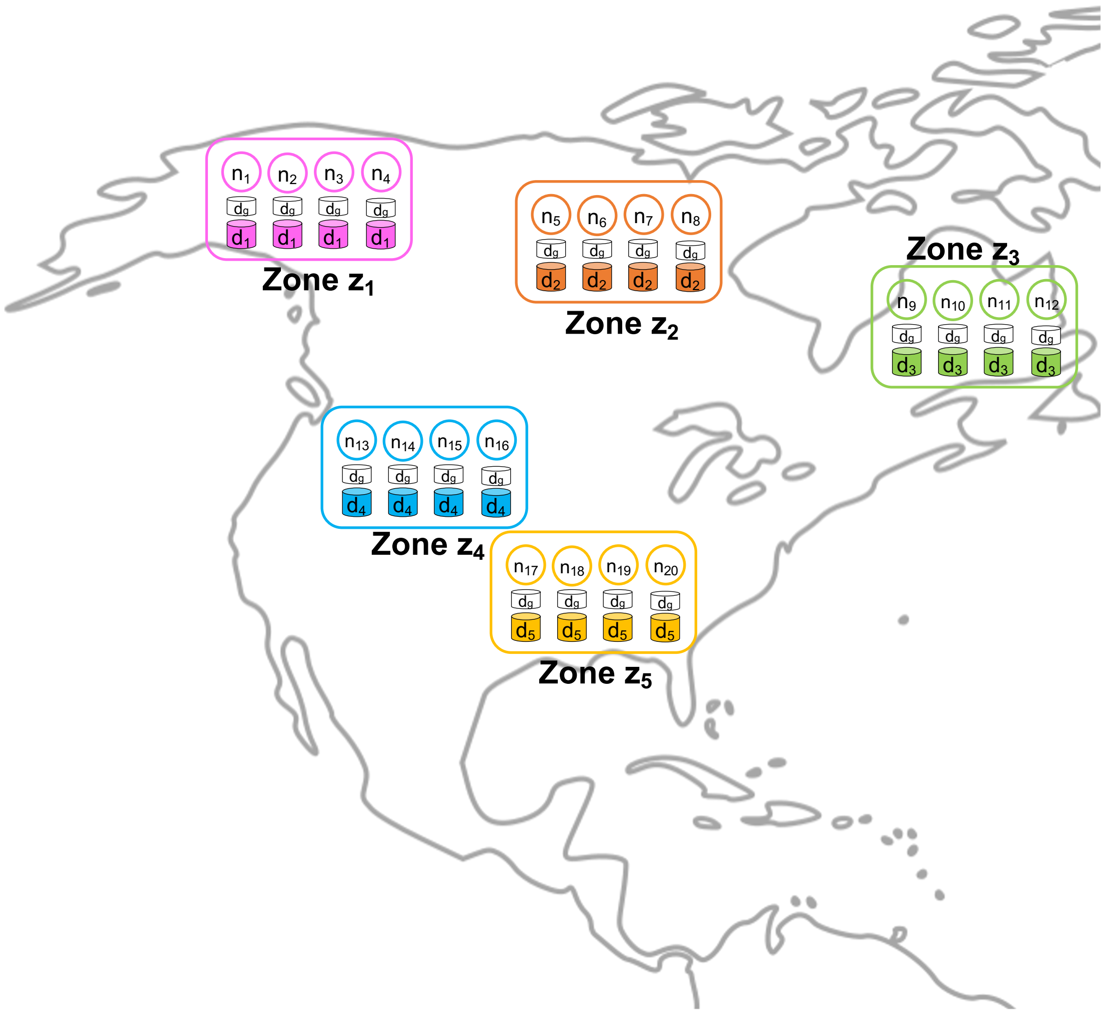
High complexity of BFT protocols in large-scale geo-distributed systems

Large-scale geo-distributed systems over wide-area

- Hierarchical fully replicated (e.g., Steward [DSN’06], Blockplane [ICDE’19])
  - Fully replicated data across multiple Byzantine fault-tolerant clusters
  - A CFT protocol is used to establish global consensus among clusters
  - The maliciousness of servers is confined within clusters
    - Every single transaction needs to be globally synchronized

- Sharded partially replicated (e.g., Caper [VLDB’19], Qanaat [VLDB’22])
  - Shard data and replicate a data shard on each cluster
  - Do not run a global consensus for every transaction
    - Use BFT protocols for global consensus

Ziziphus



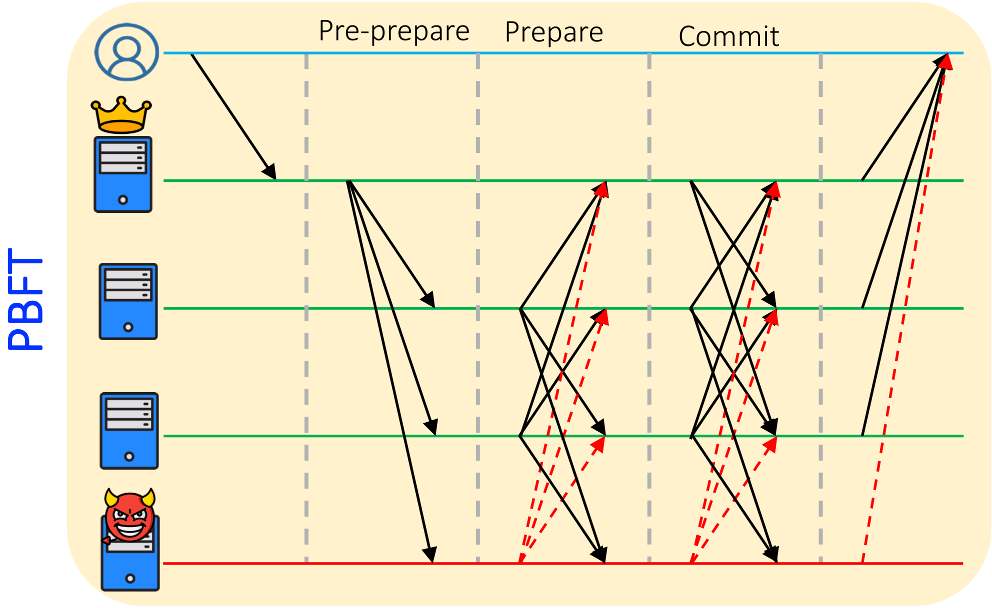
- A geo-distributed system to support edge applications with possibly mobile edge clients
- Edge nodes are partitioned into Byzantine fault-tolerant zones
- **Local level**: each zone processes local transactions initiated by its nearby clients independent of other zones.
- Nodes maintain global system meta-data
  - To enforce network-wide policies
  - E.g., a zone cannot host more than 10000 clients
- **Global synchronization** is only needed to update system meta-data
  - E.g., when a client migrates to another zone
- Confines the maliciousness of Byzantine servers

System model

- Target applications:
  - Performance in terms of throughput and latency is paramount
  - Data accesses have an affinity towards locality
  - The probability of failure of an entire zone is insignificant
- Design decisions:
  - Clustering nodes into fault-tolerant zones
  - Replicating local transactions of edge devices only on nodes of their nearby zone
- Design trade-offs:
  - Scalability vs. security
    - Ziziphus is more prone to DoS attacks in comparison to a flat system with the same number of nodes
  - Performance vs. availability
    - The availability of Ziziphus is reduced if an entire zone fails, e.g., due to natural disasters

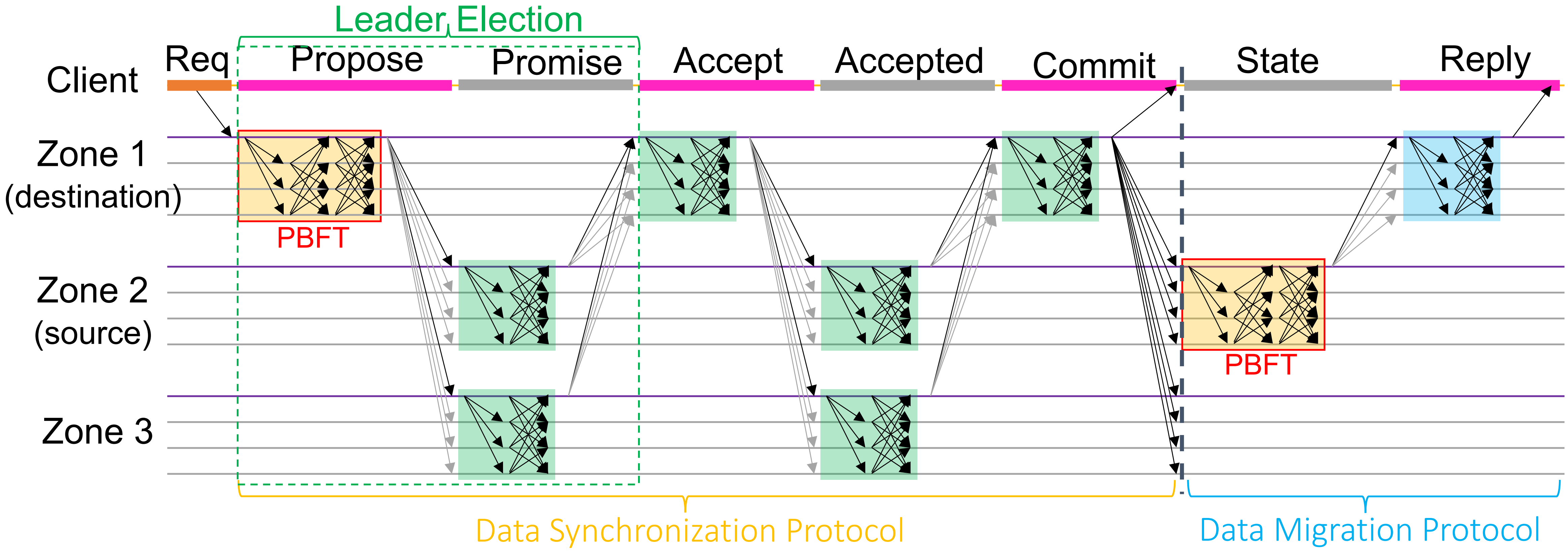
Local transactions

- Initiated by the clients of a zone on their local data in the zone
- Processes local transactions using PBFT (pluggable)



Global transactions

- Consists of two atomic sub-transactions [in case of client migration]
  - Update the global system meta-data of all zones (data synchronization protocol)
  - Copy the actual client data from the source to the destination zone (data migration protocol)

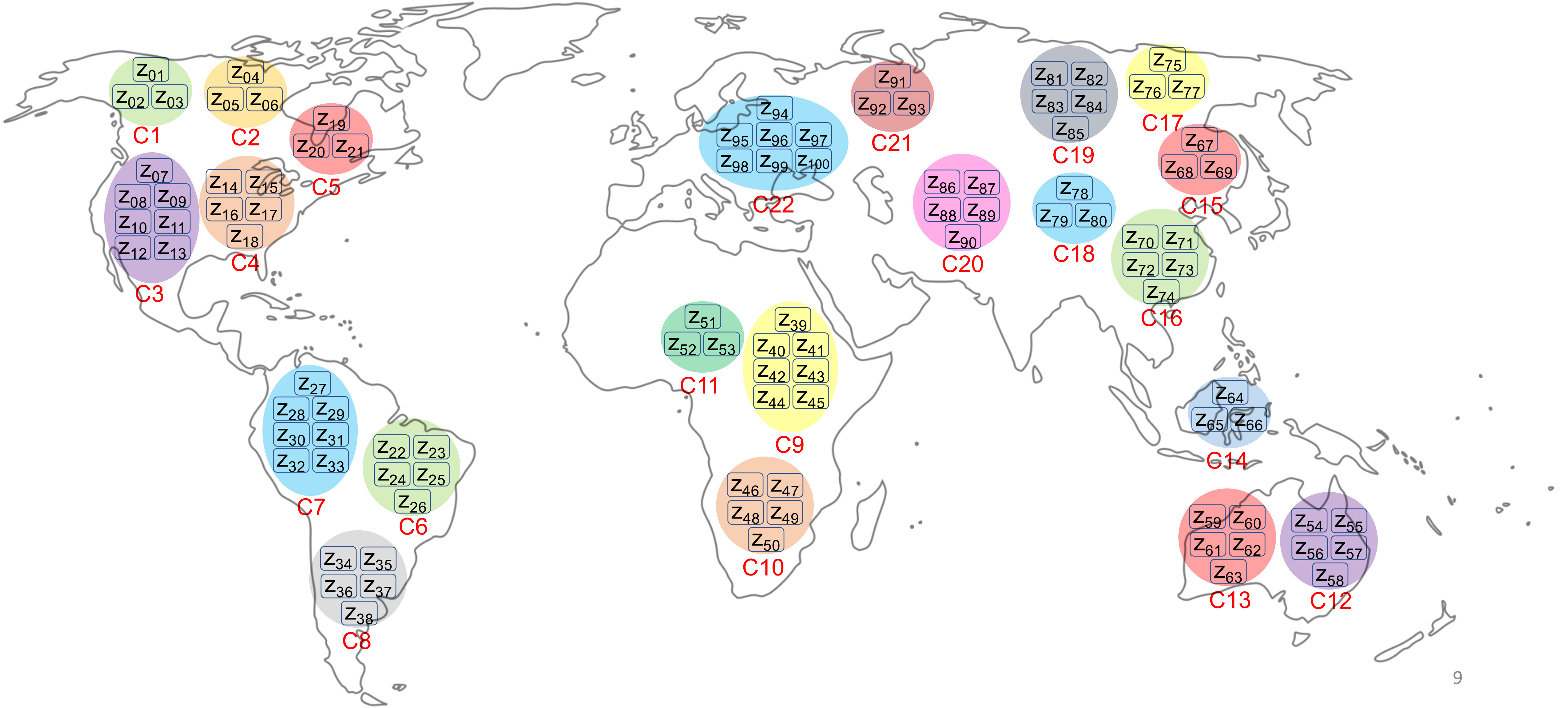


Lazy synchronization

- The zone data becomes unavailable during zone failures
- Ziziphus use checkpointing to provide (a weaker degree of) fault tolerance
  - Inspired by checkpointing mechanism of BFT protocols
- Nodes periodically share their latest stable states with each other
  - No need to run global synchronization for every transaction
- Each zone replicates the latest stable state of every zone on all its nodes
  - Stable state: all executed local transactions

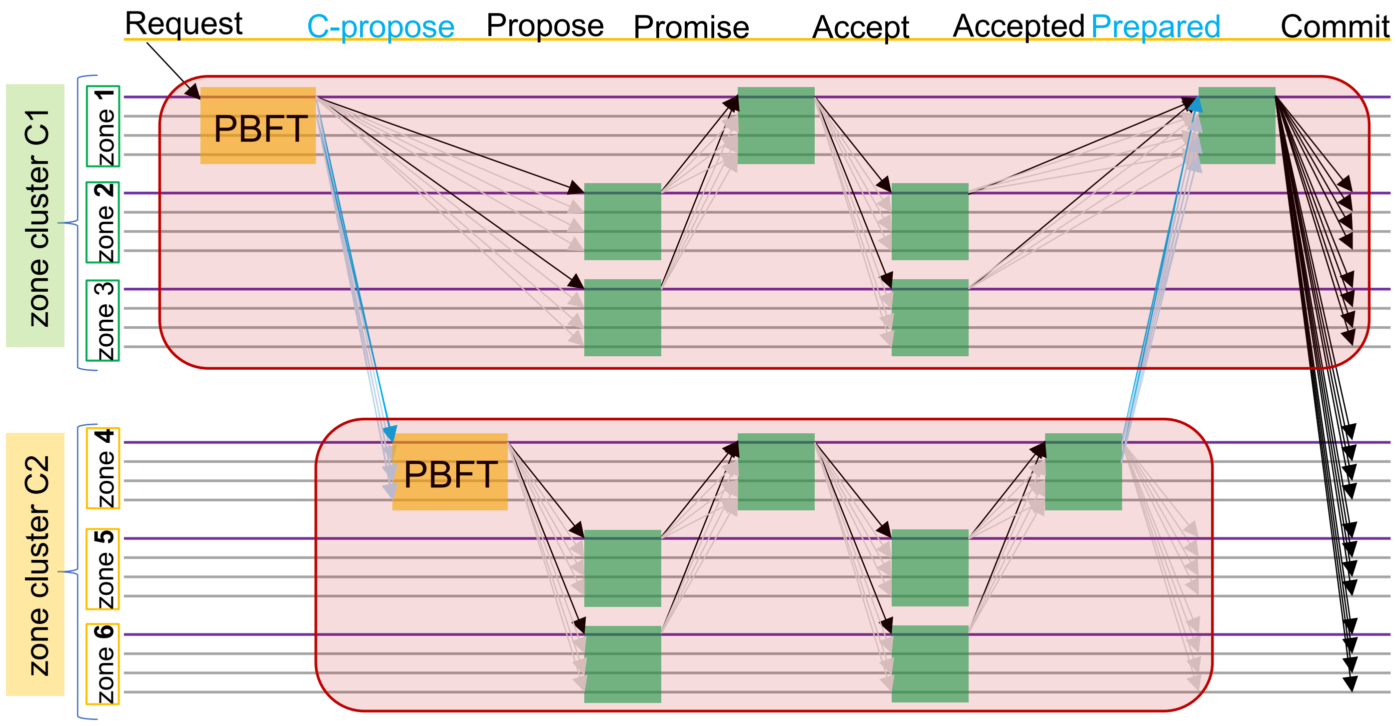
Zone clusters

(regional) system meta-data



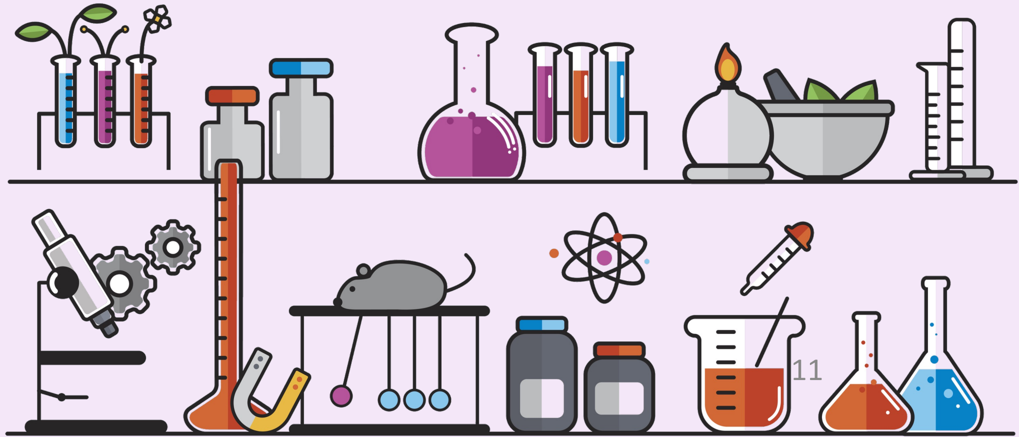
Data synchronization protocol

- Client migration within a zone cluster: data synchronization and migration protocols
- Client migration across zone clusters: cross-cluster data synchronization

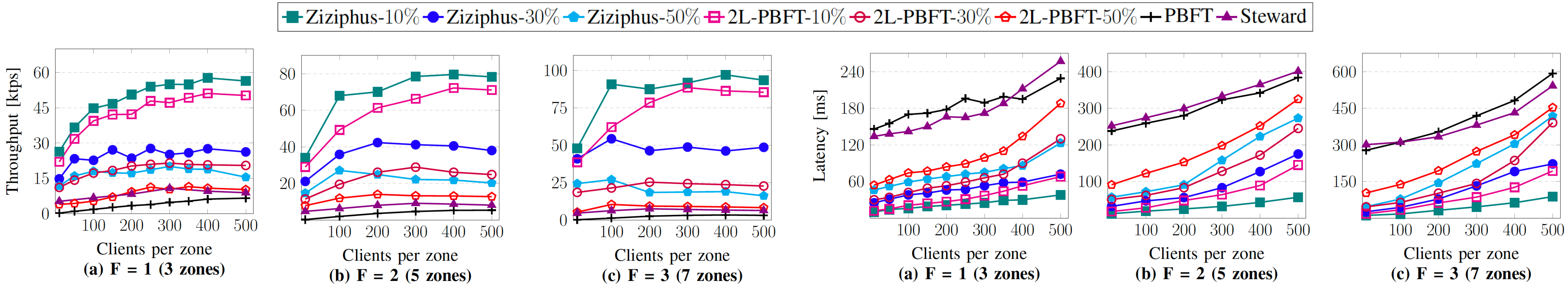


Experimental settings

- Platform: Amazon EC2
- Measuring performance
  - Throughput & Latency
- Application:
  - Banking
- Local transactions:
  - PBFT
- Systems:
  - Flat PBFT
  - Two-level PBFT
  - Steward (i.e., Ziziphus with 100% global transactions)
  - Ziziphus

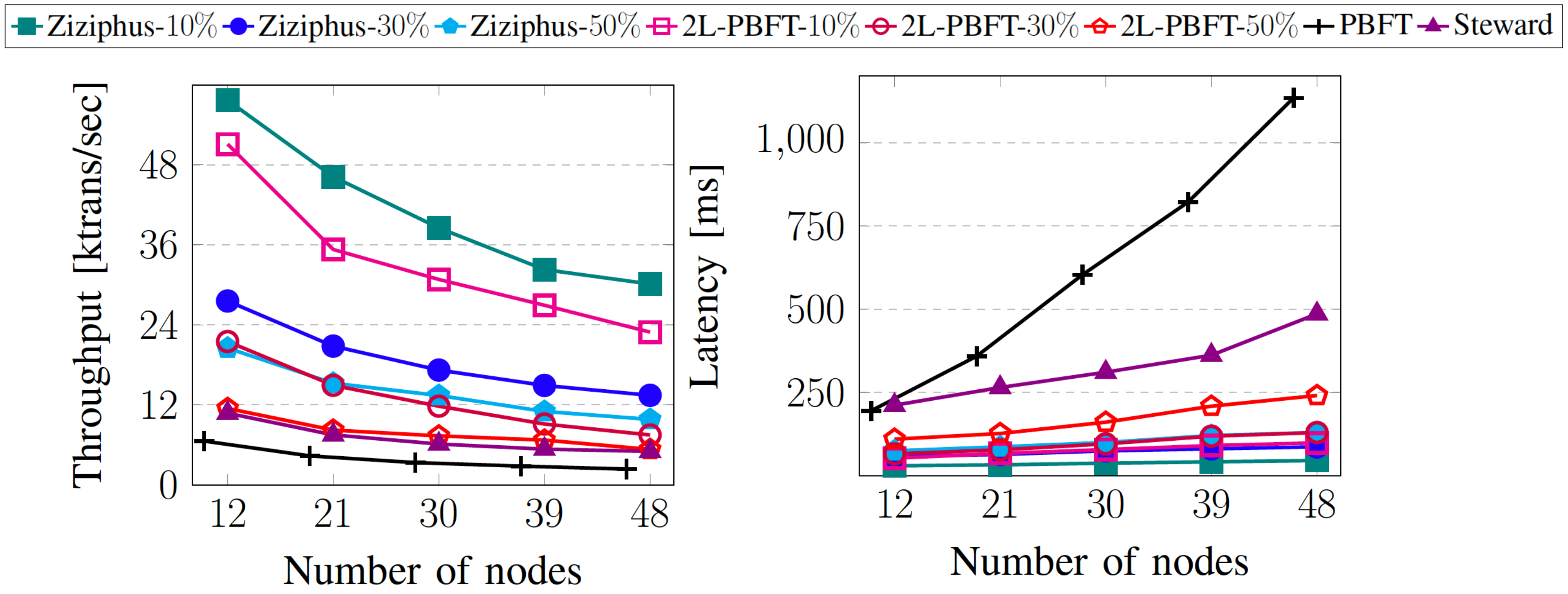


Performance with increasing the number of zones



- Ziziphus with 3 zones and 10% global transactions:
  - Processes 12% more transactions with 46% lower latency compared to two-level PBFT
  - Processes 470% more transactions with 86% lower latency compared to Steward

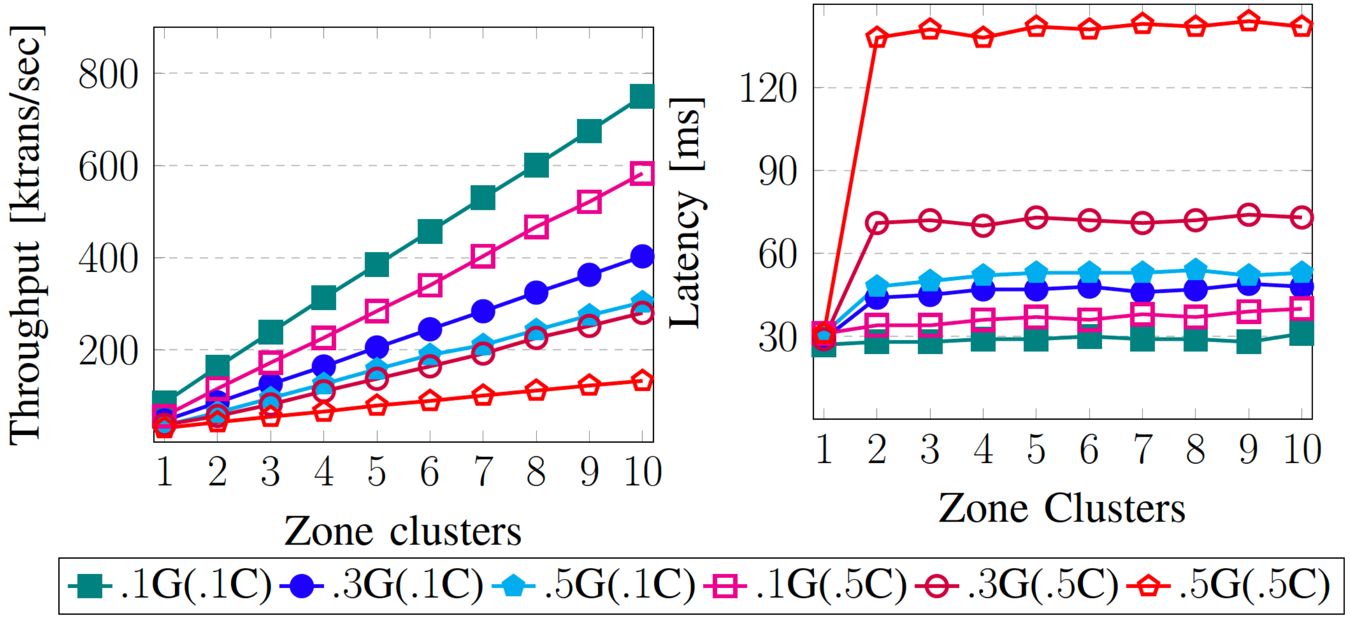
Fault tolerance scalability



Increasing f from 1 to 5, i.e., increasing zone size from 4 to 16.

- Increasing the number of nodes, reduces the overall throughput and increases latency of all protocols.
- With 10% global transaction and increasing the zone size from 4 to 16:
  - Ziziphus shows 53% higher latency
  - PBFT shows 480% higher latency

Scalability using zone clusters



Each cluster includes 3 zones and each zone contains 4 nodes

- Ziziphus demonstrates higher throughput by increasing the number of zone clusters
- Increasing the number of zone clusters does not affect latency
- With (10%G(10%C) Ziziphus is able to process 749 ktps with 31 ms latency with 10 zone clusters.

Ziziphus conclusion

- A geo-distributed system that partitions Byzantine edge servers into fault-tolerant zones.
- Global synchronization is needed when system meta-data needs to be updated.
- Provides a zonal abstraction to confine maliciousness of Byzantine servers within each zone.
- In typical workloads, Ziziphus achieves significantly better performance compared to flat PBFT, two-level PBFT and Steward.
- The performance of Ziziphus improves semi-linearly with increasing the number of zones or zone clusters.

Thank You!

Questions?

mjamiri@seas.upenn.edu