SharPer: Sharding Permissioned Blockchains Over Network Clusters

Mohammad Javad Amiri[1], Divyakant Agrawal[2], Amr El Abbadi[2]

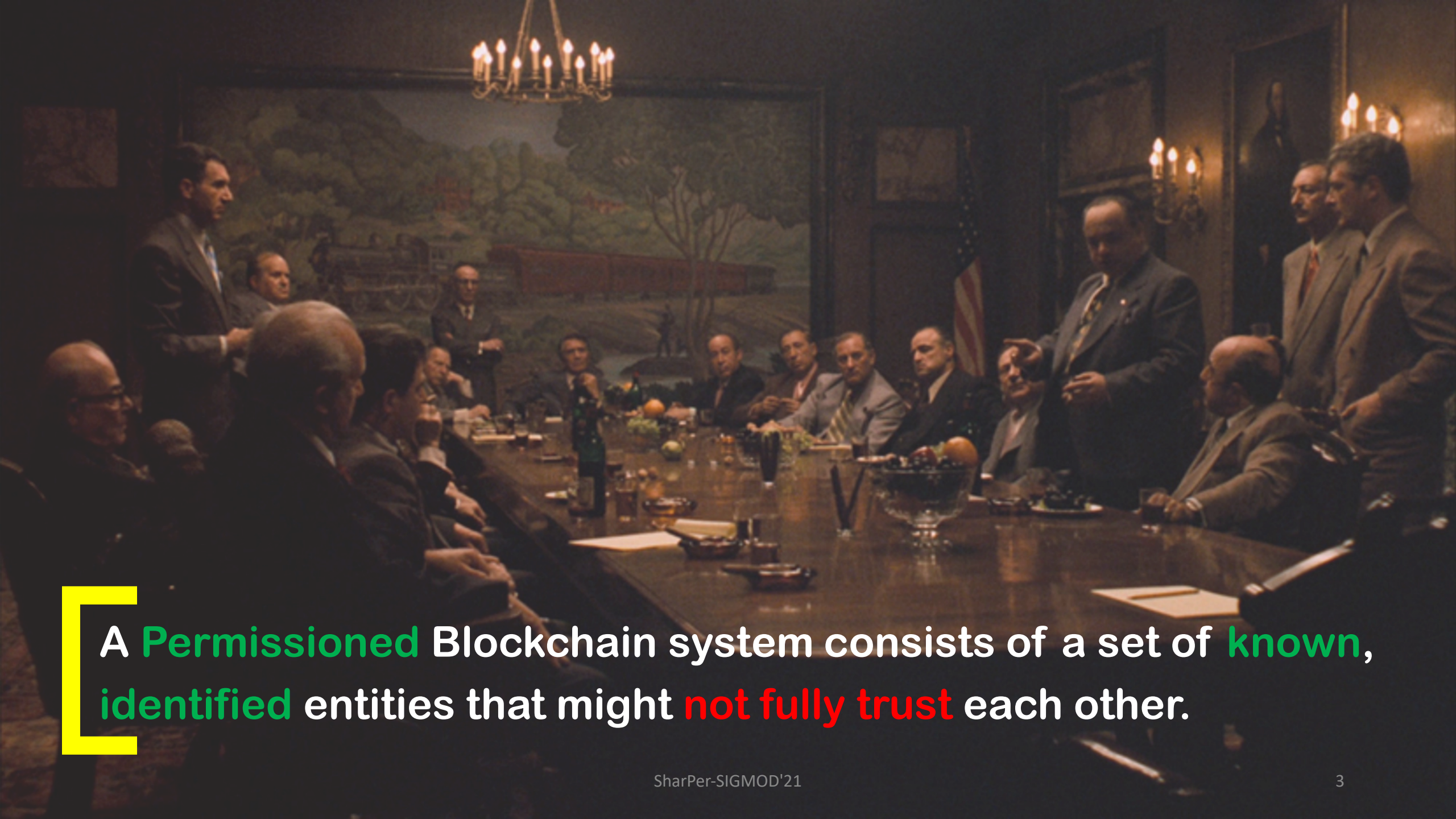
[1]University of Pennsylvania, [2]University of California Santa Barbara

Anyone can participate without a specific (physical) identity

Participants are known and Identified

Permissionless Blockchain

Permissioned Blockchain



A Permissioned Blockchain system consists of a set of known, identified entities that might not fully trust each other.

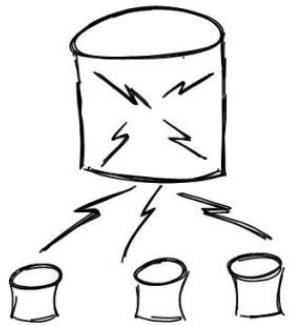
Blockchain Scalability

- The ability of a blockchain system to process an increasing number of transactions by adding resources to the system
- Two classes of solutions for scalability:

1) Off-chain (layer two): built on top of the main chain, move a portion of the transactions off the chain, e.g. lightning networks

2) On-chain (layer one): increase the throughput of the main chain
  - Vertical techniques: more power is added to each node to perform more tasks
  - Horizontal techniques: increase the number of nodes in the network

Sharding (as a horizontal technique): Partitioning the data into multiple shards that are maintained by different subsets of nodes



Sharding-based Approaches

- Proven technique to improve scalability of distributed databases
  - e.g., Amazon Dynamo, Spanner, Facebook's Tao, E-store, Calvin, H-store

1. Nodes are assumed to be crash-only
   - nodes may fail by stopping, and may restart, no malicious behavior
2. Cross-shard transactions are processed using a coordinator-based approach

- Coordinator-based approach has been used in Permissioned blockchain AHL
  - A committee (consisting of Byzantine nodes) plays the coordinator role [SIGMOD'19]

SharPer

- Support Byzantine Nodes
- Process cross-shard transactions without any coordinator
  - Requires a smaller number of nodes
  - Process cross-shard transactions in parallel

Network, Data, and Blockchain Ledger

Network

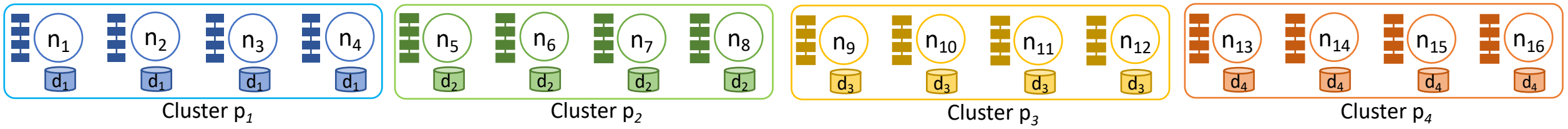
- Network is partitioned into clusters (either *2f+1* crash-only or *3f+1* Byzantine nodes)

Data

- Shard the application data and assign shards to clusters
- Each data shard is replicated on the nodes of a cluster

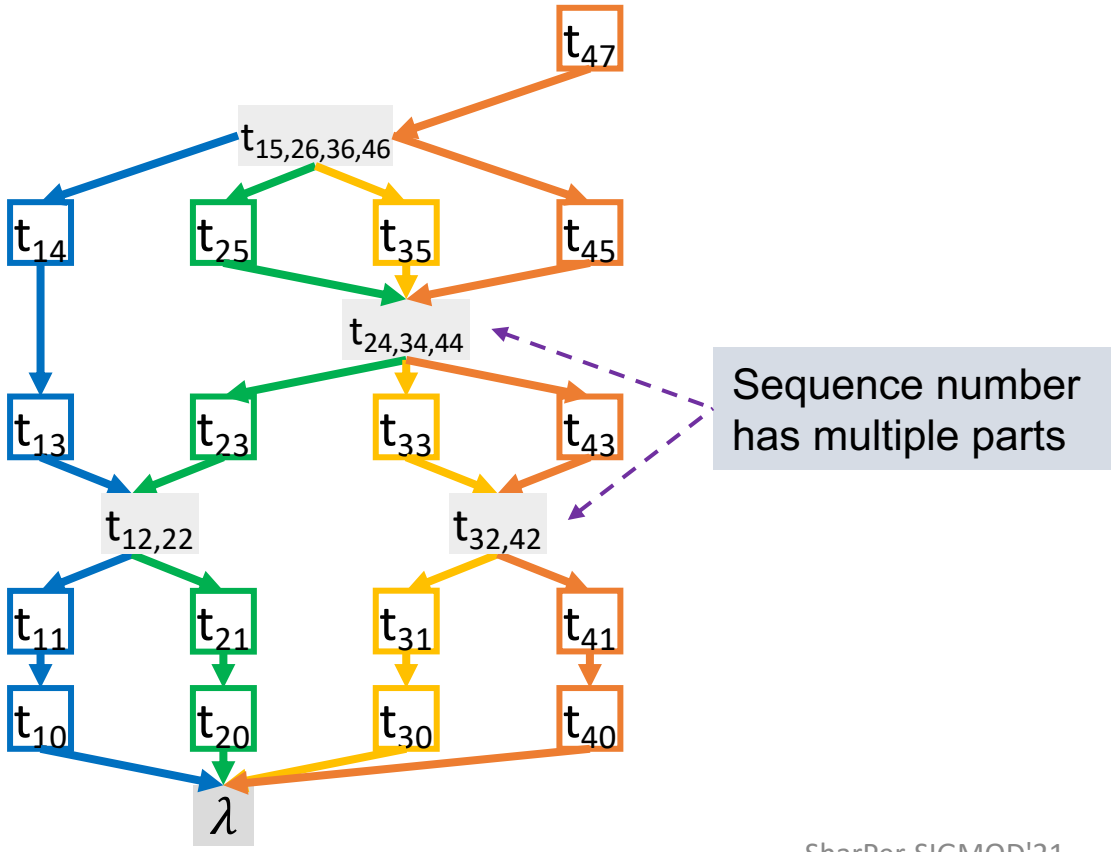
Blockchain Ledger

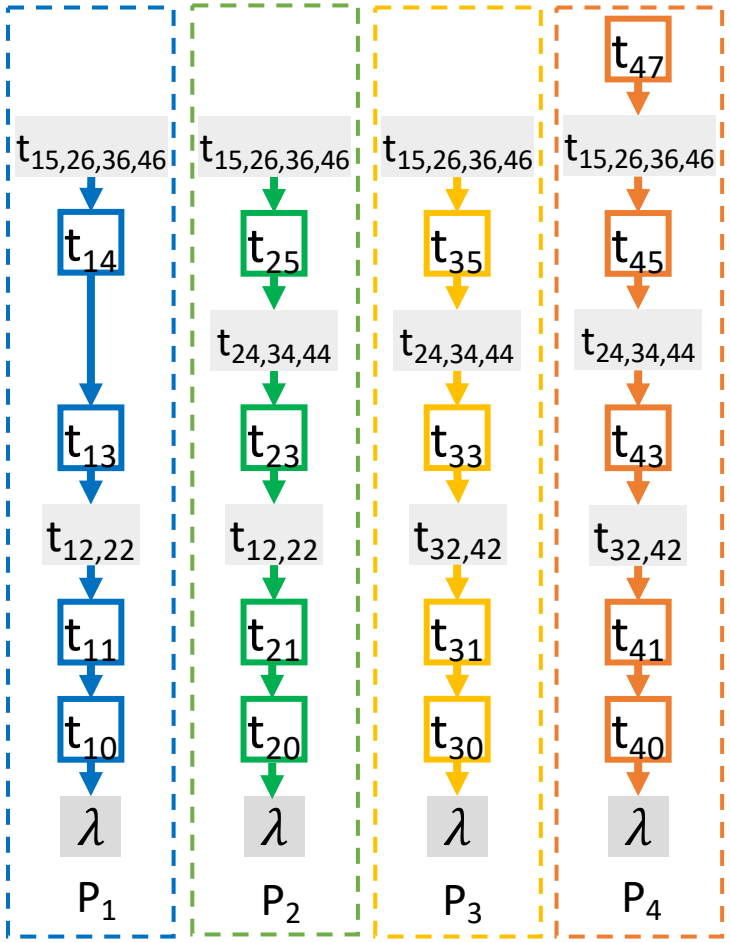
- The entire blockchain ledger is not maintained by any node
- Each cluster only maintains its own view of the blockchain ledger



SharPer Blockchain Ledger

- Intra-shard transactions of different clusters are processed in parallel
- Cross-shard transactions with non-overlapping clusters are processed in parallel
- Each cluster maintains its own view of the ledger





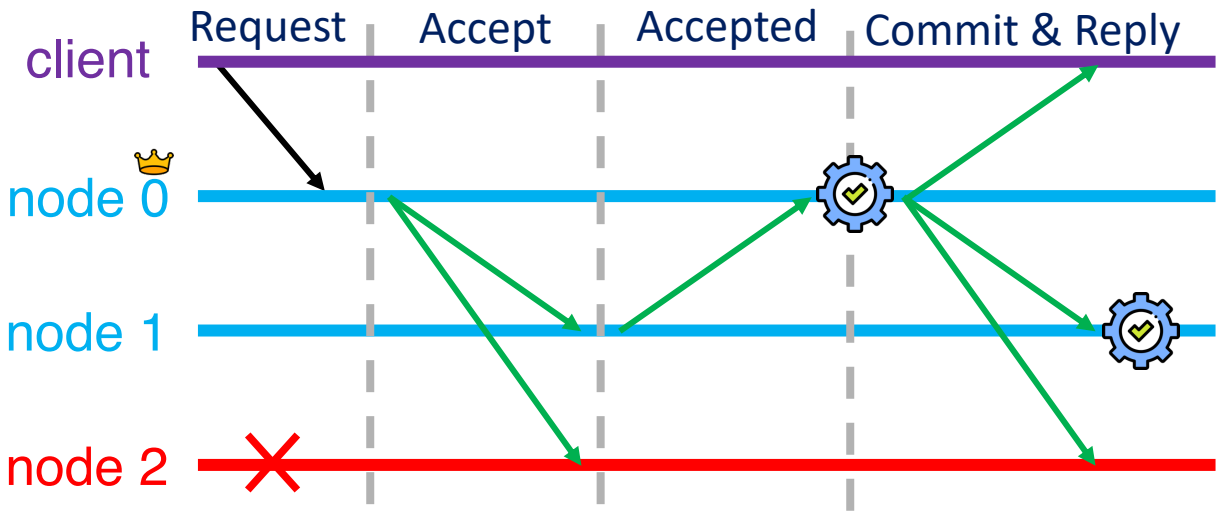
Consensus in SharPer

Intra-Shard Consensus
- Pluggable
- Depends on the failure model of nodes
  - Crash-Only: (Multi-)Paxos
  - Byzantine: PBFT

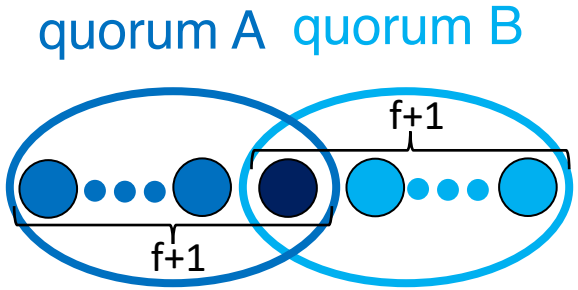
Cross-Shard Consensus
- Needs the participation of all involved clusters
  - Either f+1 crash-only or 2f+1 Byzantine nodes of every involved cluster must participate

"Jenkins, if I want another yes-man I'll build one."

(Multi-)Paxos [Lamport 1998]



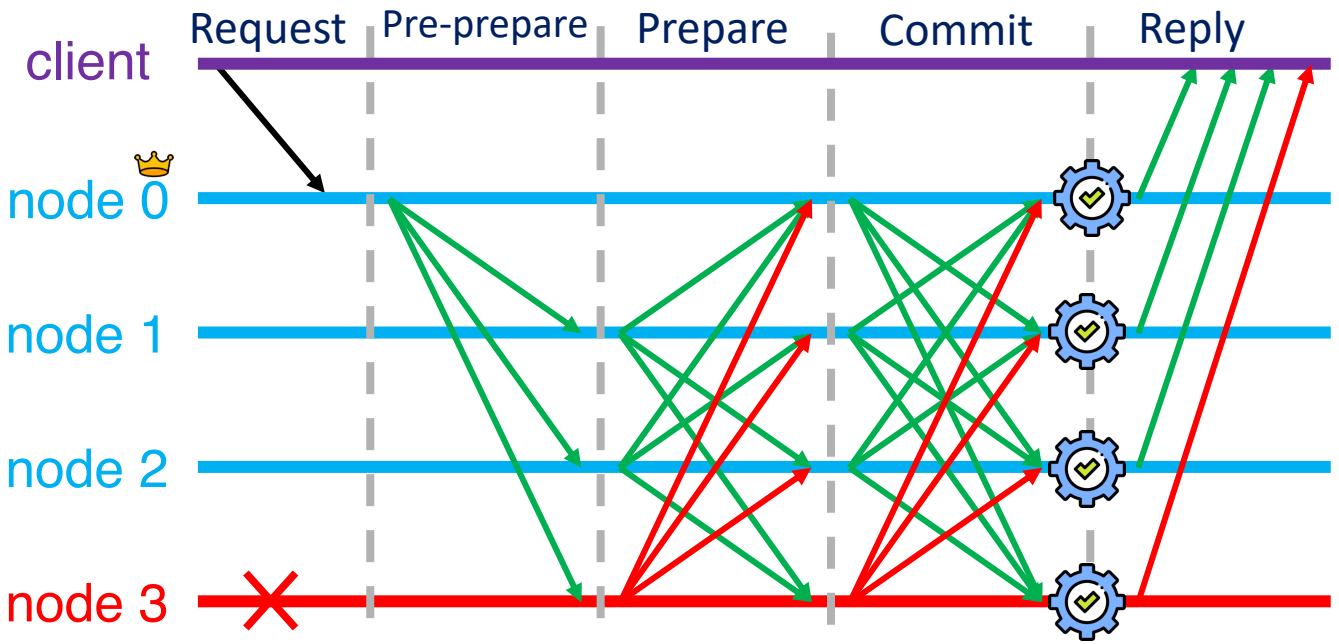
At Most f Crash Failures



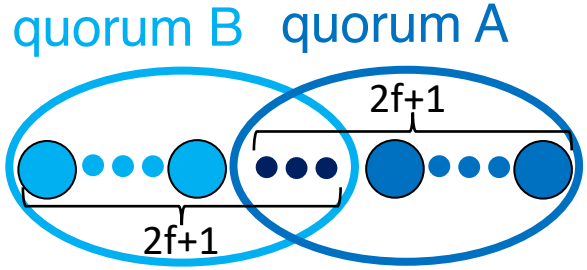
Network: 2f+1
Quorum: f+1
Intersection: 1

Phases: Two
Messages: O(n)
Quorum: f+1

Practical Byzantine Fault Tolerance [Castro and Liskov 1999]



At Most f Malicious Failures



Network: 3f+1
Quorum: 2f+1
Intersection: f+1

Phases: Three
Messages: $O(n^2)$
Quorum: 2f+1

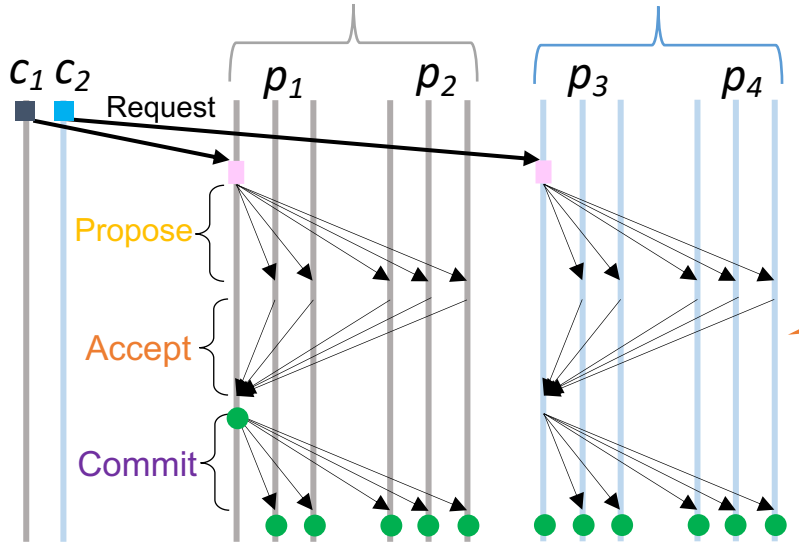
Cross-Shard Consensus with Crash-Only Nodes

Non-overlapping cross-shard transactions can be processed in parallel

<PROPOSE, h_i, d, m>

<ACCEPT, h_i, h_j, d, r>

<COMMIT, h_i, h_j, d> $\sigma_{\pi(p1)}$



Wait for f+1 matching accept from every involved cluster

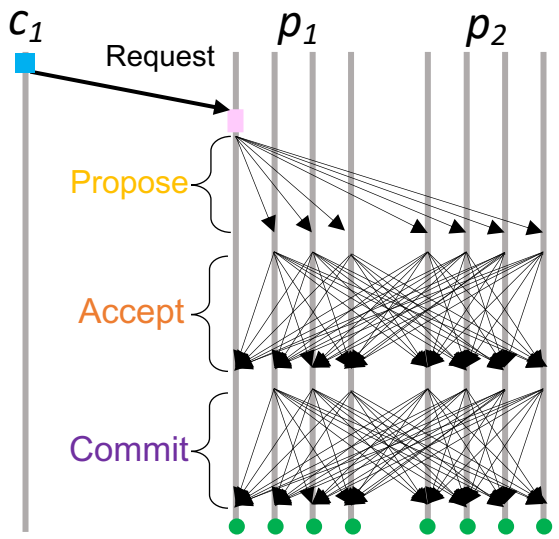
h_i: sequence number assigned by the initiator cluster (p_1 or p_3)
h_j: sequence number assigned by an involved cluster (p_2 or p_4)

Cross-Shard Consensus with Byzantine Nodes

<PROPOSE, h_i, d> $\sigma_{\pi(p1)}$, m>

<ACCEPT, h_i, h_j, d, r> $\sigma_{\pi(r)}$

<COMMIT, h_i, h_j, d, r> σ_r



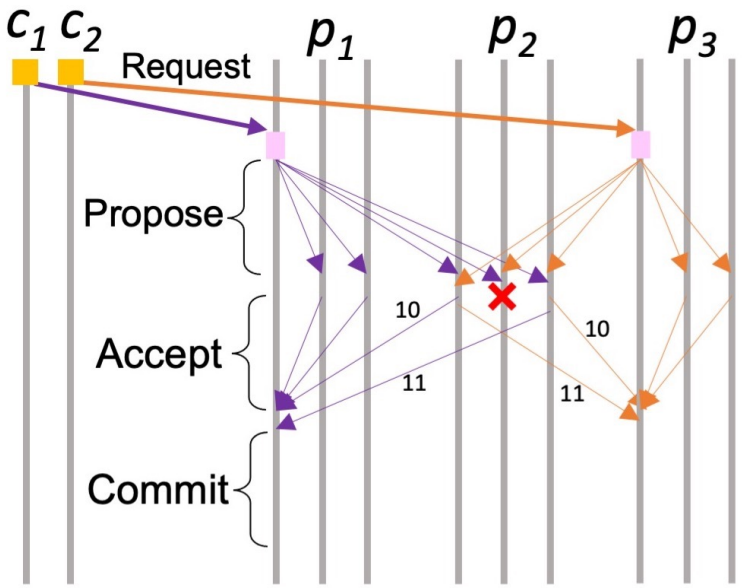
Wait for 2f+1 matching accept from every involved cluster

Wait for 2f+1 matching commit from every involved cluster

h_i: sequence number assigned by the initiator cluster (p_1)
h_j: sequence number assigned by an involved cluster (p_2)

Deal With Conflicting Messages

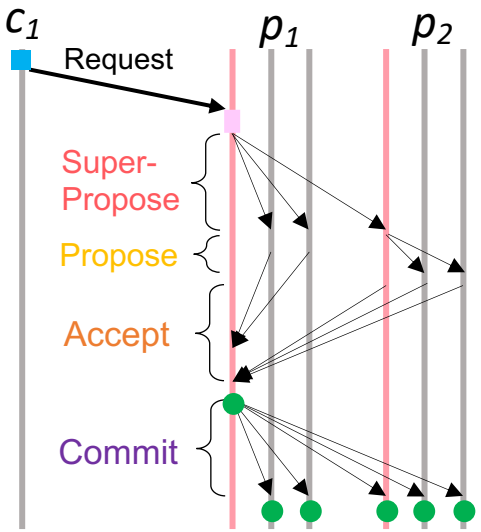
- A quorum of matching `Accept` messages from each cluster might not be received
  1. Nodes of a cluster assign inconsistent sequence numbers
     - e.g., an overlapping cluster receives parallel requests
  2. There is more than one overlapping cluster
     - Nodes do not process the second transaction before committing the first transaction to ensure consistency
     - Might result in deadlock situation



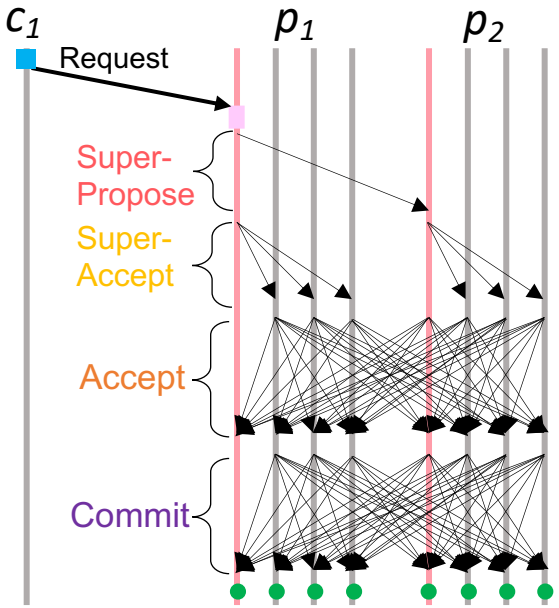
- SharPer uses Timers
  - Crash-only nodes: The initiator primary multicasts `Super-Propose` message to the primary nodes of conflicting clusters
  - Byzantine nodes: all nodes of conflicting clusters multicast `Super-Accept` messages
  - Deadlock situations: reach a unique order between deadlocked messages.

Deal with Heavy Workloads

- Only the primary node of each cluster assigns all sequence numbers: no conflicts occur
- Requires an extra intra-cluster message passing



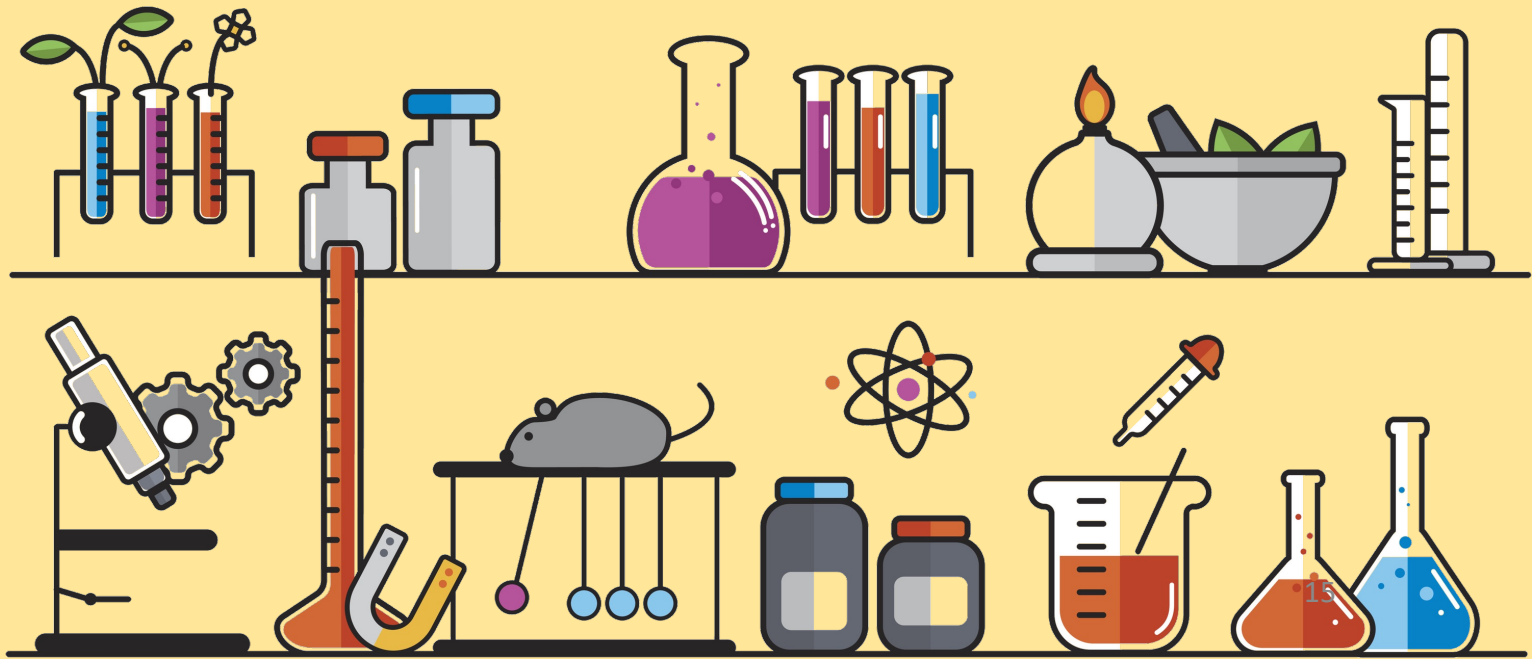
Crash-only nodes



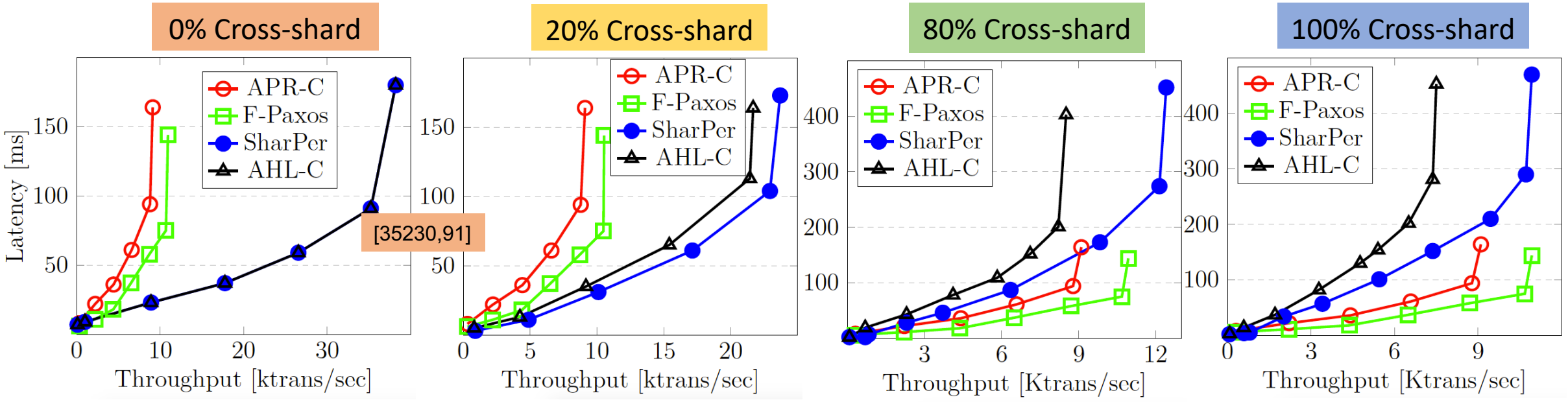
Byzantine nodes

Experimental Settings

- Systems:
  - Active/Passive Replication (APR-C, APR-B)
  - Fast Agreement (F-Paxos, FaB)
  - AHL-C, AHL-B
  - SharPer
- Platform: Amazon EC2
- Measuring performance
  - Throughput
  - Latency



Cross-Shard Transactions (Crash-only)



With no cross-shard transaction the performance of SharPer scales linearly

With low percentage of cross-shard transactions, SharPer demonstrates the best performance.

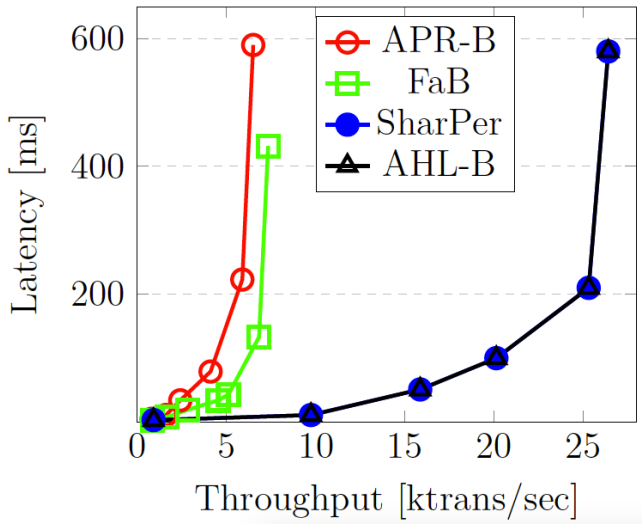
With high percentage of cross-shard transactions, using sharding has no advantage.

4 Clusters

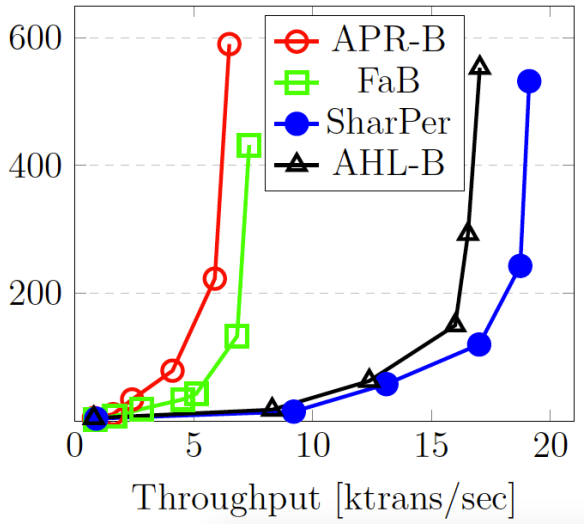
f = 1

Cross-Shard Transactions (Byzantine)

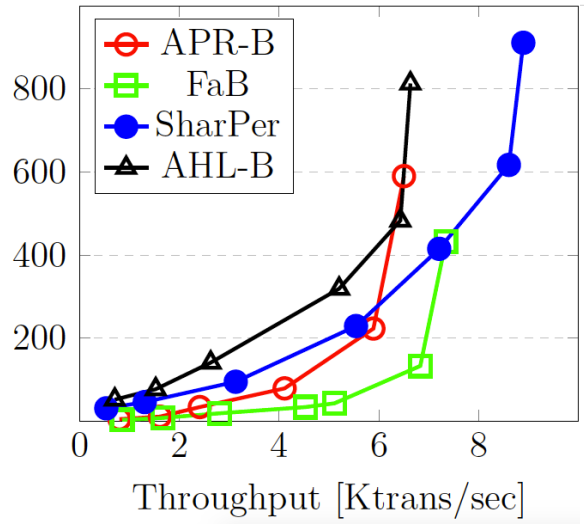
0% Cross-shard



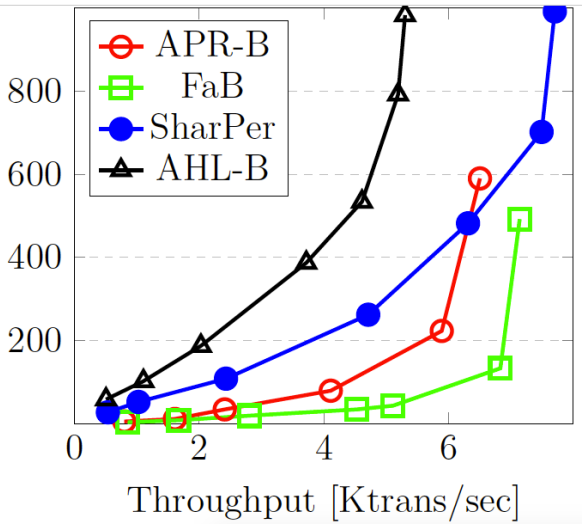
20% Cross-shard



80% Cross-shard



100% Cross-shard



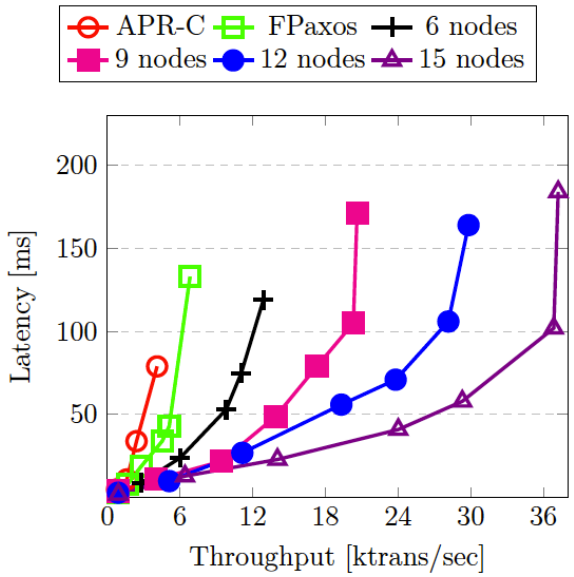
4 Clusters

f = 1

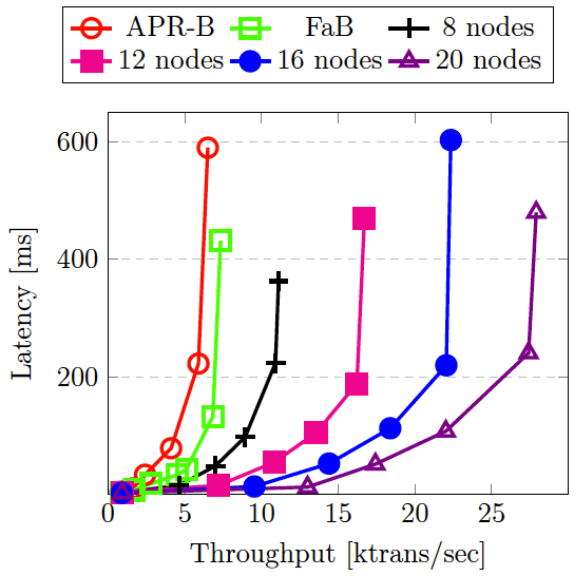
Performance with Different Number of Nodes

10% Cross-Shard

Crash-Only Nodes



Byzantine Nodes



The overall throughput of SharPer improves semi-linearly

Conclusion

- SharPer, a permissioned blockchain system that improves scalability by clustering (partitioning) the nodes
- Nodes of each cluster maintain a data shard and only a view of the blockchain ledger
- SharPer incorporates two flattened cross-shard consensus protocols for crash-only and Byzantine nodes
- The protocols order cross-shard transactions with non-overlapping clusters in parallel.
- The throughput of SharPer increases semi-linearly by increasing the number of clusters

Thank You!