ByShard: Sharding in a Byzantine Environment

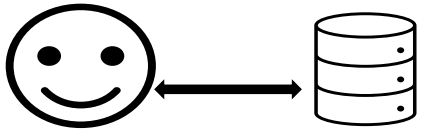
Hellings, J., & Sadoghi, M. (2021). ByShard. Proceedings of the VLDB Endowment, 14(11), 2230–2243.

October 30, 2023

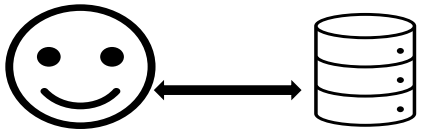
Presented By: Amoolya, Manali, Mariana, Shravani

What is Sharding?

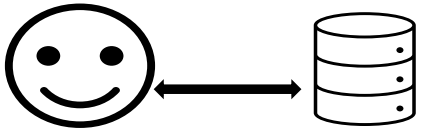
Location: US



Location: US Location: US

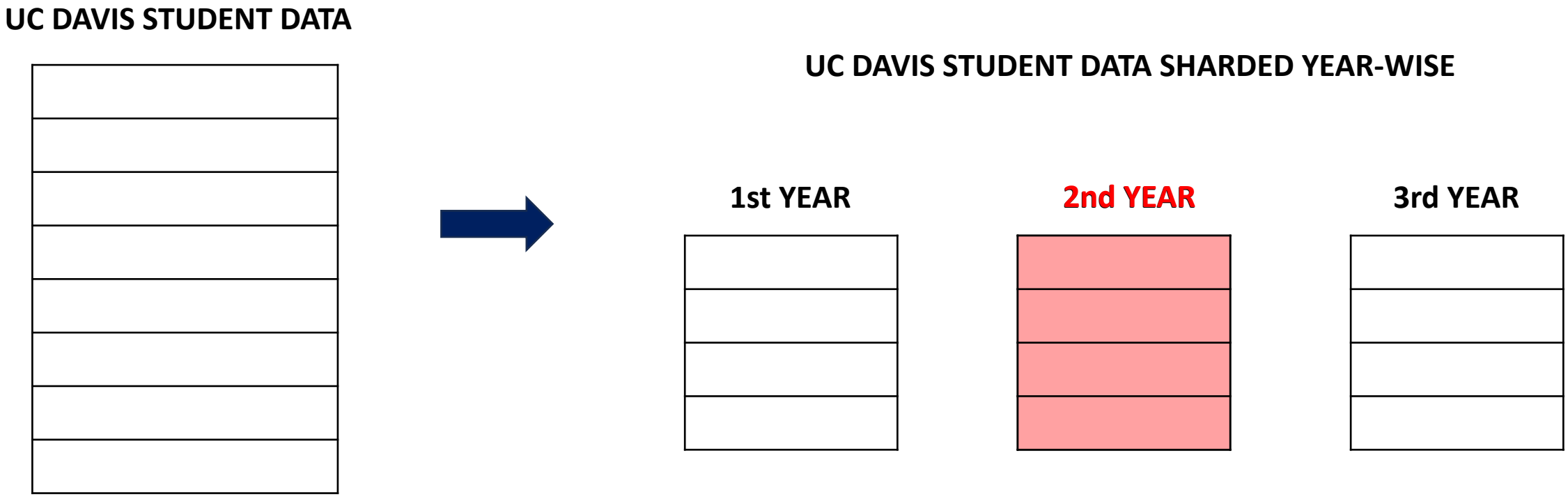


Location: Asia Location: Asia



Location: Africa Location: Africa

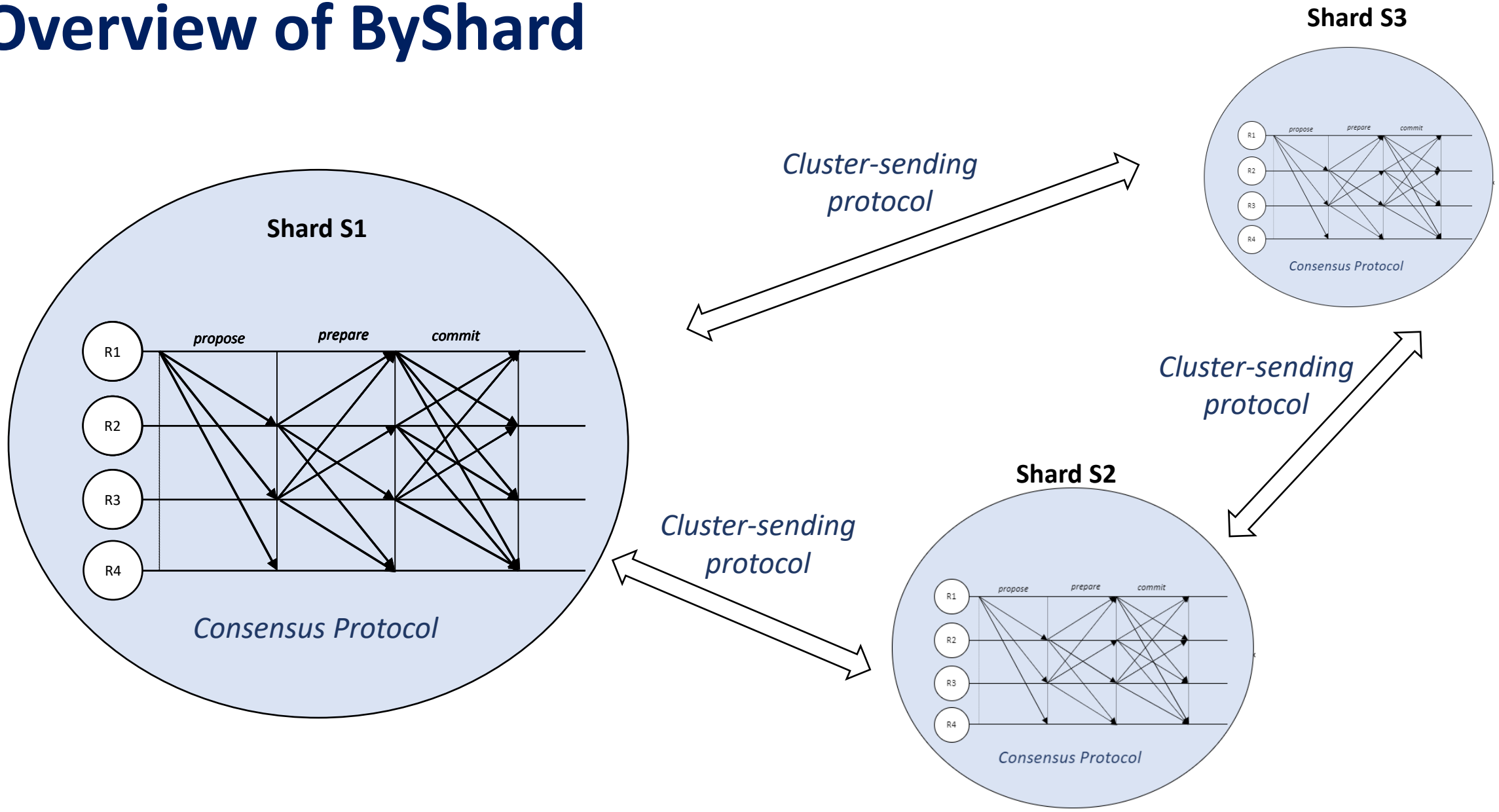
What is Sharding?



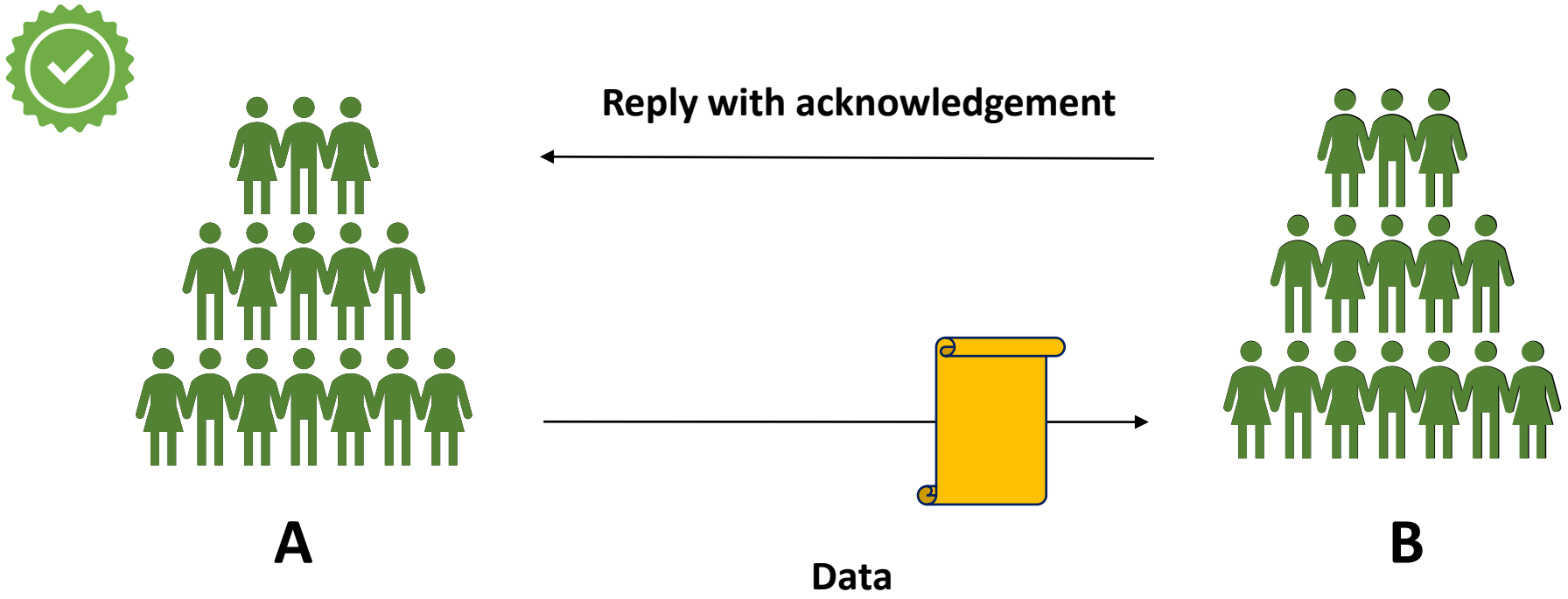
ByShard: A resilient sharding framework

- A base for general-purpose resilient sharded data management systems
  - Implemented principles of traditional distributed databases into sharded resilient systems
- Minimize the use of consensus

Overview of ByShard



Cluster Sending Protocol



Orchestrate-Execute Model

Orchestration

- Linear orchestration
- Centralized orchestration
- Distributed orchestration

Execution

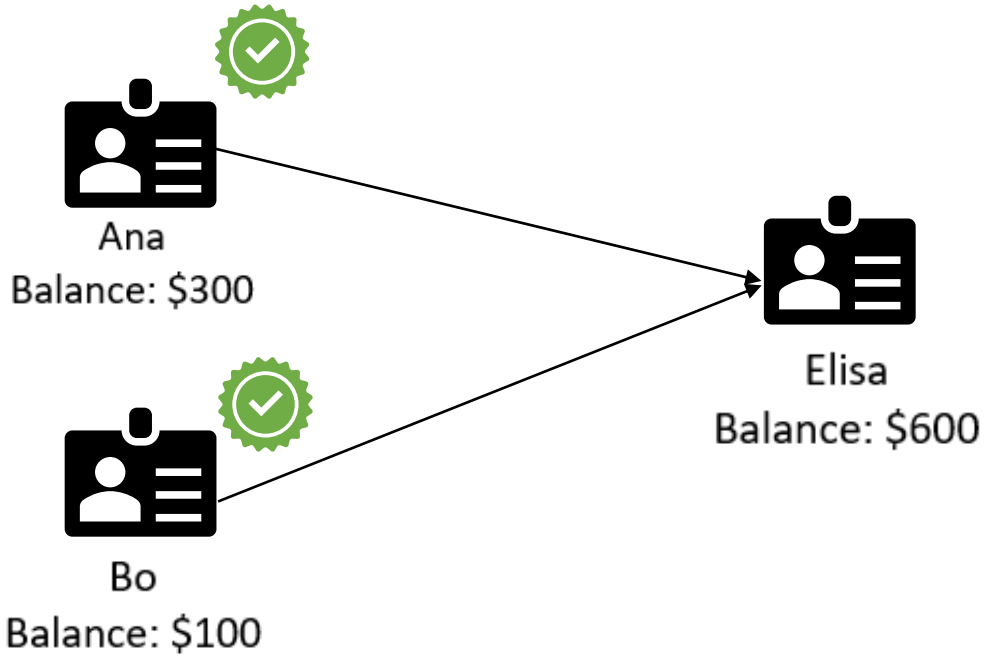
- Isolated-free direct execution
- Lock-based execution

Orchestrate-Execute Model

This model reduces the number of consensus steps involved in each transaction.

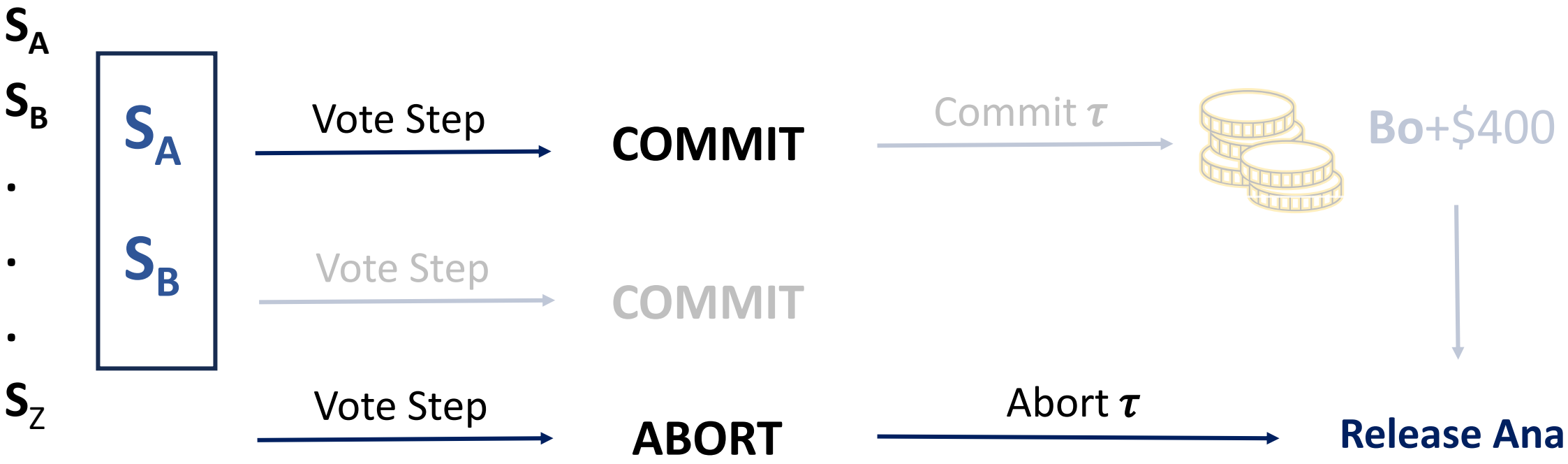
Ex: If Ana has atleast $500 and Bo atleast $200 than transfer $200 from Ana and $100 from Bo to Elisa.

- Vote Step
- Commit Step
- Abort Step



Sharded system - Example

τ = "if Ana has atleast $500 and Bo has atleast $200, then move $400 from Ana to Bo"



Important Terminologies

τ - Transaction

nv - Vote Steps

na - Abort Steps

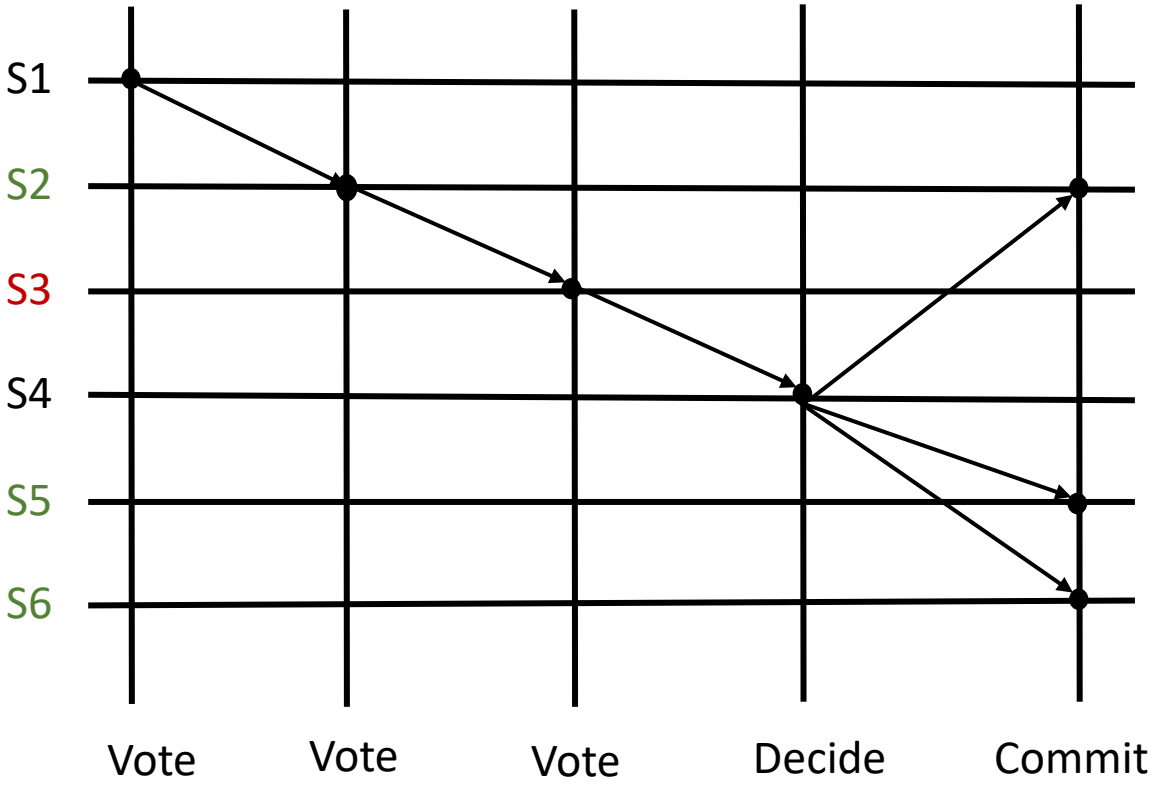
nc - Commit Steps

$\{S1, \ldots, Sn\}$ - Total no. Of Shards

shards(τ) - Shards involved in the transaction

Sr - Root Shard

Linear Orchestration



For transaction τ,

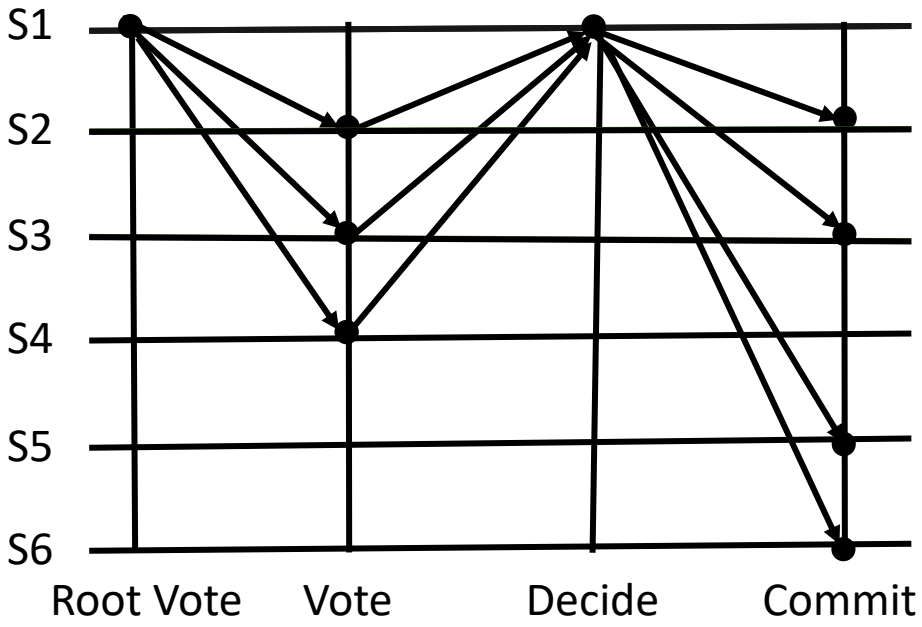
Consensus Steps:

$nv + nc$

Cluster Sending:

$nv + nc$ -1

Centralized Orchestration



For transaction τ,

- Consensus Steps:

$nv + nc + 1$

- Cluster Sending Steps:

$2(nv - 1) + nc$

Important Terminologies

τ - Transaction

nv - Vote Steps

na - Abort Steps

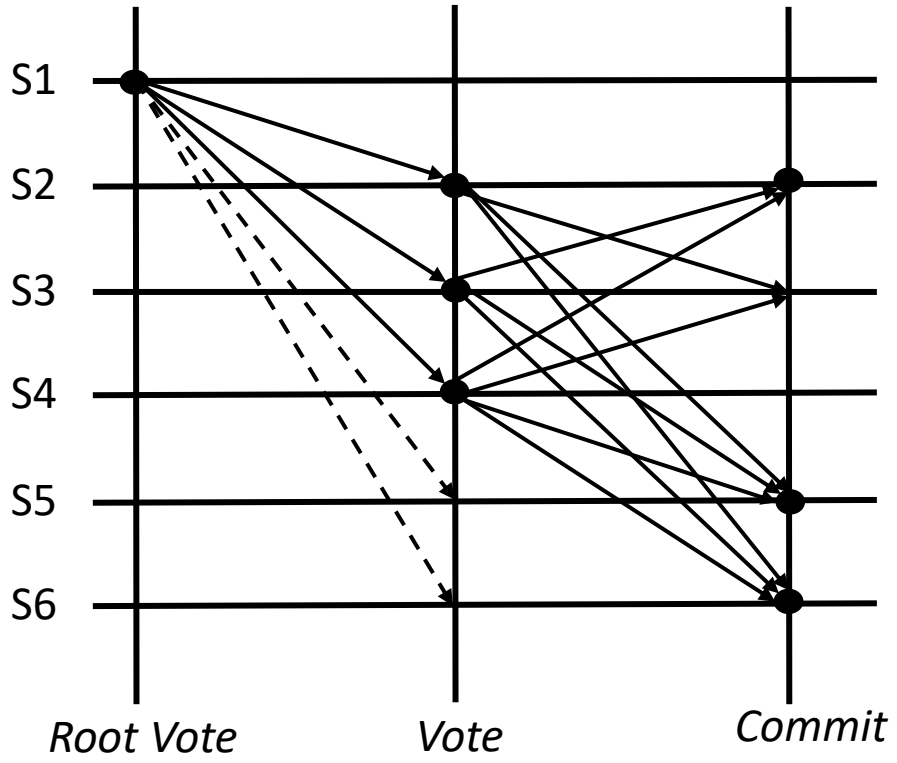
nc - Commit Steps

$\{S1, \ldots, Sn\}$ - Total no. of Shards

shards(τ) - Shards involved in the transaction

Sr - Root Shard

Distributed Orchestration



For transaction τ,

Consensus Steps:
$nv + nc$

Cluster Sending:
$nv\ (na + nc\ ) + (nv - 1)$

Orchestration Summary

Orchestration	Consecutive Consensus Steps	Consensus Steps ●	Cluster Sending Steps →
Linear	nv+1	$nv + nc$	$nv + nc$ -1
Centralized	4	$nv + nc + 1$	$2(nv - 1) + nc$
Distributed	3	$nv + nc$	$nv\ (na + nc\ ) + (nv - 1)$

Execution in OEM

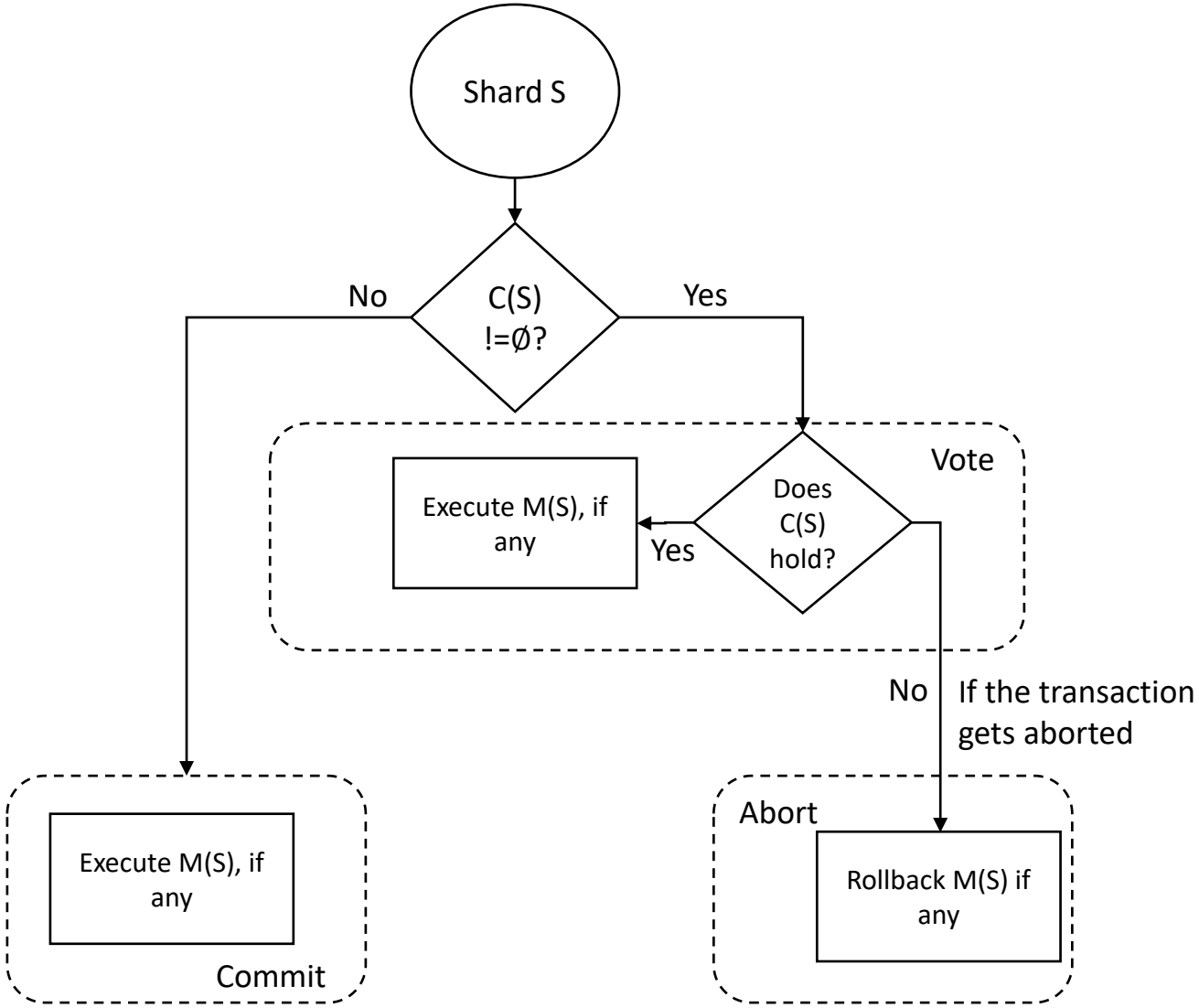
- Constraints

  CON(X, y) = “*the balance of X is at least y*”

- Modifications

  MOD(X, y) = “*add y to the balance of X*”

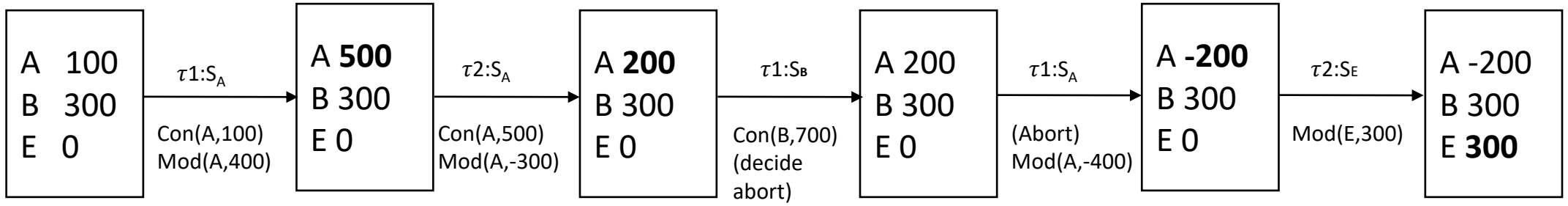
Isolated-free direct execution



Isolated-free direct execution

Provides degree 0 isolation which can lead to violations of constraints.

$\tau 1$ = Con(A, 100), Con(B, 700), Mod(A, 400), Mod(B, -400)
$\tau 2$ = Con(A, 500), Mod(A, -300), Mod(E, 300)

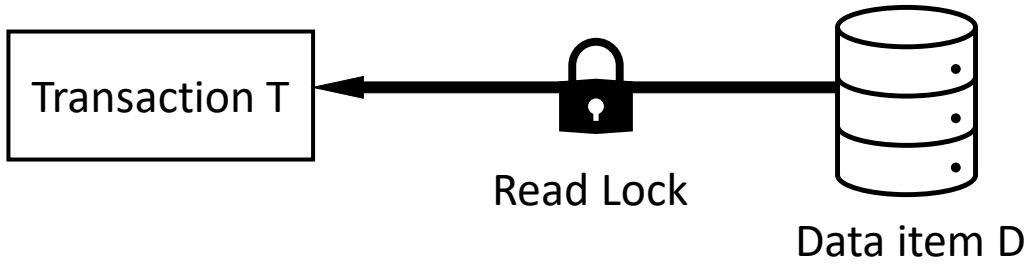


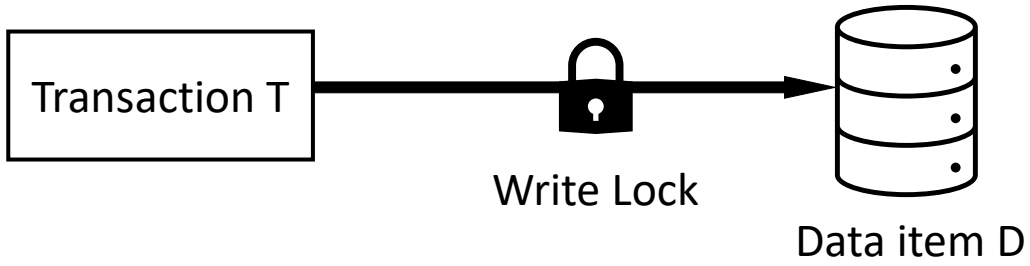
Isolated-free direct execution

- Safe – Rolling back M(X, y) with y<=0 (removal) => Increases the balance of X and X will never be negative -> Vote Step
- Unsafe – Rolling back M(X, y) with y>=0 (addition) => Decreases the balance of X and X can be negative -> Commit Step
- Worst case: 2 shard steps when committing.

Lock-based execution

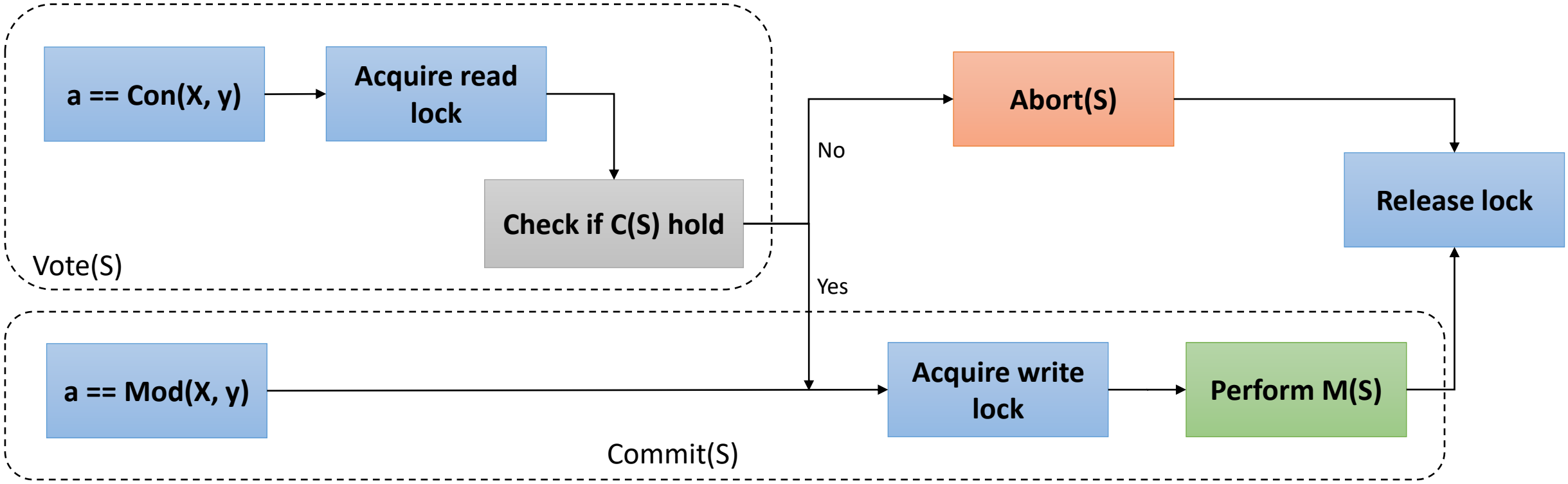
- Two-phase locking
- Possibility of higher isolation – degree 3



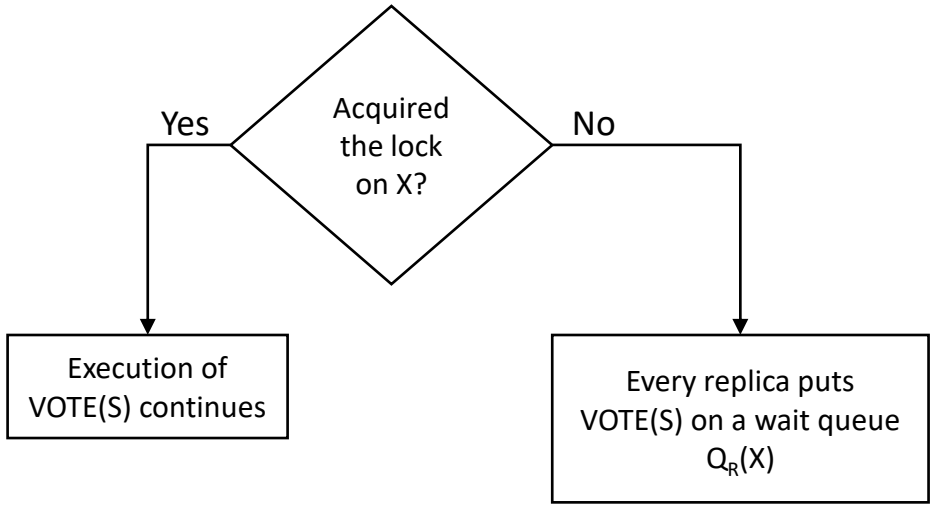


Lock-based execution

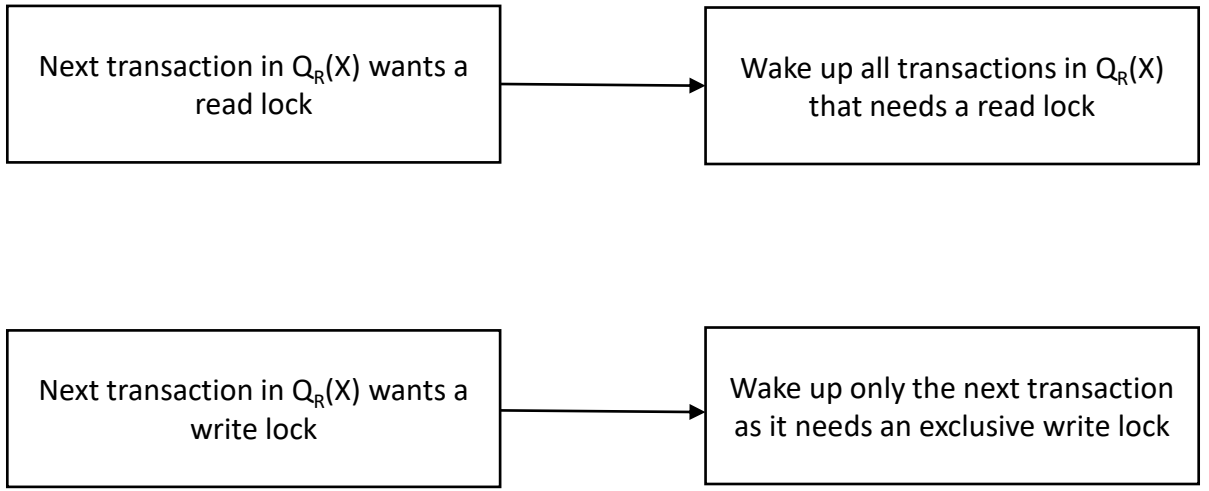
- Let S ∈ Shards(τ)
- Accounts(S) = {a | Con(X, y) ∈ C(S) ∨ Mod(X, y) ∈ M(S)}



Lock-based execution



After the existing lock is released



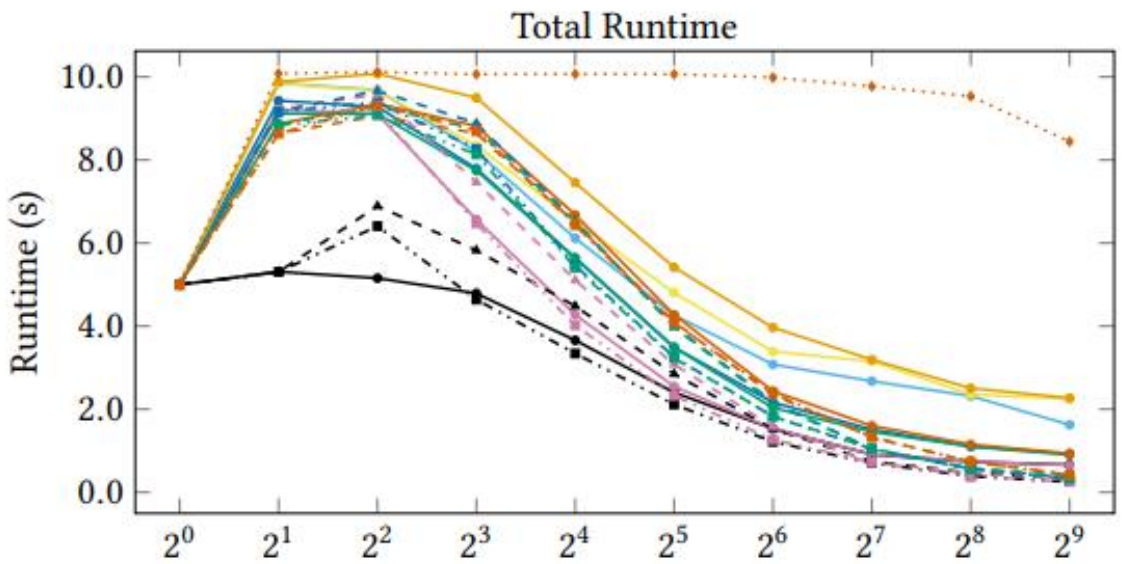
- Hence vote-step can obtain all its locks in only a single consensus step. Wake up step is deterministic. No extra consensus are necessary to determine which of the next transactions will be executed.

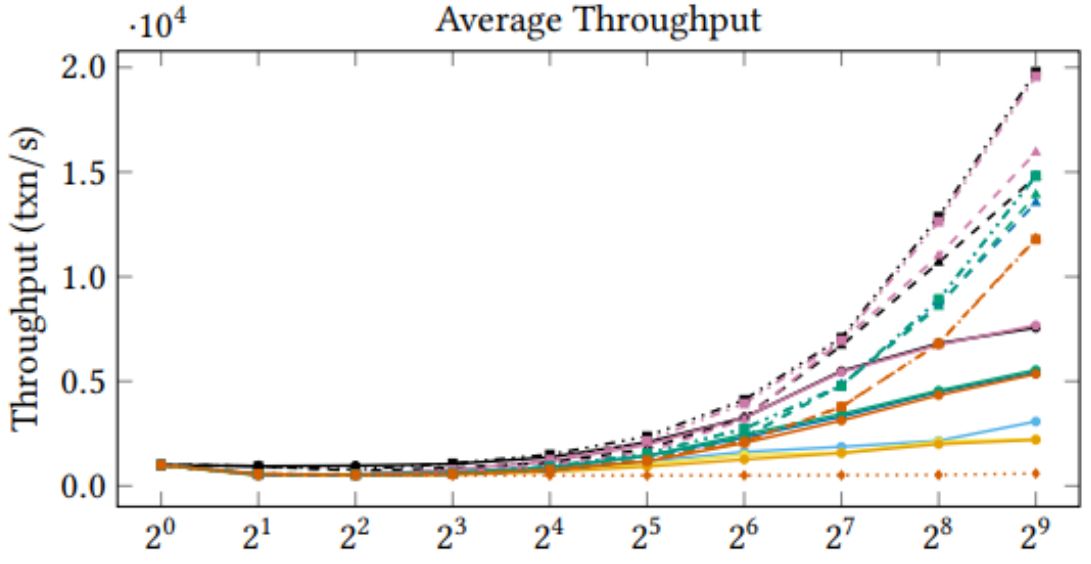
Lock-based execution

- Pro: Provides serializability and highest degree of isolation
- Con: Large transaction processing latency whenever contention is high
- Read uncommitted execution
  - Degree 1 isolation
  - No read locks on any data item
  - Very useful for read-heavy workloads
- Read committed execution
  - Degree 2 isolation
  - Read locks are released directly after reading an item
  - Minimizes the time read locks are held

Performance Evaluation

	Isolation-Free execution (write uncommitted)		Lock-based execution							
			Read Uncommitted		Read Committed		Serializable			
	unsafe	*safe*	*blocking*	*non-blocking*	*blocking*	*non-blocking*	*blocking*	*non-blocking*		
Linear	LIFU	LIFS	LRUB	LRUNB	LRCB	LRCNB	LSB	LSNB		
Centralized	CIFU	CIFS		CRUNB		CRCNB		CSNB	AHL	(reference committee)
Distributed	DIFU	DIFS		DRUNB		DRCNB		DSNB		





* Graphs taken from the paper figure 6.1

Conclusion

- ByShard - a general-purpose framework for sharded resilient data management systems
- Orchestrate-Execute Model (OEM)
- 18 multi-shard transaction processing protocols with tradeoffs between throughput, latency, isolation level and abort rate

Thank you