ByShard: Sharding in Byzantine Environment

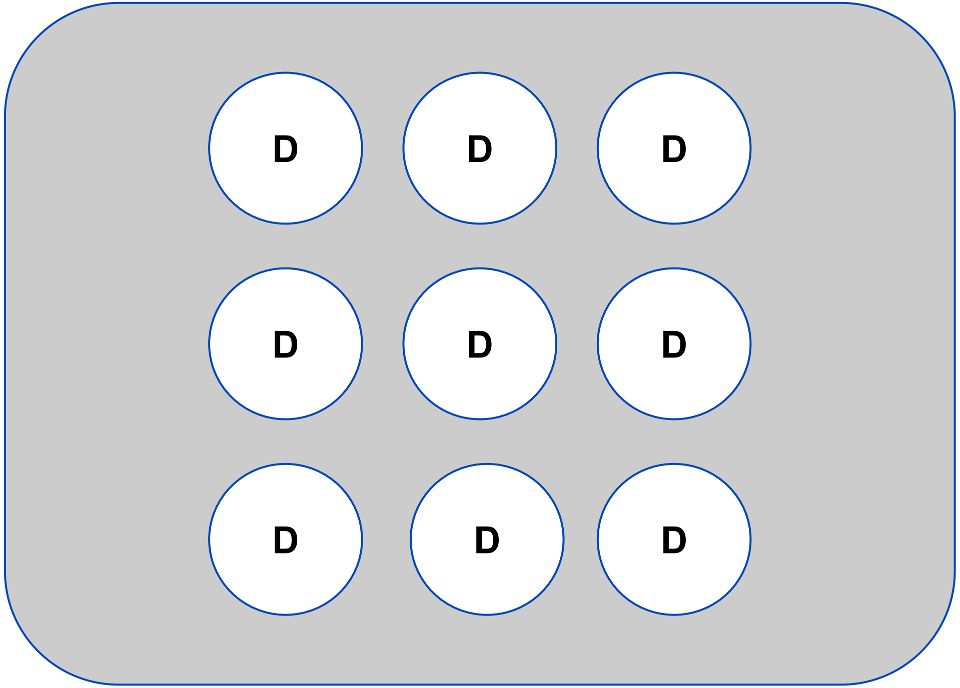
By Jelle Hellings and Mohammad Sadoghi

Presenters: Manjunath Jakaraddi, Rohith Raj Srinivasan, Abrar Syed & Mohammed Asim

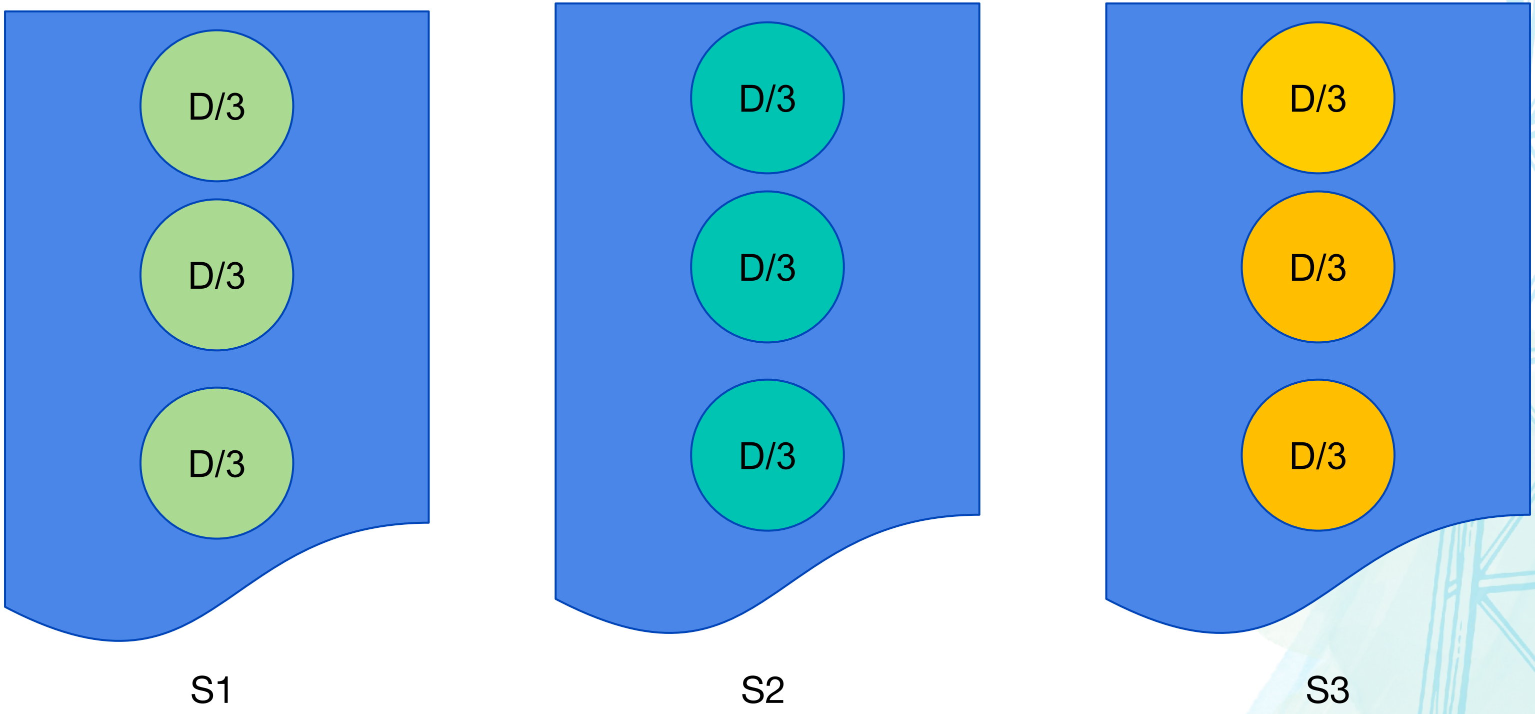
Table of Contents

1. Introduction
2. ByShard Framework
3. OEM model
4. Orchestrate
5. Execute
6. Performance Evaluation
7. Q&A

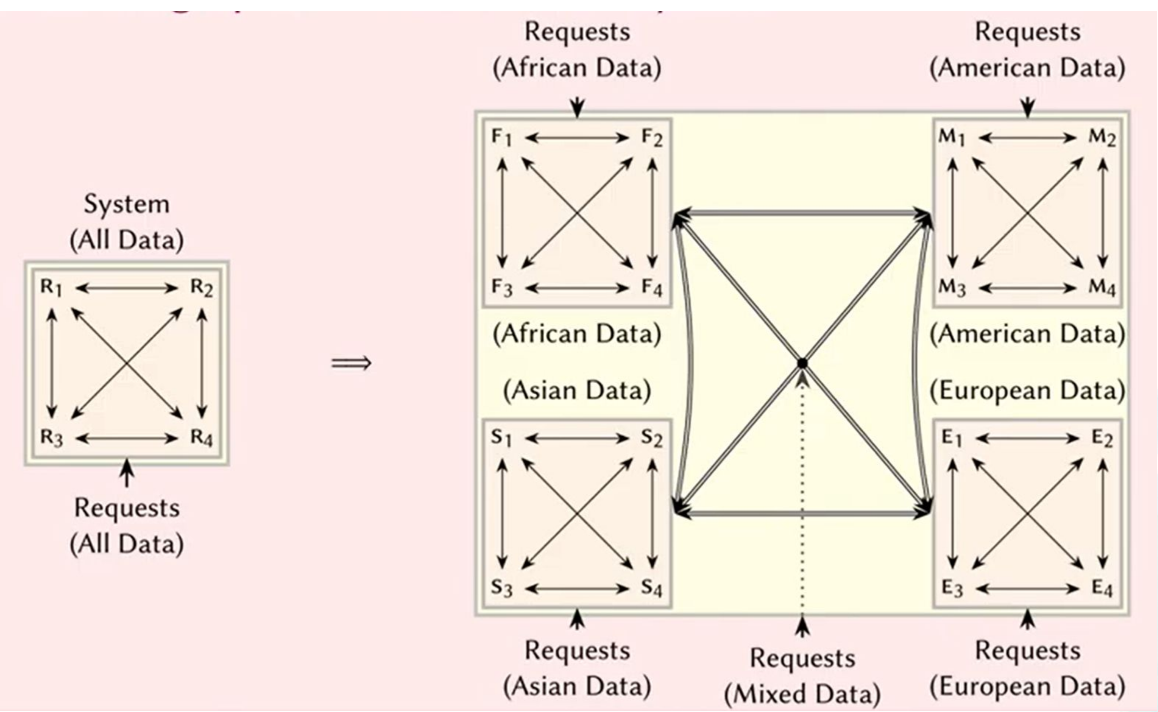
Typical Blockchain Inspired System



Data Distribution after Sharding



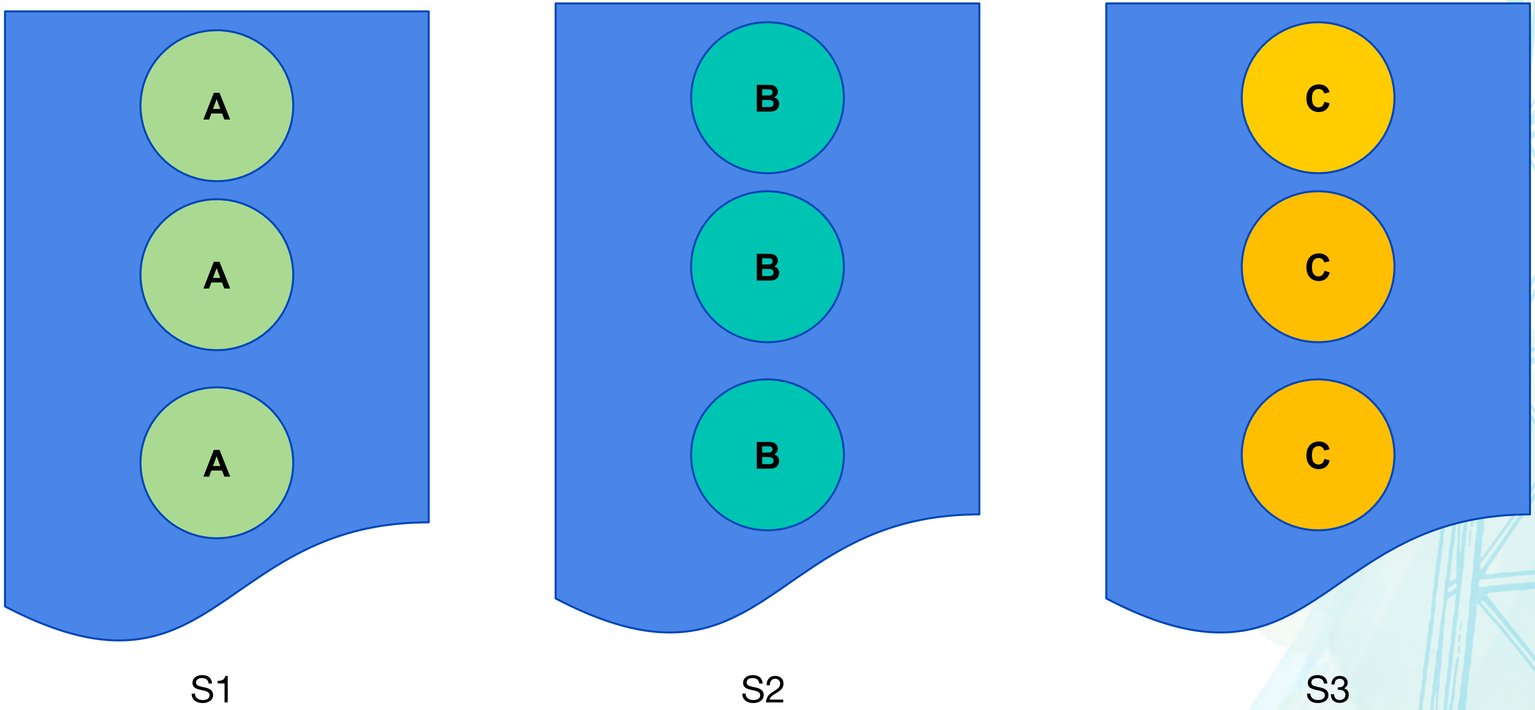
Sharding



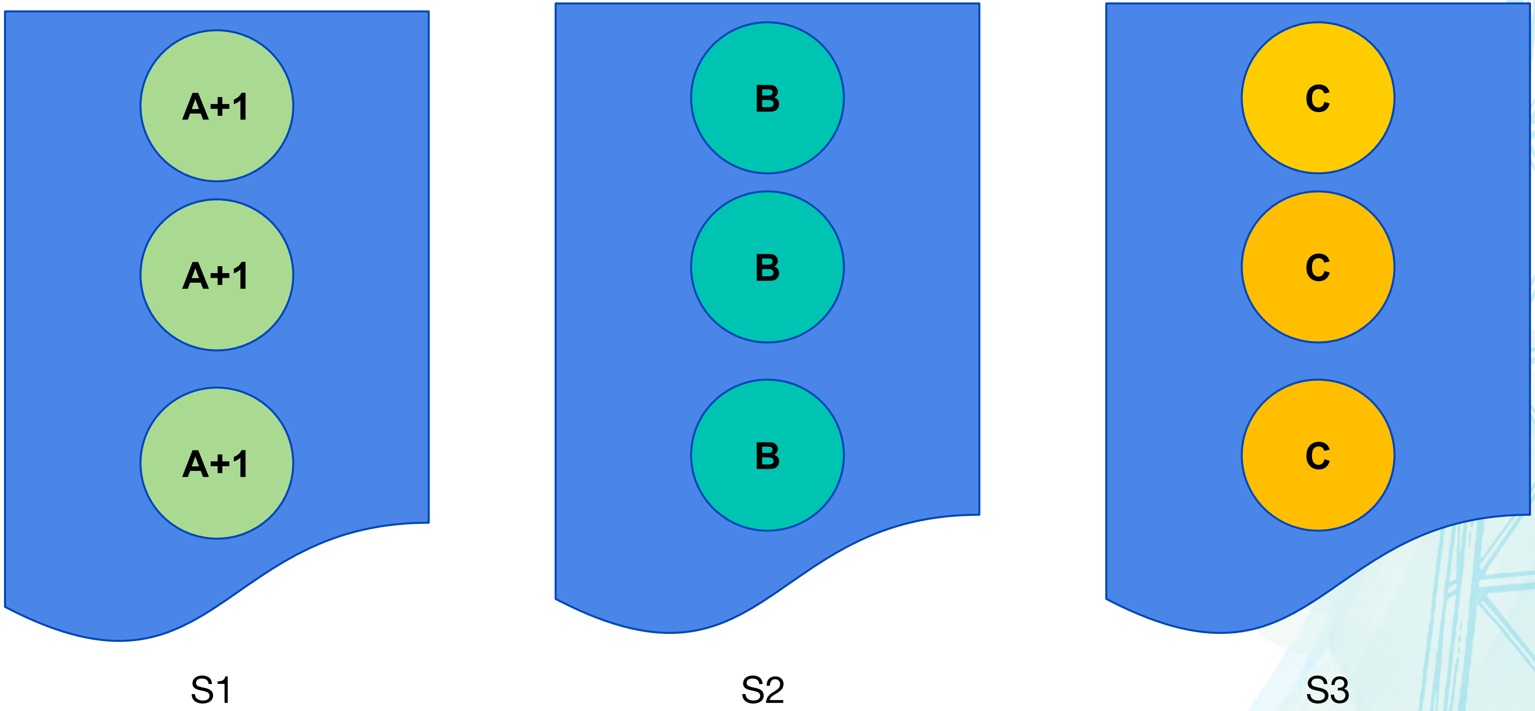
UCDAVIS

Cite: Hellings, Jelle, and Mohammad Sadoghi. "Byshard: Sharding in a byzantine environment." Proceedings of the VLDB Endowment 14.11 (2021): 2230-2243.

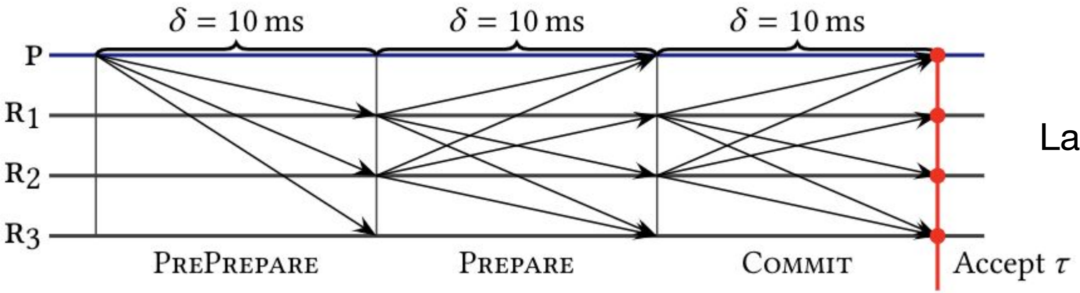
Merits of Sharding



Merits of Sharding



Latency in PBFT



Latency = 3δ = 30 ms

Cluster Sending Protocol

Definition

A cluster-sending protocol provides reliable communication between resilient clusters S1 and S2. To enable S1 to send a value v to S2, cluster sending protocols provide the following guarantees:

(1) S1 is able to send v to S2 only if there is agreement on sending v among the non-faulty replicas in S1;

(2) all non-faulty replicas in S2 will receive the value v; and

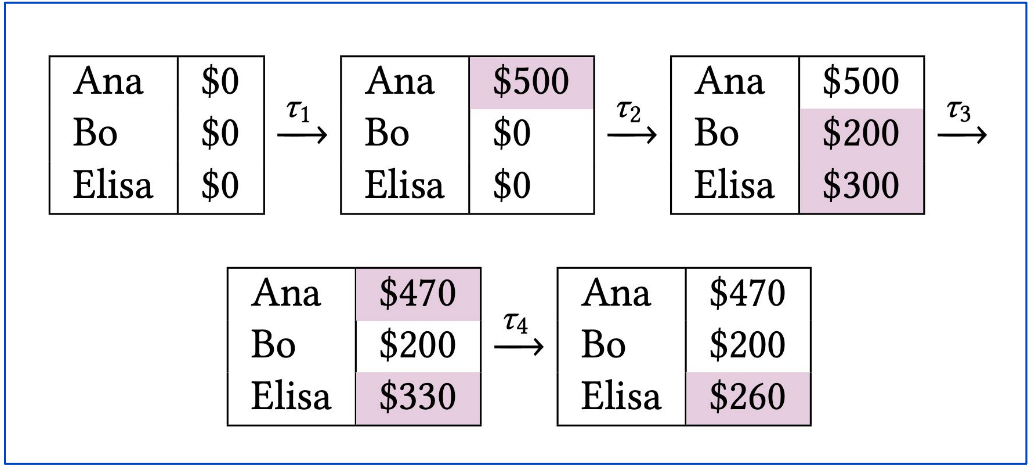
(3) all non-faulty replicas in S1 obtain confirmation of receipt.

Banking Example

- Consider a banking system transactions
  - $\boldsymbol{\tau}_1$ = “add $500 to **Ana”**
  - $\boldsymbol{\tau}_2$ = “add $200 to **Bo** and $300 to **Elisa**”
  - $\boldsymbol{\tau}_3$ = “move $30 from **Ana** to **Elisa**”
  - $\boldsymbol{\tau}_4$ = “remove $70 from **Elisa**”

Cite

Banking Example



Ana	\$0
Bo	\$0
Elisa	\$0

$\xrightarrow{\tau_1}$

Ana	\$500
Bo	\$0
Elisa	\$0

$\xrightarrow{\tau_2}$

Ana	\$500
Bo	\$200
Elisa	\$300

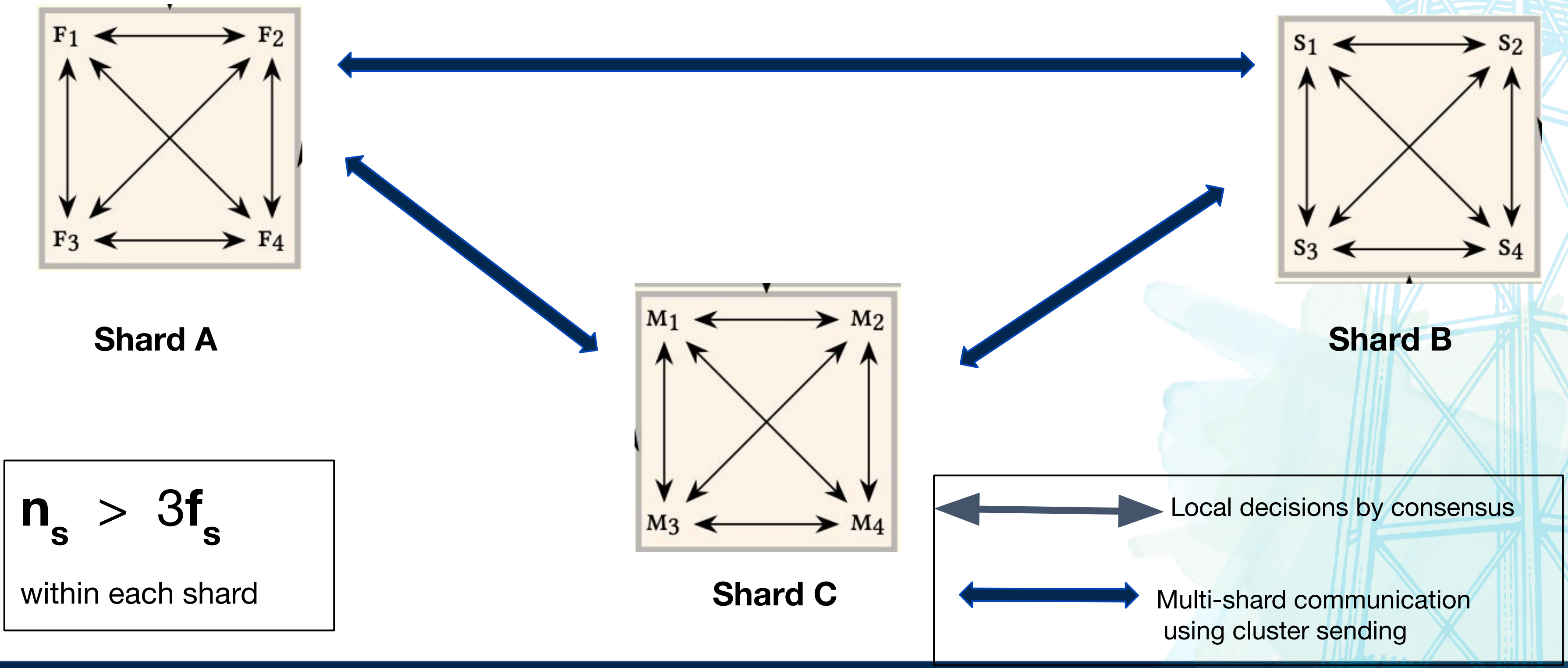
$\xrightarrow{\tau_3}$

Ana	\$470
Bo	\$200
Elisa	\$330

$\xrightarrow{\tau_4}$

Ana	\$470
Bo	\$200
Elisa	\$260

Multi shard transaction execution in a Byzantine Environment



Shard A

Shard B

Shard C

$n_s > 3f_s$

within each shard

Local decisions by consensus

Multi-shard communication using cluster sending

UCDAVIS

Hellings, Jelle, and Mohammad Sadoghi. "Byshard: Sharding in a byzantine environment." Proceedings of the VLDB Endowment 14.11 (2021): 2230-2243.

OEM Model

We process multi-shard transaction using the **orchestrate-execute model (OEM)**.

OEM model's components:

- **Orchestration**: replicates transactions among all replicas in all involved shards and reach an atomic decision.
- **Execution**: executes operations of a shard step.

Multishard Transaction Example

Each of the accounts are in different shards:

Ana → **Shard A** (S_a)

Bo → **Shard B** (S_b)

Elisa → **Shard E** (S_e)

τ = “if **Ana** has $500 and **Bo** has $200, then

move $400 from **Ana** to **Elisa**; move $100 from **Bo** to **Elisa**”

Multishard Transaction

$\sigma_1 \rightarrow$ “if Ana has \$500, then remove \$400 from Ana; ⇒ $S_b(\boldsymbol{\sigma}_2)$

else ⇒ send failure to c”

$\sigma_2 \rightarrow$ “if Bo has \$200, then remove \$100 from Bo; ⇒ $S_e\ (\sigma 3)$

else ⇒ $Sa\ (\sigma 4)$”

$\sigma_3 \rightarrow$ “add \$500 to Elisa and ⇒ send success to c”

$\sigma_4 \rightarrow$ “add \$400 to Ana and ⇒ send failure to c”

Where $\boldsymbol{\sigma}$ is the shard step and ⇒ is the cluster sending step

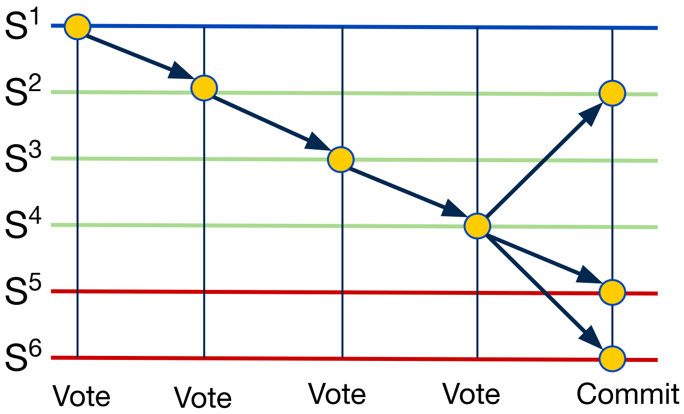
Types of Shard Steps

- **Vote Step (σ_1 and σ_2)**
  - verifies constraints and vote to commit or not.
- **Commit Step (σ_3)**
  - performs operation to finalize the transaction τ
- **Abort Step (σ_4)**
  - Abort the operation and rollback all the changes

Types of Orchestration

1. Linear orchestration
2. Centralized orchestration
3. Distributed orchestration

Linear orchestration



# of steps:

Consecutive consensus steps = $\mathbf{n_v + 1}$

Consensus steps = $\mathbf{n_v + n_c}$

Cluster-sending steps = $\mathbf{n_v + n_c - 1}$

● = Consensus

→ = Cluster-sending step

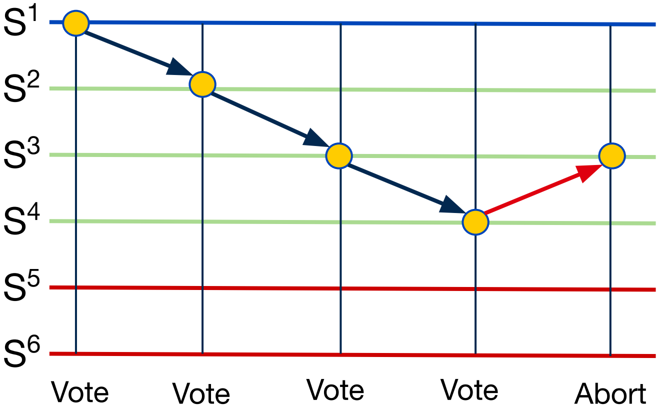
$\mathbf{S^1, S^2, S^3, S^4}$ → Vote-steps

$\mathbf{S^2, S^5, S^6}$ → Commit-steps

$\mathbf{S^3}$ → Abort-step

UCDAVIS

Linear orchestration



● = Consensus

→ = Cluster-sending step

S^1, S^2, S^3, S^4 → Vote-steps
S^2, S^5, S^6 → Commit-steps
S^3 → Abort-step

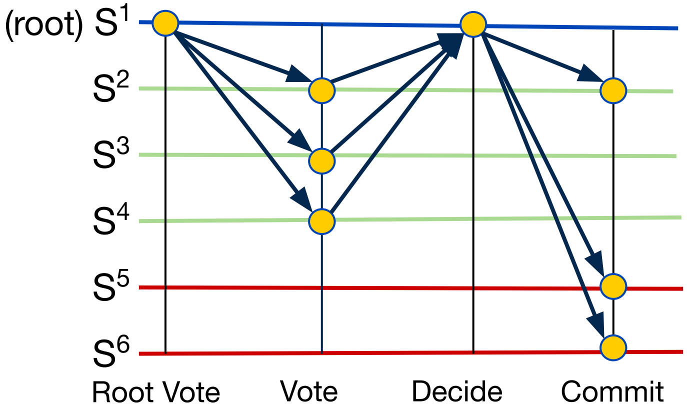
<u># of steps:</u>
Consecutive consensus steps = $\mathbf{n_v + 1}$
Consensus steps = $\mathbf{n_v + n_a}$
Cluster-sending steps = $\mathbf{n_v + n_a - 1}$

Merits:

1. Simplicity
2. Abort-fast ability

Limitations:
# Consecutive consensus steps (worst-case) = $|\mathit{shards}(\tau)| + \mathit{1}$

Centralized orchestration



<u># of steps:</u>

Consecutive consensus steps = **4**

Consensus steps = 1 + (n_v - 1) + 1 + n_c = **n_v + n_c + 1**

Cluster-sending steps = (n_v - 1) + (n_v - 1) + n_c = **2(n_v - 1) + n_c**

● = Consensus

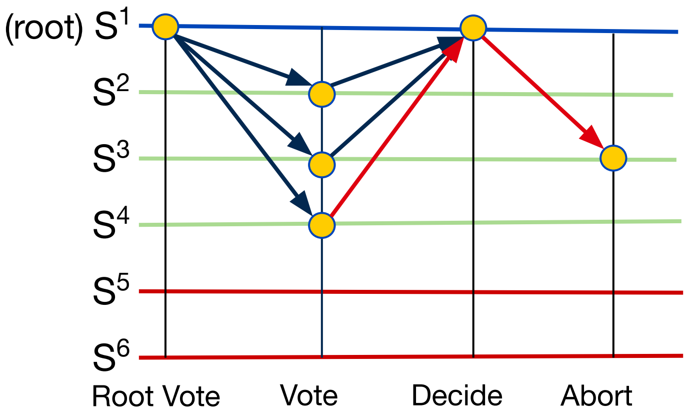
→ = Cluster-sending step

S^1, S^2, S^3, S^4 → Vote-steps

S^2, S^5, S^6 → Commit-steps

S^3 → Abort-step

Centralized orchestration



<u># of steps:</u>

Consecutive consensus steps = **4**

Consensus steps = $1 + (n_v - 1) + 1 + n_a = \mathbf{n_v + n_a + 1}$

Cluster-sending steps = $(n_v - 1) + (n_v - 1) + n_a = \mathbf{2(n_v - 1) + n_a}$

● = Consensus

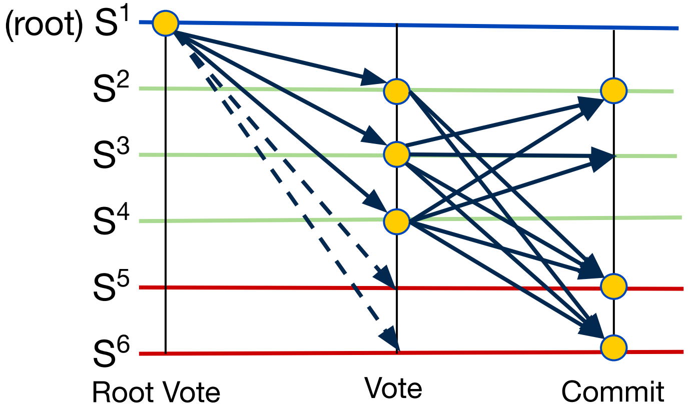
→ = Cluster-sending step

S^1, S^2, S^3, S^4 → Vote-steps

S^2, S^5, S^6 → Commit-steps

S^3 → Abort-step

Distributed orchestration



# of steps:

Consecutive consensus steps = **3**

Consensus steps = $1 + (n_v - 1) + n_c = \mathbf{n_v + n_c}$

Cluster-sending steps = $(n_v - 1) + (n_a + n_c) + (n_v - 1)*(n_a + n_c)$

$= \mathbf{n_v(n_a + n_c) + (n_v - 1)}$

● = Consensus

⟶ = Cluster-sending step

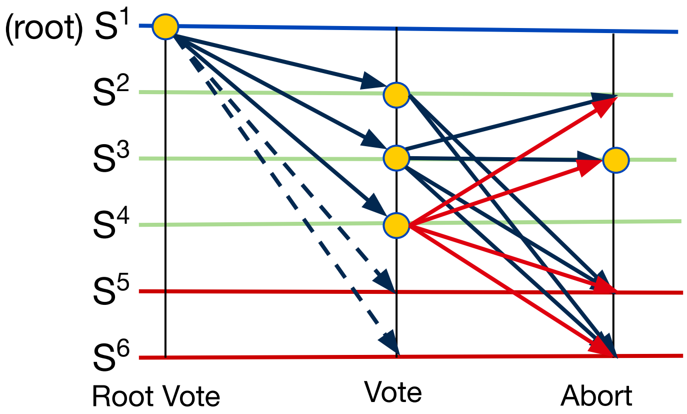
⇢ = Wait step

$\mathbf{S^1, S^2, S^3, S^4}$ → Vote-steps

$\mathbf{S^2, S^5, S^6}$ → Commit-steps

$\mathbf{S^3}$ → Abort-step

Distributed orchestration



<u># of steps:</u>

Consecutive consensus steps = **3**

Consensus steps = $1 + (n_v - 1) + n_a = \mathbf{n_v + n_a}$

Cluster-sending steps = $(n_v - 1) + (n_a + n_c) + (n_v - 1)*(n_a + n_c)$

$= \mathbf{n_v(n_a + n_c) + (n_v - 1)}$

● = Consensus

⟶ = Cluster-sending step

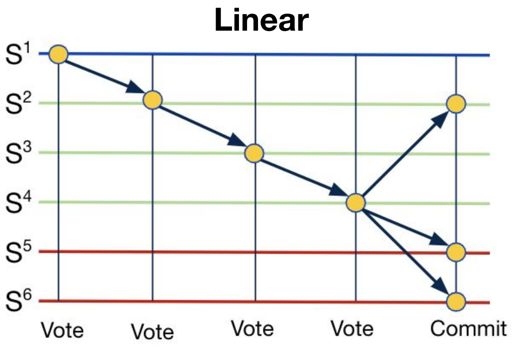
- - ⟶ = Wait step

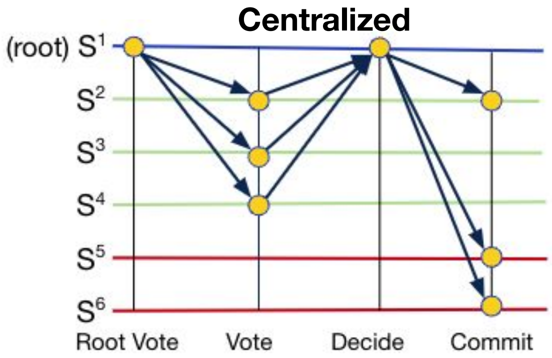
S^1, S^2, S^3, S^4 → Vote-steps

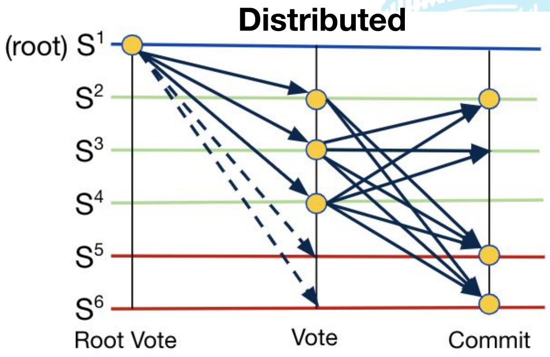
S^2, S^5, S^6 → Commit-steps

S^3 → Abort-step

Summary of Orchestration


Linear


Centralized


Distributed

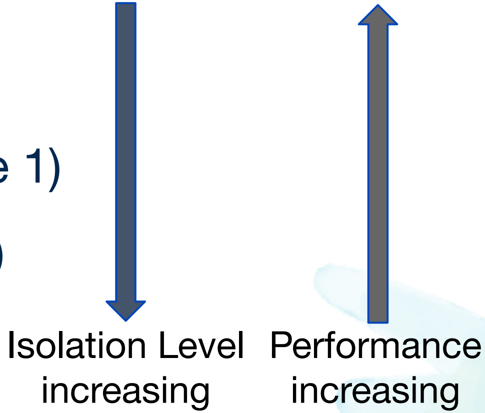
	Linear	Centralized	Distributed
Consecutive consensus steps	$n_v + 1$	4	3
Consensus steps	$n_v + n_c$	$n_v + n_c + 1$	$n_v + n_c$
Cluster-sending steps	$n_v + n_c - 1$	$2(n_v - 1) + n_c$	$n_v(n_a + n_c) + (n_v - 1)$

Execute

- Single Shard steps are **ordered** via consensus and executed **sequentially** at shard level.
- **Multi-shard transactions** can have several shard steps.
- Transactions can **interleave** while executing.
- Isolation necessary in some form of **concurrency control.**

Degrees of Isolation

- **Isolation free execution** (Degree 0)
- **Read uncommitted execution** (Degree 1)
- **Read committed execution** (Degree 2)
- **Serializable execution** (Degree 3)



Isolation Level increasing

Performance increasing

NOTE: Stronger Isolation levels prevent more anomalies at the cost of performance.

Constraint and Modification model

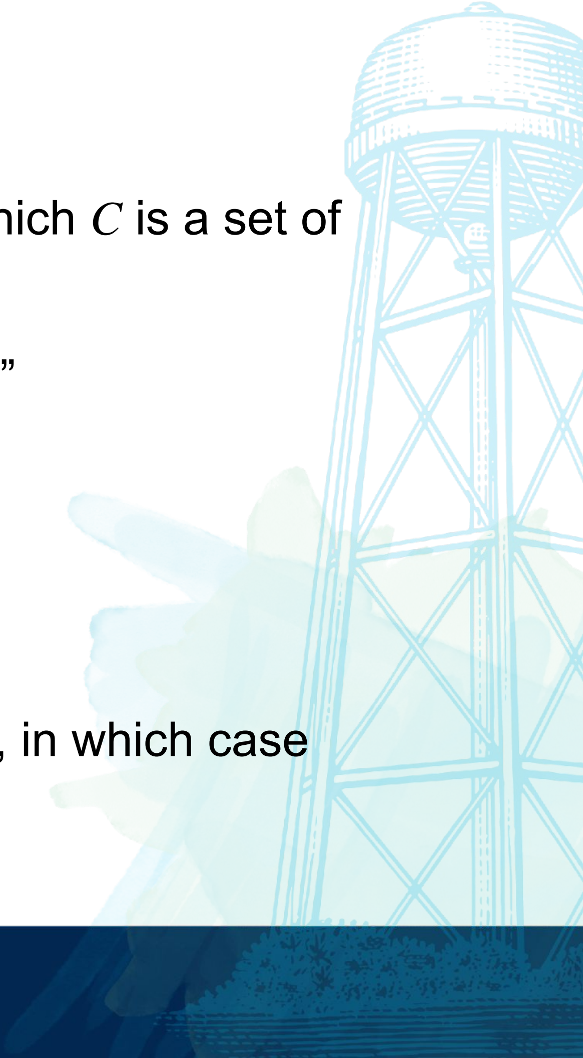
We assume that each transaction $\boldsymbol{\tau}$ is a pair (C, M) in which C is a set of constraints of the form

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

and M a set of modifications of the form

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

The system commits to $\boldsymbol{\tau}$ only if all constraints in C hold, in which case all modifications in M are applied to the system



Example

Considering the sharded banking example from before. And assuming the system does **not allow negative** accounts balances and consider transactions.

τ_1 = Con(A, 100), Con(B, 700), Mod(A, 400), Mod(B, -400);

Transfer 400 from B to A if A has 100 and B has 700

τ_2 = Con(A, 500), Mod(A, -300), Mod(E, 300);

Transfer 300 from A to E if A has 500

Example

Considering the sharded banking example from before. And assuming the system does **not allow negative** accounts balances and consider transactions.

τ_1 = Con(A, 100), Con(B, 700), Mod(A, 400), Mod(B, -400);

Transfer 400 from B to A if A has 100 and B has 700

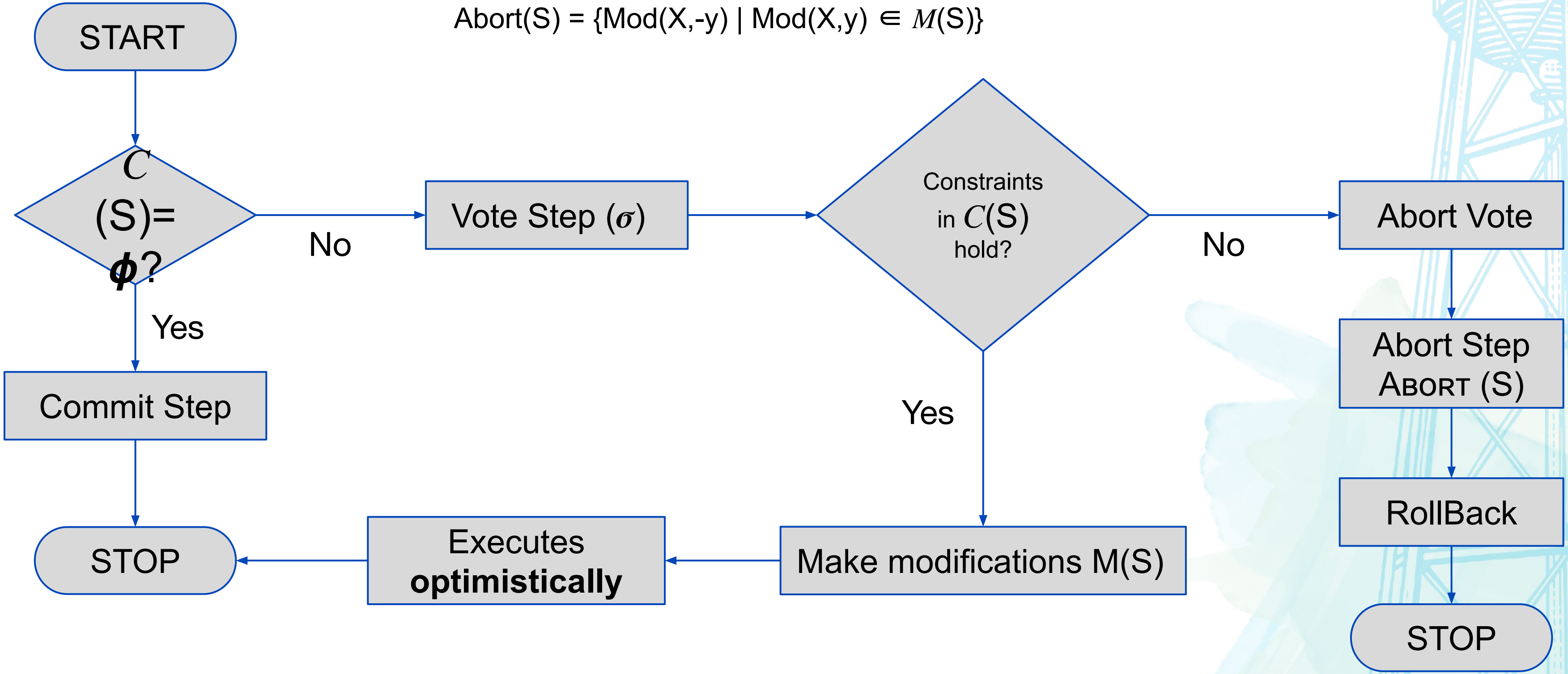
τ_2 = Con(A, 500), Mod(A, -300), Mod(E, 300);

Transfer 300 from A to E if A has 500

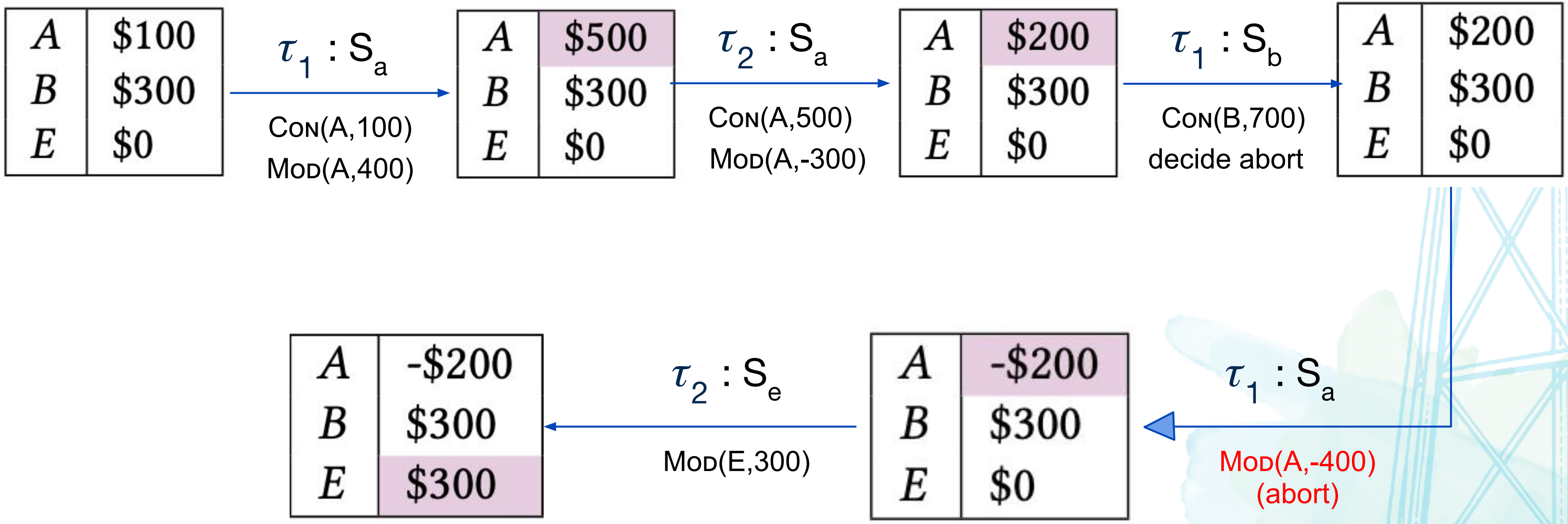
Isolation free direct execution

Let, $\tau = (C, M)$
$S \in \text{shards}(\tau)$

Abort(S) = {Mod(X,-y) | Mod(X,y) ∈ M(S)}



Example



A	\$100
B	\$300
E	\$0

τ_1 : S$_a$

Con(A,100)
Mod(A,400)

A	\$500
B	\$300
E	\$0

τ_2 : S$_a$

Con(A,500)
Mod(A,-300)

A	\$200
B	\$300
E	\$0

τ_1 : S$_b$

Con(B,700)
decide abort

A	\$200
B	\$300
E	\$0

τ_1 : S$_a$

Mod(A,-400)
(abort)

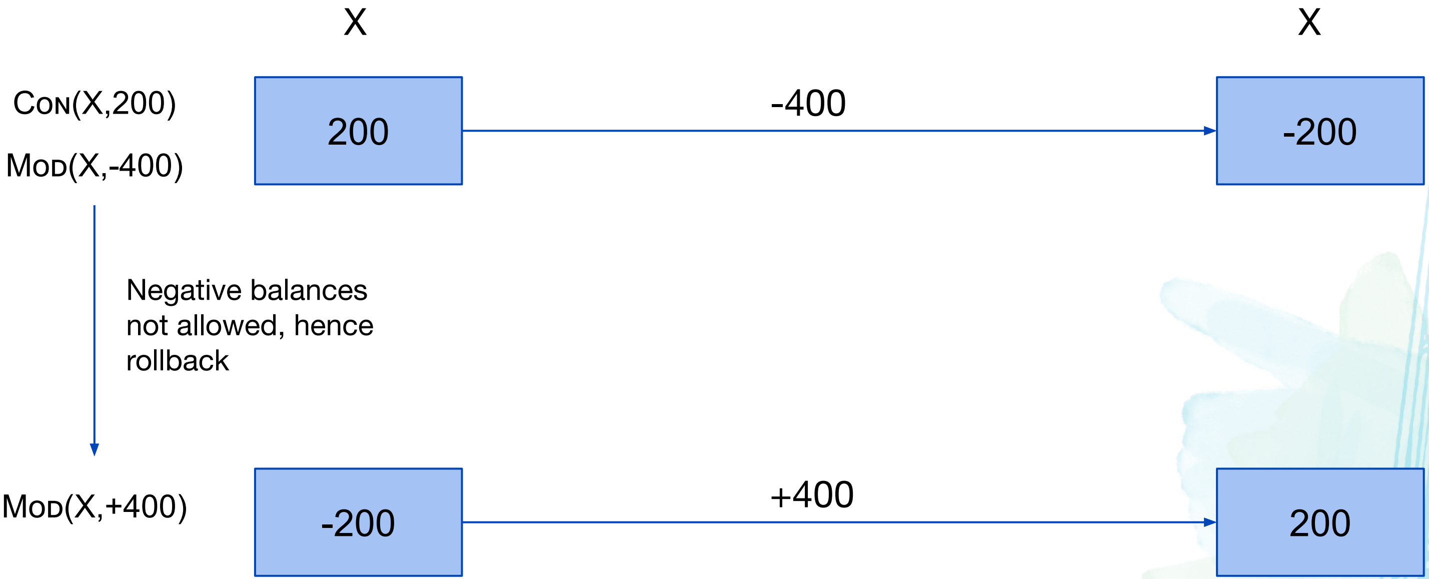
A	-\$200
B	\$300
E	\$0

τ_2 : S$_e$

Mod(E,300)

A	-\$200
B	\$300
E	\$300

Safe Transaction



X

X

Con(X,200)

Mod(X,-400)

200

-400

-200

Negative balances not allowed, hence rollback

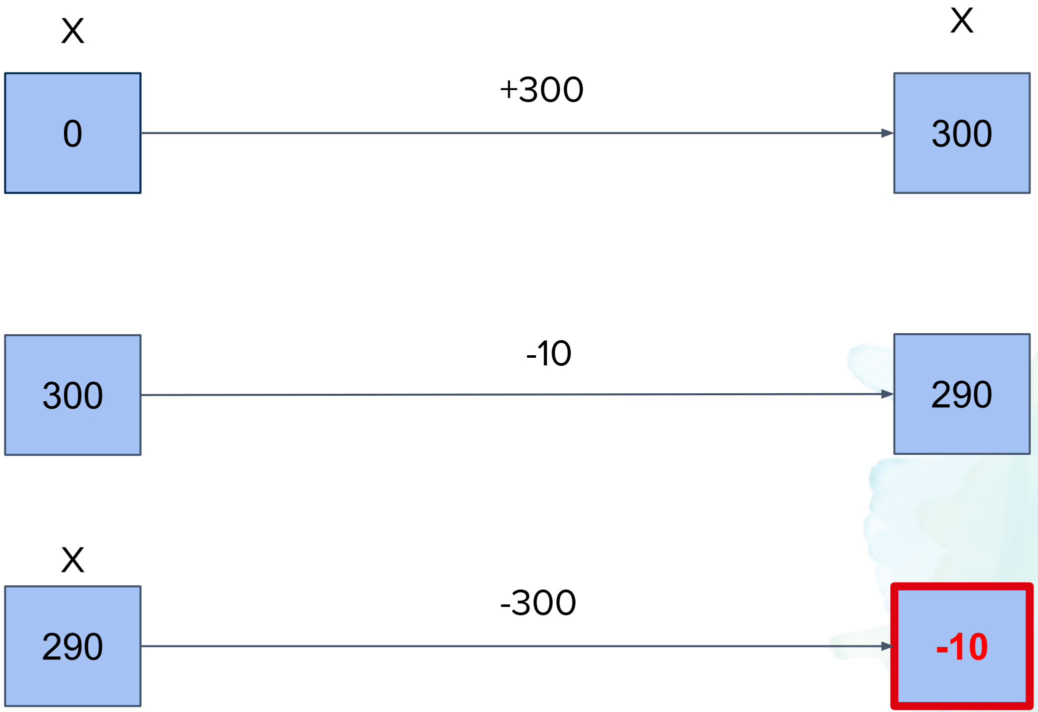
Mod(X,+400)

-200

+400

200

UnSafe Transaction



Mod(X,300)
for τ_1

X

0

+300

X

300

Expenditure τ_2

300

-10

290

Rollback due to
τ_1 constraint
failure:
Mod(X, -300)

X

290

-300

-10

Safe Isolation Free Execution

- **Safe modifications** are executed as part of **Vote step**

- **Unsafe modifications** are executed as part of **Commit Step**

Lock Based Execution

- if τ' where, $\tau' \neq \tau$, holds a write lock on D, then τ cannot obtain any locks on D.
- if τ' where, $\tau' \neq \tau$, holds a read lock on D, then τ cannot obtain a write lock on D, but can obtain a read lock on D.
- **Several transactions can hold a read lock on D at the same time, but write locks are exclusive.**

Lock Based Execution

- Let τ = "if Ana has $500 and Bo has $300 then

  move $200 from Ana to Ben".

- Assume shards are ordered as S_a,……,S_z, accounts are ordered on account holder name.
- Write lock on the account of Ana in S_a, write lock on the account of Ben in S_b, read lock on the account of Bo in S_b.

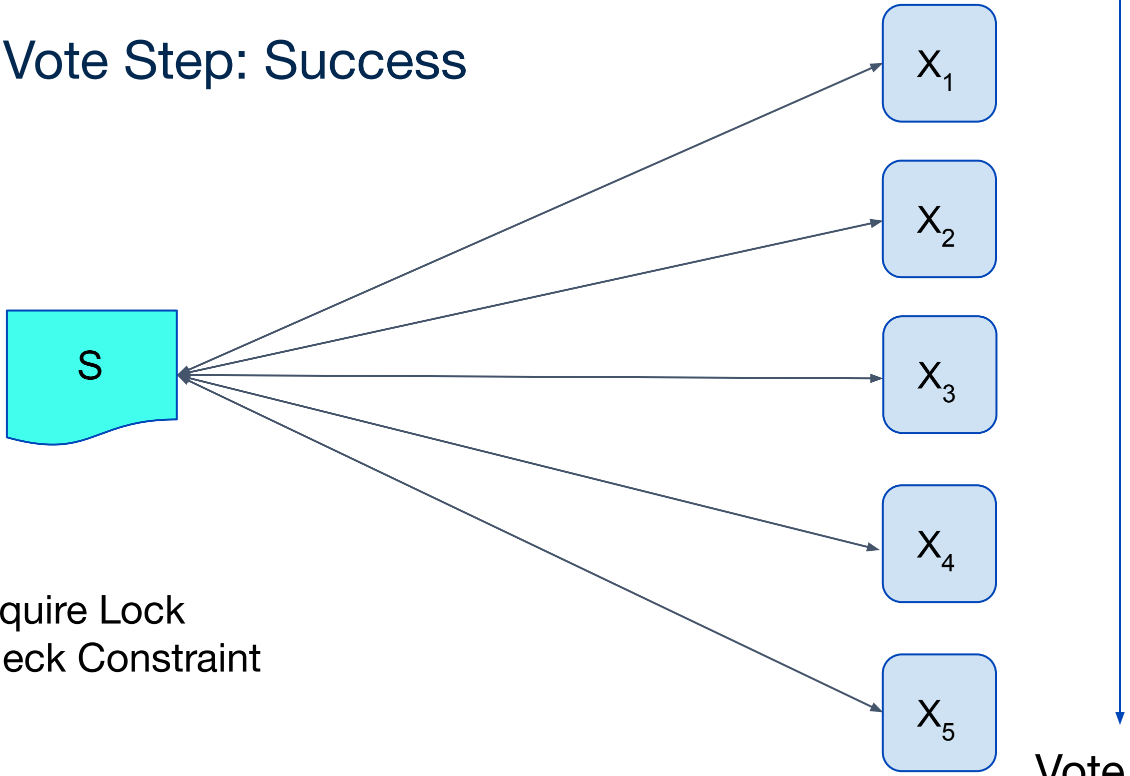
Lock Based Execution

- Let $\tau = (C, M)$ be a transaction, let S ∈ shards(τ), and

  Accounts(S) = $\{X \mid$ CON(X, y) ∈ C(S) ∨ MOD(X, y) ∈ M(S)$\}$
- Steps involved:
  - Vote-step
  - Commit Step
  - Release step

Lock Based Execution

- Vote Step: Success



τ = (C,M)
S = shards(τ)
X_i=ACCOUNTS(S)

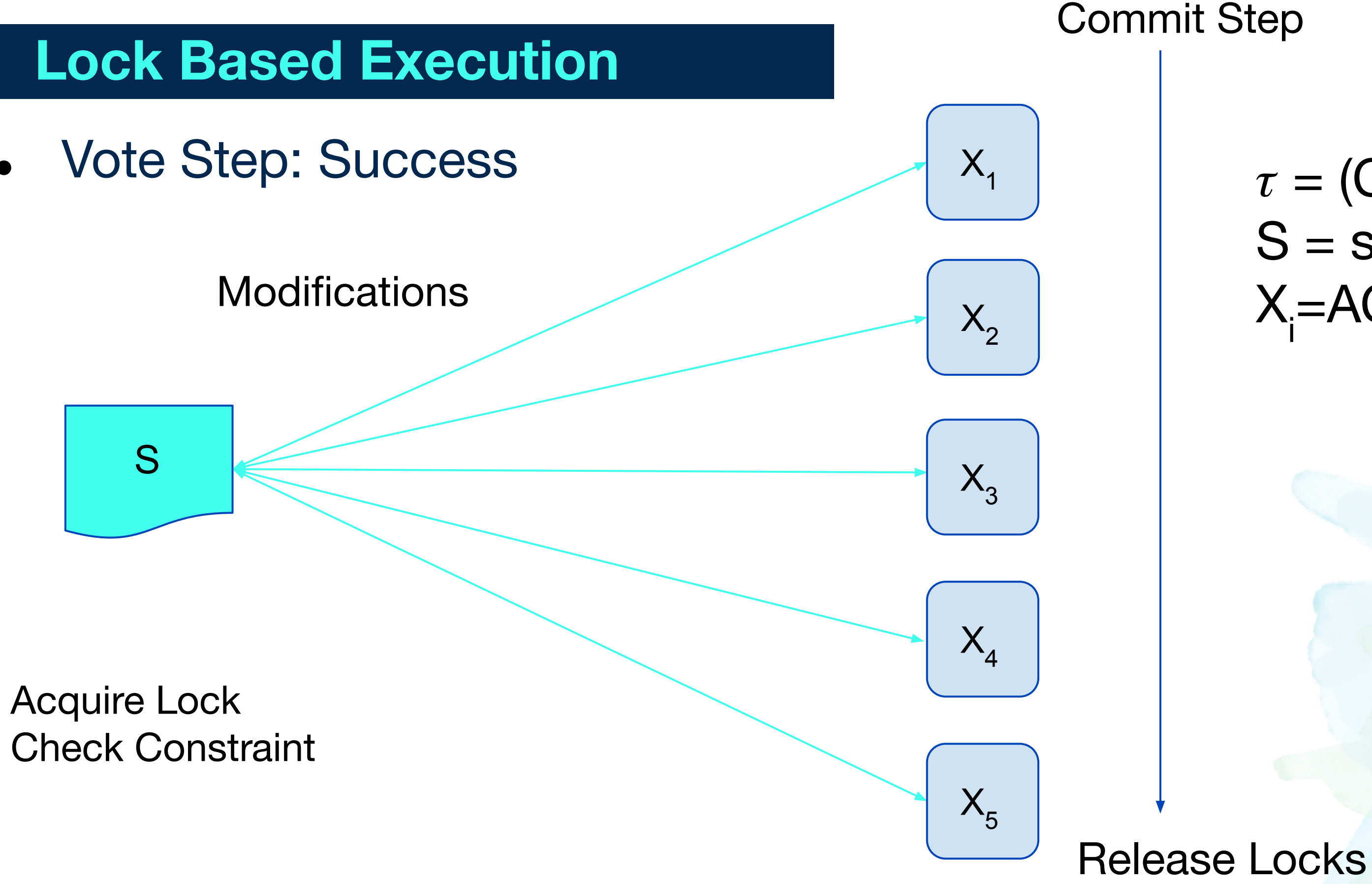
Acquire Lock
Check Constraint

Vote Commit

UCDAVIS

Lock Based Execution

- Vote Step: Success



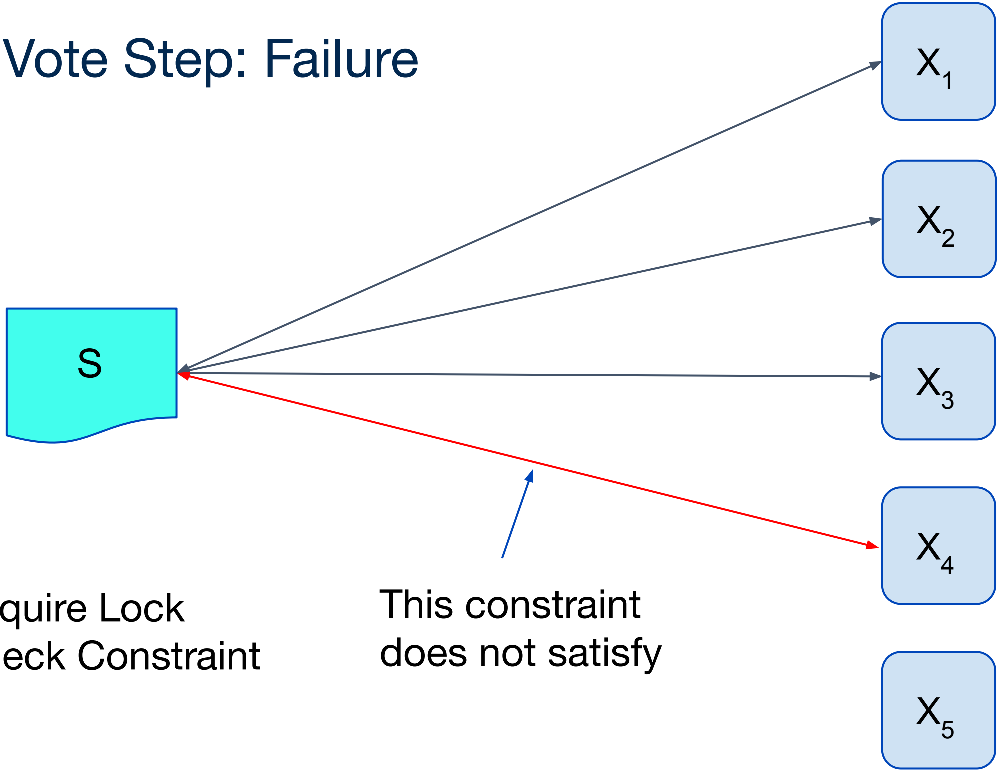
τ = (C,M)

S = shards(τ)

X_i=ACCOUNTS(S)

Lock Based Execution

- Vote Step: Failure



Acquire Lock
Check Constraint

This constraint
does not satisfy

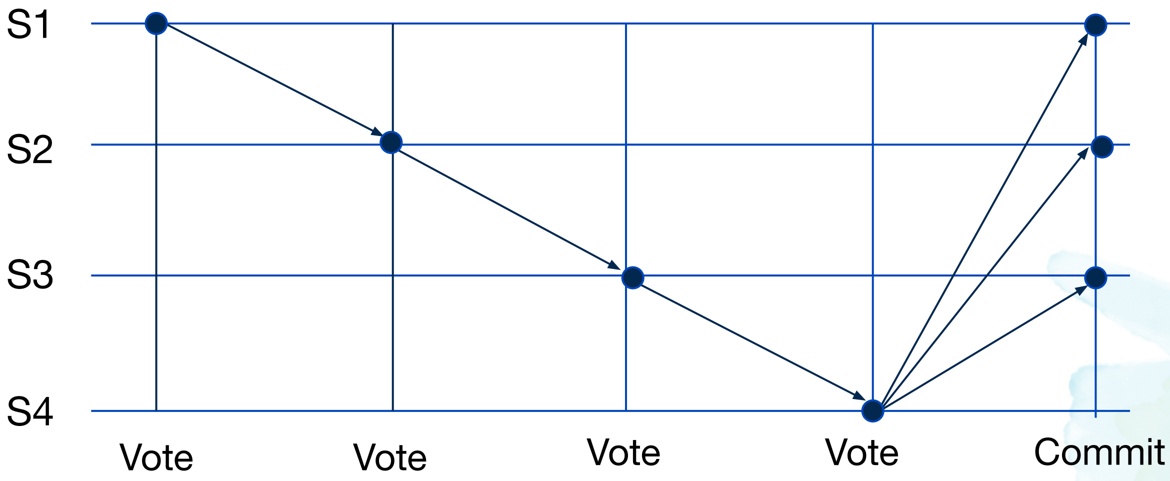
τ = (C,M)
S = shards(τ)
X_i=ACCOUNTS(S)

Vote abort, locks are released

UCDAVIS

Lock Based Execution

Transaction τ with n = |shards(τ)|



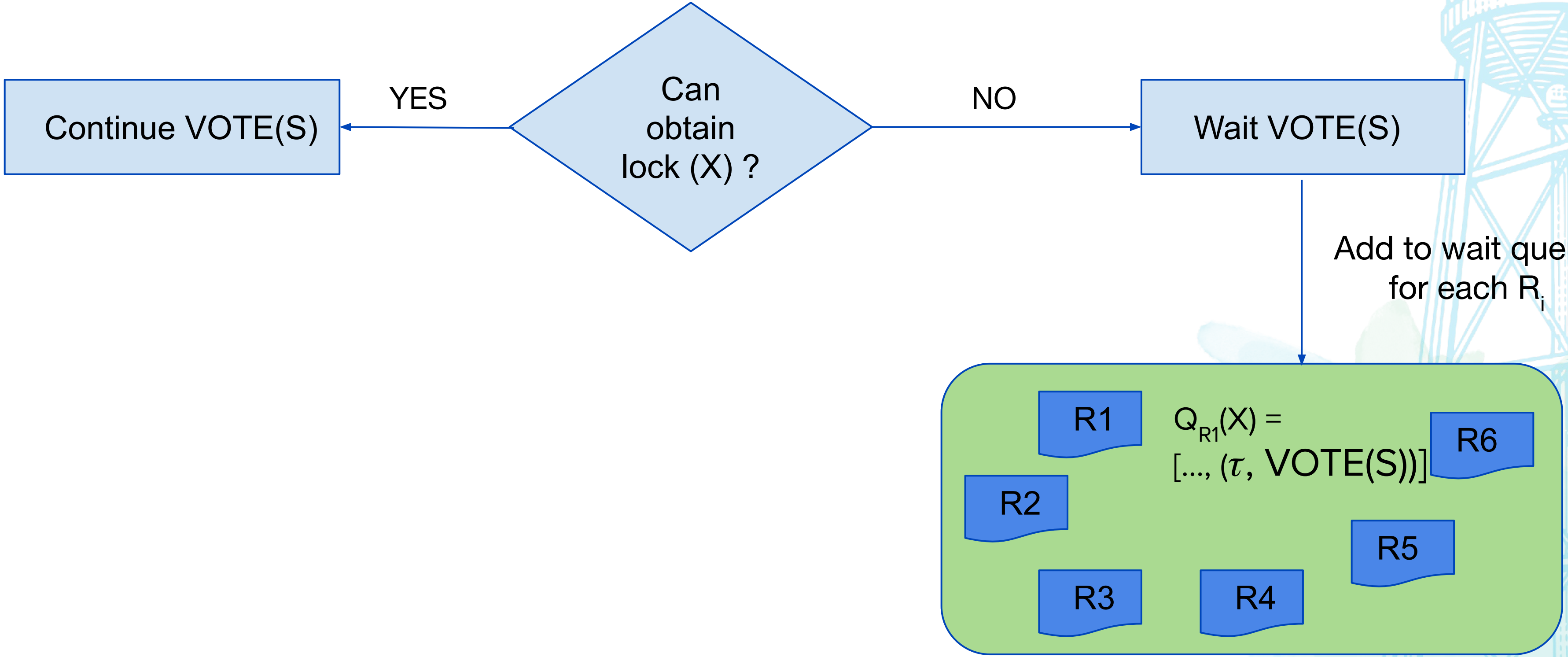
● = Consensus step

→ = Cluster-sending step

Consensus Steps = n + (n-1) = **2n - 1**

Cluster-sending steps = (n-1) + (n-1) = **2n - 2**

Lock Based Execution

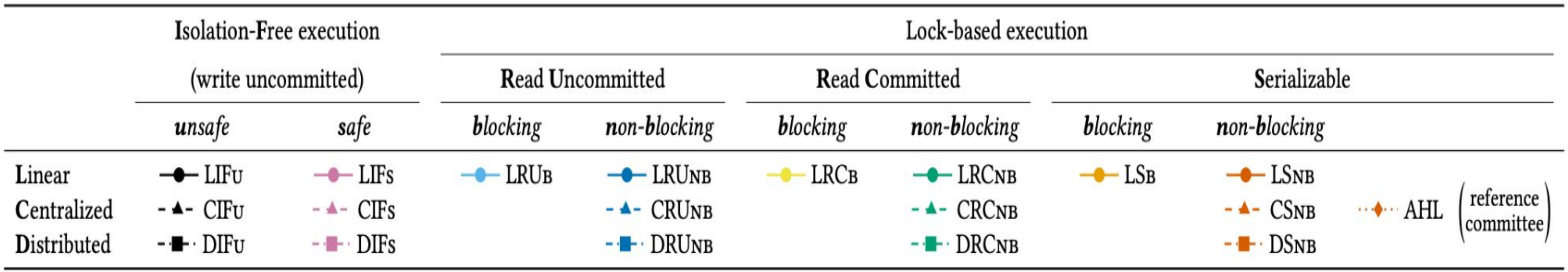


Protocols

We have **18 different protocols** that emerge from different configurations of the below three parameters:

- Linear, Centralized, and Distributed Orchestration
- Four Isolation Degrees in Execution
- Blocking and Non-blocking locks

Protocols



	Isolation-Free execution		Lock-based execution						
	(write uncommitted)		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	

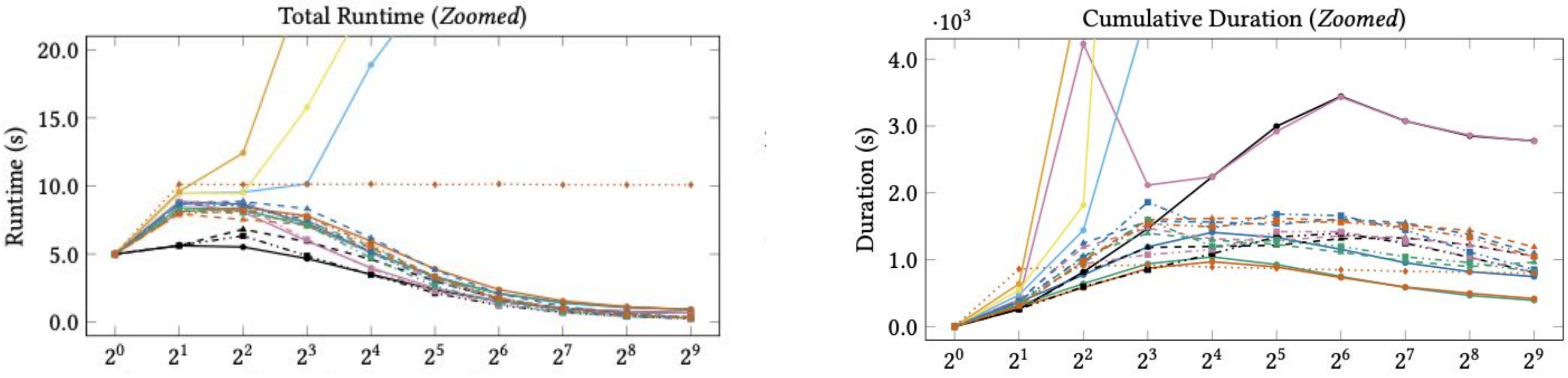
UCDAVIS

Experimental Setup

- Workload of 5000 transactions
- Each transaction affects 16 distinct accounts
  - Putting constraints on 8 accounts (read operations)
  - Removing balance from 4 accounts (write operations)
  - Adding balance to 4 accounts (write operations)
- Each account on each shard
  - Initial balance of 2000
  - Add or remove 500 balance per modification
- 64 shards and 8192 active accounts (128/shard)

Performance Evaluation: Results : Scalability

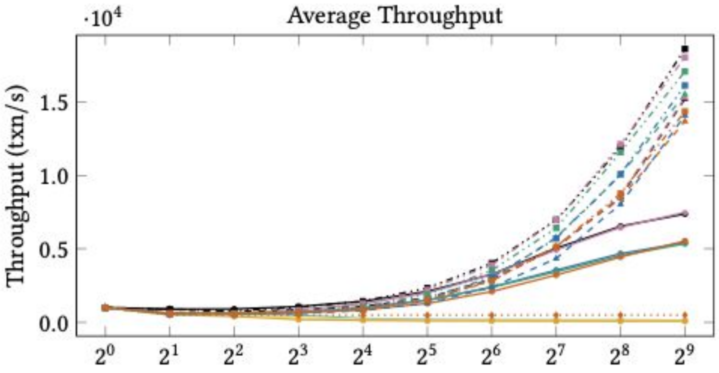
	Isolation-Free execution (write uncommitted)		Lock-based execution						
			Read Uncommitted		Read Committed		Serializable		
	unsafe	*safe*	*blocking*	*non-blocking*	*blocking*	*non-blocking*	*blocking*	*non-blocking*	
Linear	—●— LIFU	—●— LIFs	—●— LRUB	—●— LRUNB	—●— LRCB	—●— LRCNB	—●— LSB	—●— LSNB	
Centralized	-▲- CIFU	-▲- CIFs		-▲- CRUNB		-▲- CRCNB		-▲- CSNB	···◆··· AHL (reference committee)
Distributed	-■· DIFU	-■· DIFs		-■· DRUNB		-■· DRCNB		-■· DSNB	

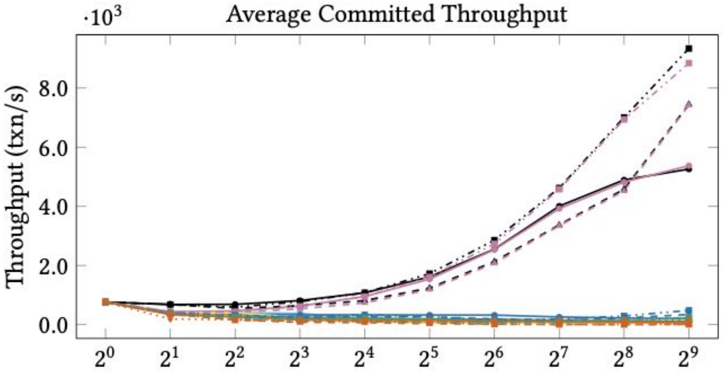


Hellings, Jelle, and Mohammad Sadoghi. "Byshard: Sharding in a byzantine environment." Proceedings of the VLDB Endowment 14.11 (2021): 2230-2243.

Performance Evaluation: Results : Scalability

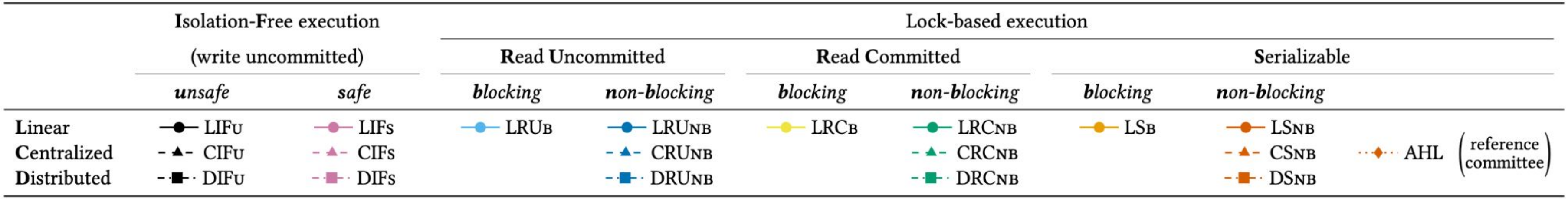
	Isolation-Free execution		Lock-based execution						
	(write uncommitted)		Read Uncommitted		Read Committed		Serializable		
	unsafe	***safe***	***blocking***	***non-blocking***	***blocking***	***non-blocking***	***blocking***	***non-blocking***	
Linear	● LIFU	● LIFS	● LRUB	● LRUNB	● LRCB	● LRCNB	● LSB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



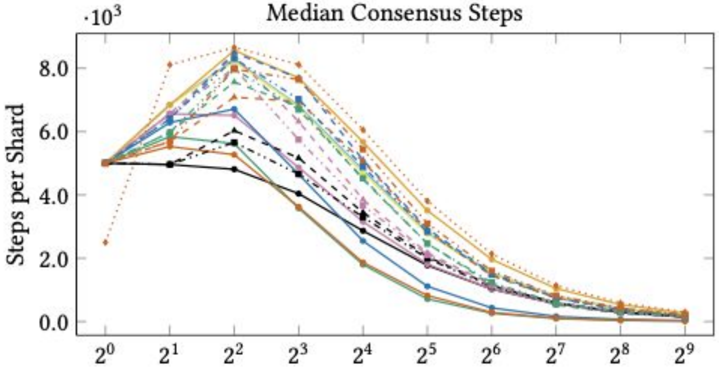


Hellings, Jelle, and Mohammad Sadoghi. "Byshard: Sharding in a byzantine environment." Proceedings of the VLDB Endowment 14.11 (2021): 2230-2243.

Performance Evaluation: Results : Scalability

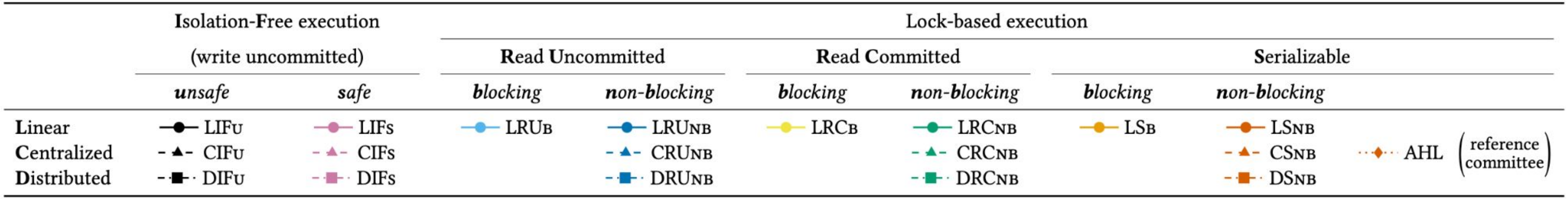


	Isolation-Free execution		Lock-based execution						
	(write uncommitted)		Read Uncommitted		Read Committed		Serializable		
	unsafe	*safe*	*blocking*	*non-blocking*	*blocking*	*non-blocking*	*blocking*	*non-blocking*	
Linear	—●— LIFU	—●— LIFs	—●— LRUB	—●— LRUNB	—●— LRCB	—●— LRCNB	—●— LSB	—●— LSNB	
Centralized	-▲- CIFU	-▲- CIFs		-▲- CRUNB		-▲- CRCNB		-▲- CSNB	··◆·· AHL (reference committee)
Distributed	-■- DIFU	-■- DIFs		-■- DRUNB		-■- DRCNB		-■- DSNB	

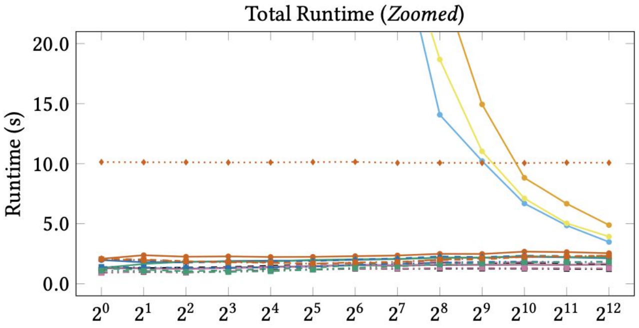


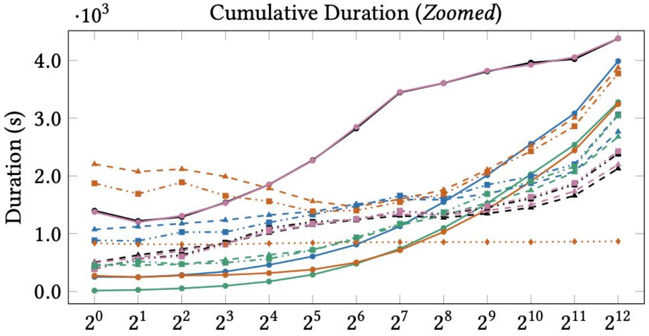
Hellings, Jelle, and Mohammad Sadoghi. "Byshard: Sharding in a byzantine environment." Proceedings of the VLDB Endowment 14.11 (2021): 2230-2243.

Performance Evaluation: Results : Contention



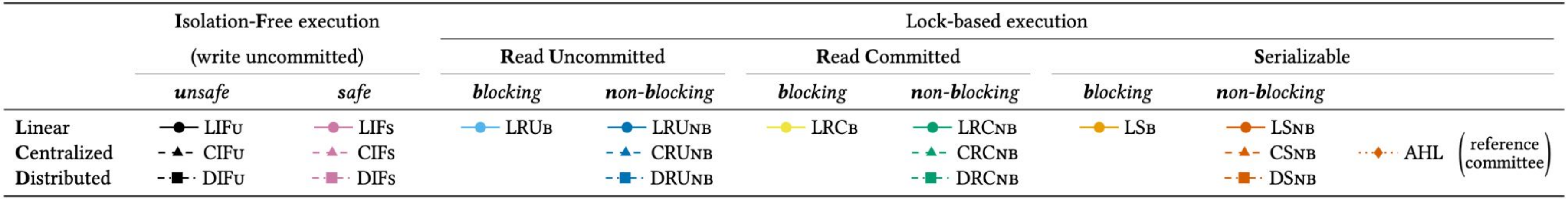
	Isolation-Free execution (write uncommitted)		Lock-based execution						
			Read Uncommitted		Read Committed		Serializable		
	unsafe	*safe*	*blocking*	*non-blocking*	*blocking*	*non-blocking*	*blocking*	*non-blocking*	
Linear	● LIFU	● LIFS	● LRUB	● LRUNB	● LRCB	● LRCNB	● LSB	● LSNB	
Centralized	▲ CIFU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



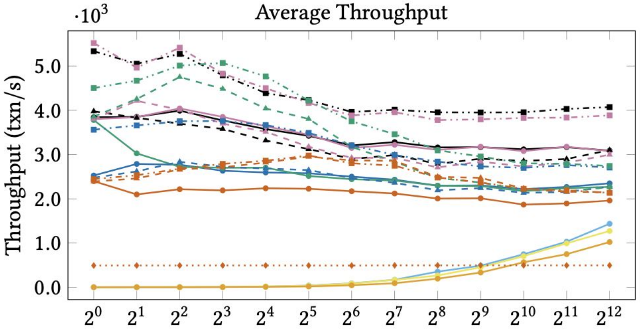


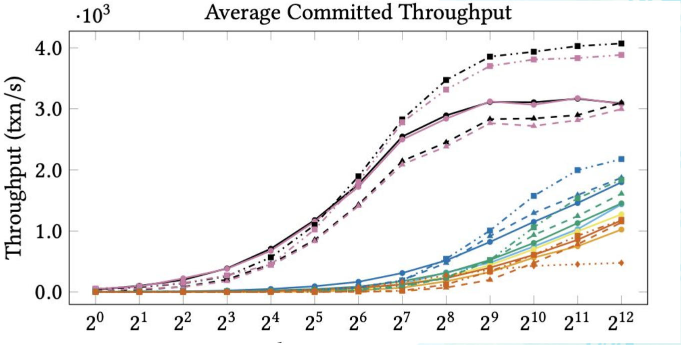
Hellings, Jelle, and Mohammad Sadoghi. "Byshard: Sharding in a byzantine environment." Proceedings of the VLDB Endowment 14.11 (2021): 2230-2243.

Performance Evaluation: Results : Contention



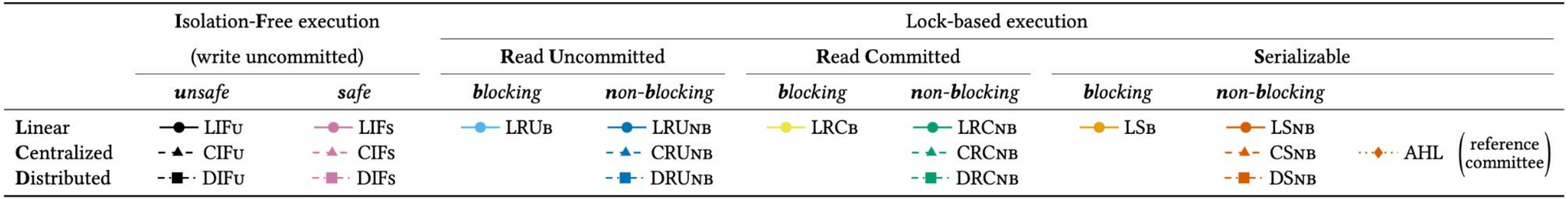
	Isolation-Free execution (write uncommitted)		Lock-based execution						
			Read Uncommitted		Read Committed		Serializable		
	unsafe	*safe*	*blocking*	*non-blocking*	*blocking*	*non-blocking*	*blocking*	*non-blocking*	
Linear	LIFU	LIFS	LRUB	LRUNB	LRCB	LRCNB	LSB	LSNB	
Centralized	CIFU	CIFS		CRUNB		CRCNB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



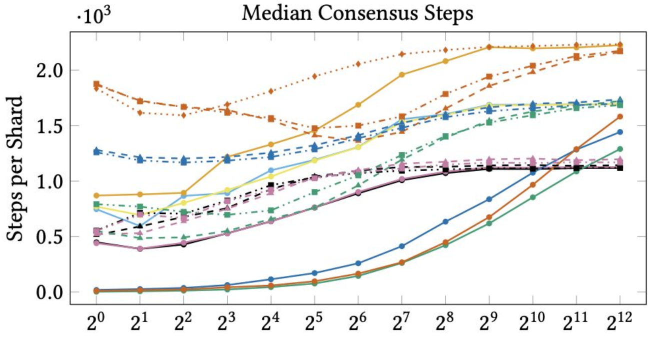


Hellings, Jelle, and Mohammad Sadoghi. "Byshard: Sharding in a byzantine environment." Proceedings of the VLDB Endowment 14.11 (2021): 2230-2243.

Performance Evaluation: Results : Contention



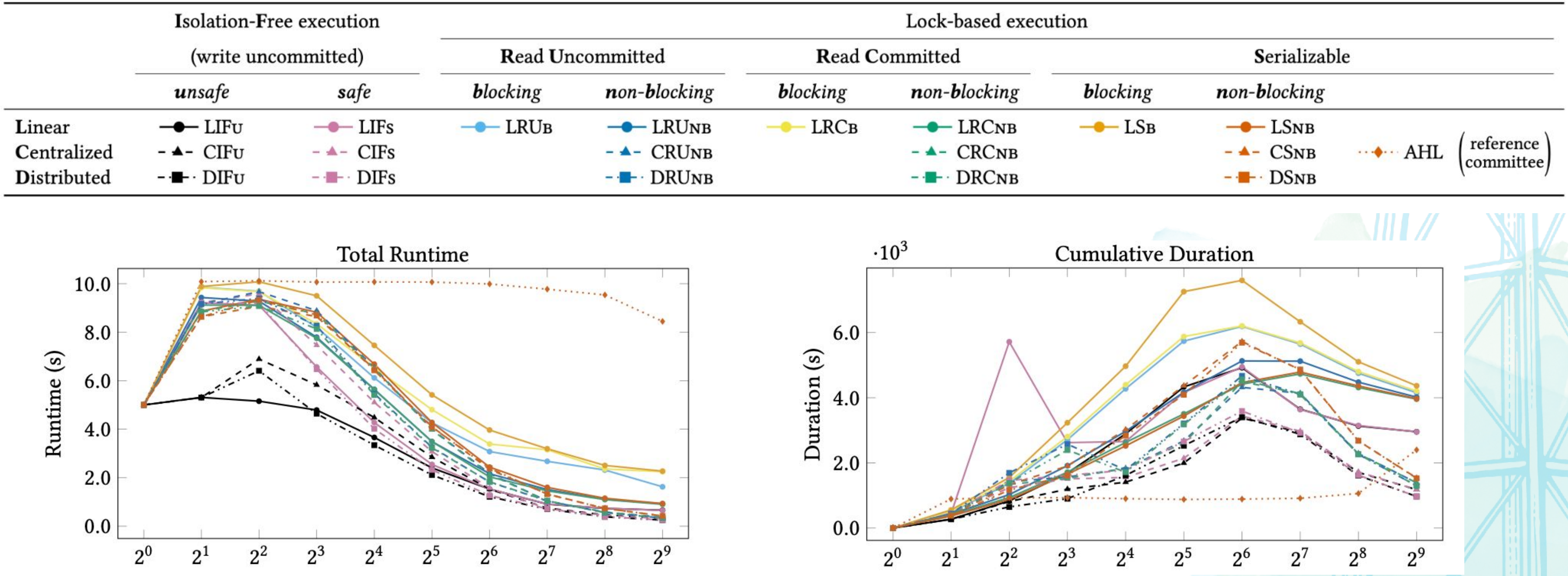
	Isolation-Free execution		Lock-based execution						
	(write uncommitted)		Read Uncommitted		Read Committed		Serializable		
	unsafe	*safe*	*blocking*	*non-blocking*	*blocking*	*non-blocking*	*blocking*	*non-blocking*	
Linear	● LIFU	● LIFS	● LRUB	● LRUNB	● LRCB	● LRCNB	● LSB	● LSNB	
Centralized	▲ CIFU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



Median Consensus Steps

Hellings, Jelle, and Mohammad Sadoghi. "Byshard: Sharding in a byzantine environment." Proceedings of the VLDB Endowment 14.11 (2021): 2230-2243.

Performance Evaluation: Results : Factor-Scalability



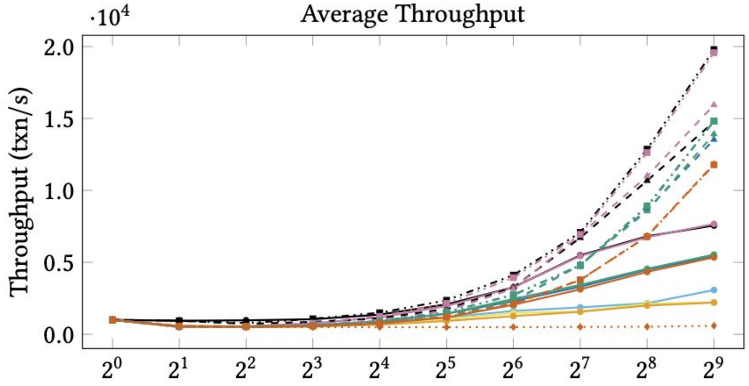
	Isolation-Free execution (write uncommitted)		Lock-based execution						
			Read Uncommitted		Read Committed		Serializable		
	unsafe	*safe*	*blocking*	*non-blocking*	*blocking*	*non-blocking*	*blocking*	*non-blocking*	
Linear	● LIFU	● LIFs	● LRUB	● LRUNB	● LRCB	● LRCNB	● LSB	● LSNB	
Centralized	▲ CIFU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

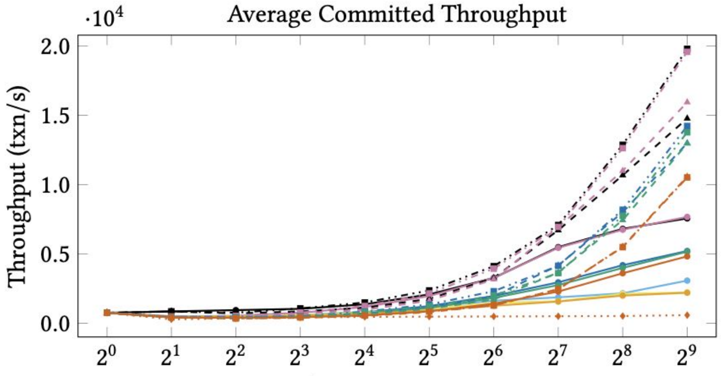
UCDAVIS

Hellings, Jelle, and Mohammad Sadoghi. "Byshard: Sharding in a byzantine environment." Proceedings of the VLDB Endowment 14.11 (2021): 2230-2243.

Performance Evaluation: Results : Factor-Scalability

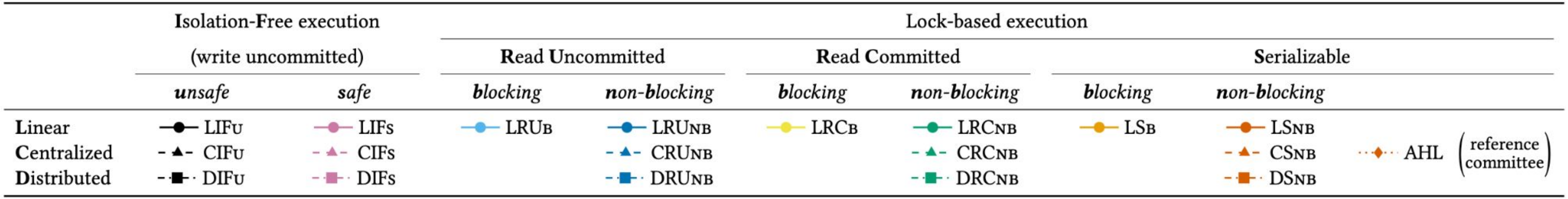
	Isolation-Free execution (write uncommitted)		Lock-based execution						
			Read Uncommitted		Read Committed		Serializable		
	unsafe	*safe*	*blocking*	*non-blocking*	*blocking*	*non-blocking*	*blocking*	*non-blocking*	
Linear	─●─ LIFU	─●─ LIFS	─●─ LRUB	─●─ LRUNB	─●─ LRCB	─●─ LRCNB	─●─ LSB	─●─ LSNB	
Centralized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



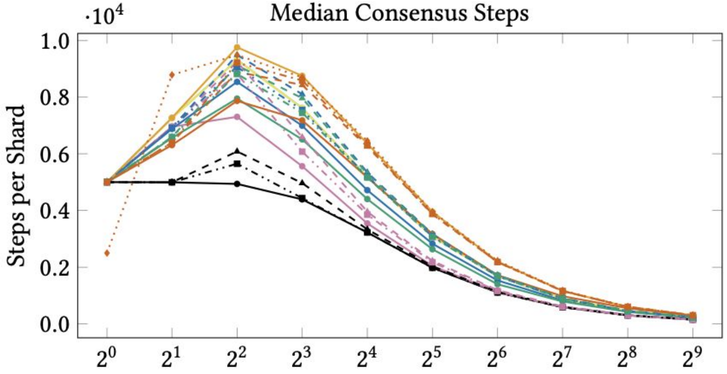


Hellings, Jelle, and Mohammad Sadoghi. "Byshard: Sharding in a byzantine environment." Proceedings of the VLDB Endowment 14.11 (2021): 2230-2243.

Performance Evaluation: Results : Factor-Scalability



	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	



Hellings, Jelle, and Mohammad Sadoghi. "Byshard: Sharding in a byzantine environment." Proceedings of the VLDB Endowment 14.11 (2021): 2230-2243.

Q&A

THANK YOU!