UNIVERSITY OF CAMBRIDGE

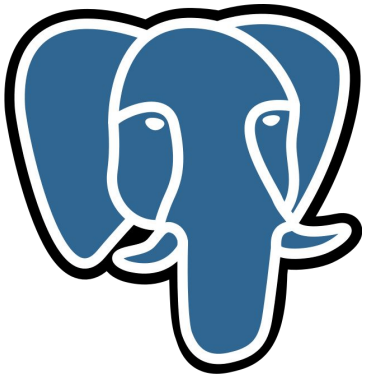
Unanimous 2PC

Chris Jensen, Heidi Howard,
Antonis Katsarakis, Richard Mortier

chris.jensen@cl.cam.ac.uk

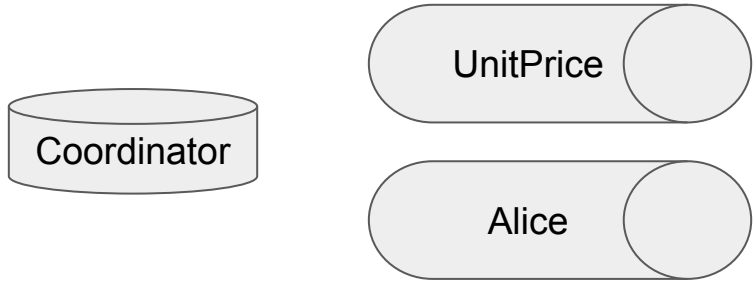
github.com/Cjen1/u2pc-tla/

Sharded datastores require distributed transactions



Execution Phase

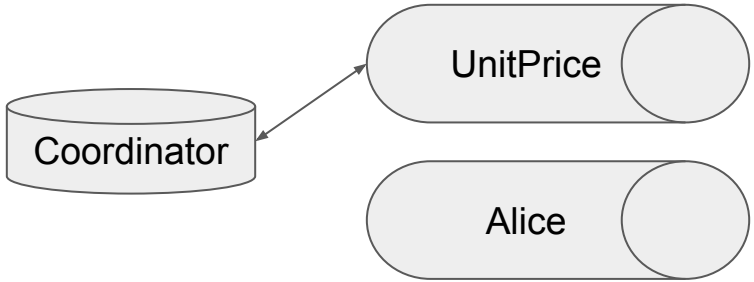
```
txn = Kvs.Create()
```



Read		Modify		

Execution Phase

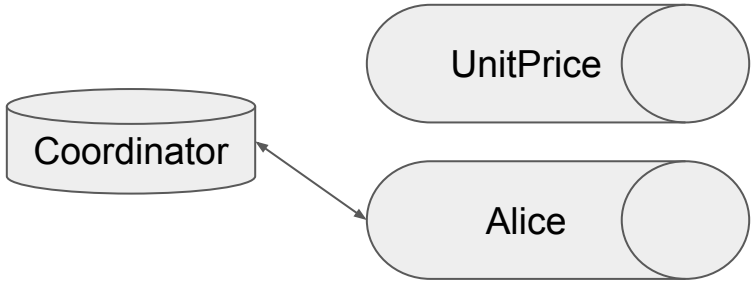
```
txn = Kvs.Create()

u = txn.Read('UnitPrice')
```



Read		Modify		
UnitPrice	$1			

Execution Phase

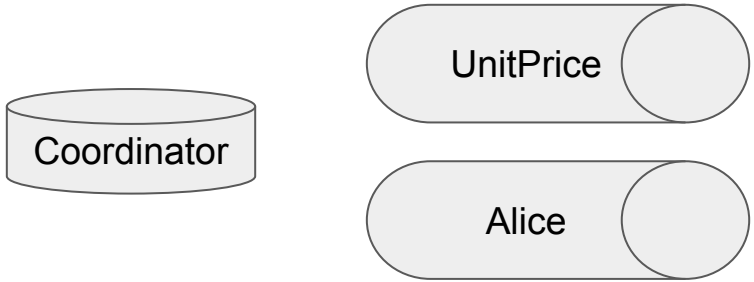
```
txn = Kvs.Create()

u = txn.Read('UnitPrice')

a = txn.Read('Alice')
```



Read		Modify		
UnitPrice	$1			
Alice	$10			

Execution Phase

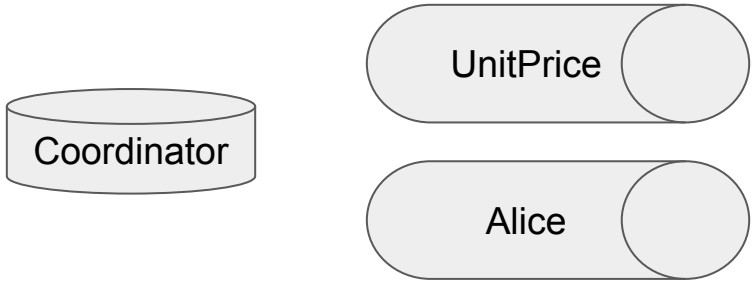
```
txn = Kvs.Create()

u = txn.Read('UnitPrice')

a = txn.Read('Alice')
if a - u < 0: txn.Abort()
```



Read		Modify		
UnitPrice	$1			
Alice	$10			

Execution Phase

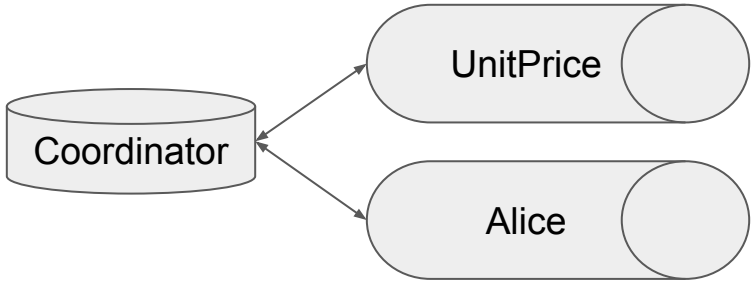
```
txn = Kvs.Create()

u = txn.Read('UnitPrice')

a = txn.Read('Alice')
if a - u < 0: txn.Abort()

txn.Write('Alice', a - u)
```



Read		Modify		
UnitPrice	$1	Alice	Write	$9
Alice	$10			

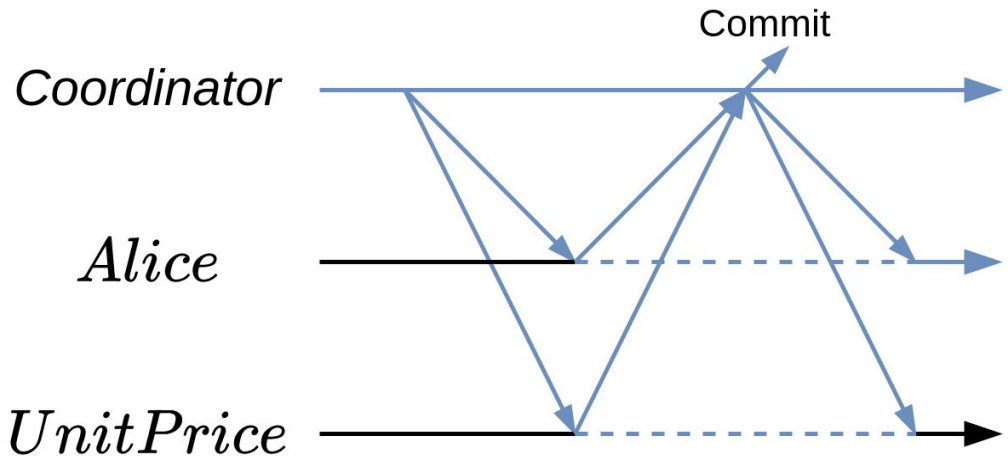
Execution Phase

```
txn = Kvs.Create()

u = txn.Read('UnitPrice')

a = txn.Read('Alice')
if a - u < 0: txn.Abort()

txn.Write('Alice', a - u)

return txn.Commit()
```

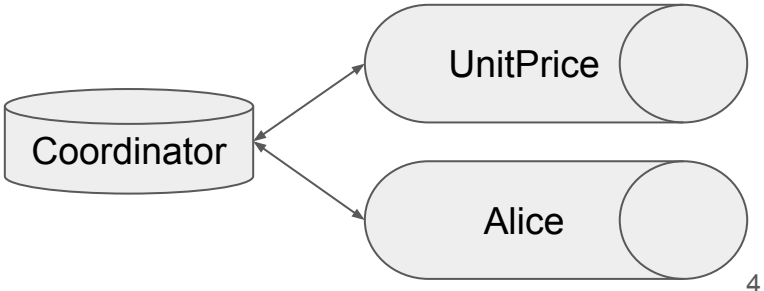


Read		Modify		
UnitPrice	$1	Alice	Write	$9
Alice	$10			

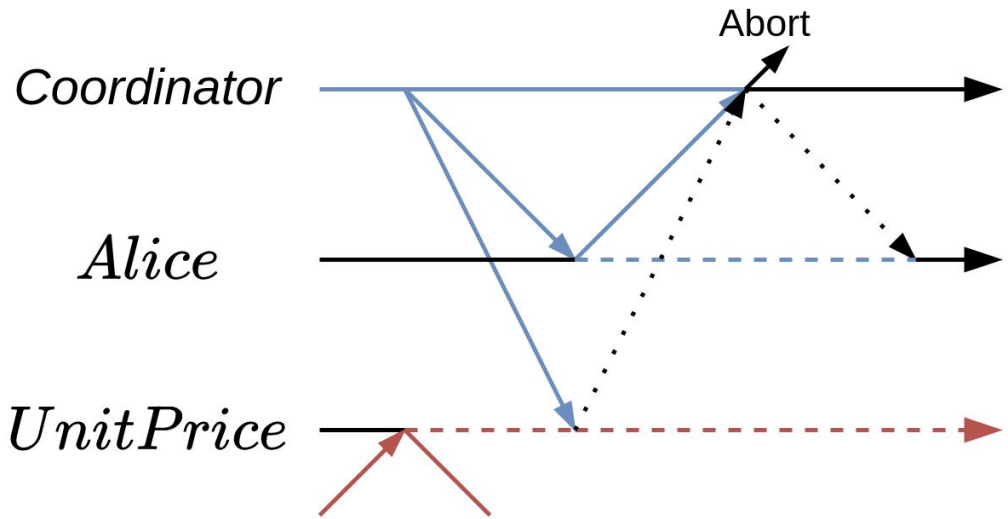
Commit Phase: 2 Phase Commit



Read		Modify		
UnitPrice	$1	Alice	Write	$9
Alice	$10			

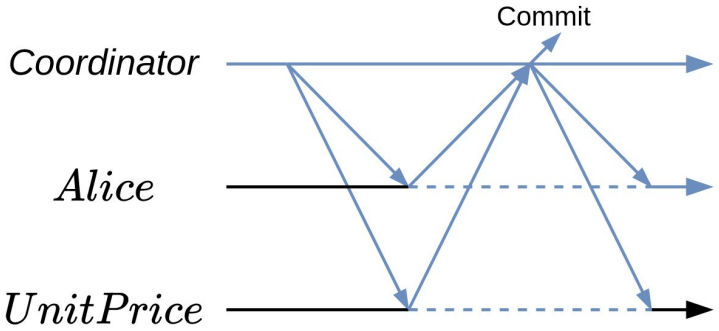


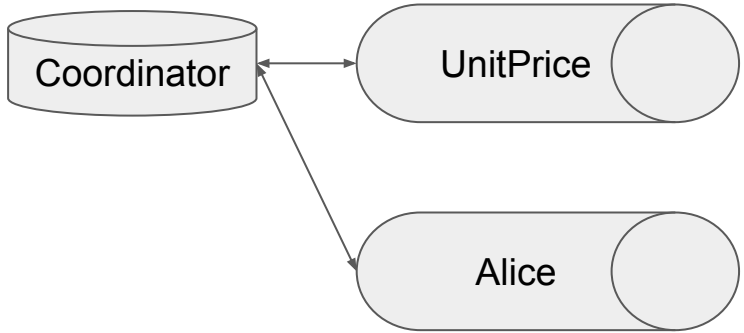
2 Phase Commit - Abort



2PC - Scalable but not fault tolerant

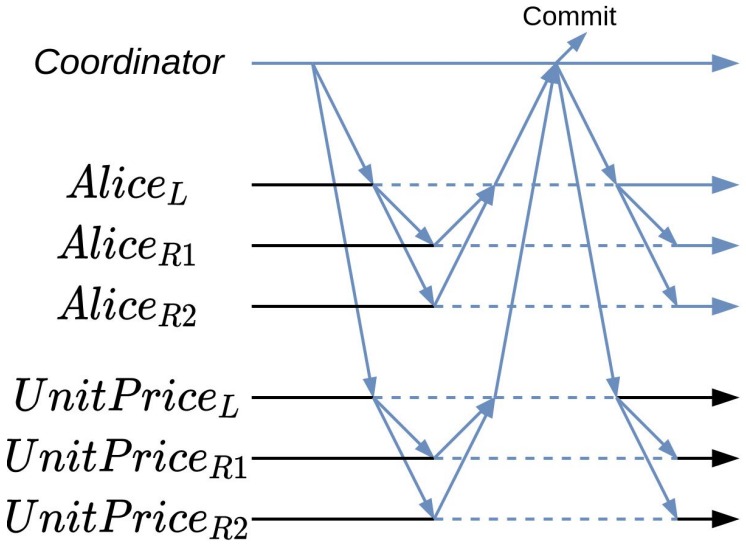
	Shard size	Write (msg)	Read (msg)	Commit (RTT)	Abort (RTT)
2PC	1	3	3	1	1

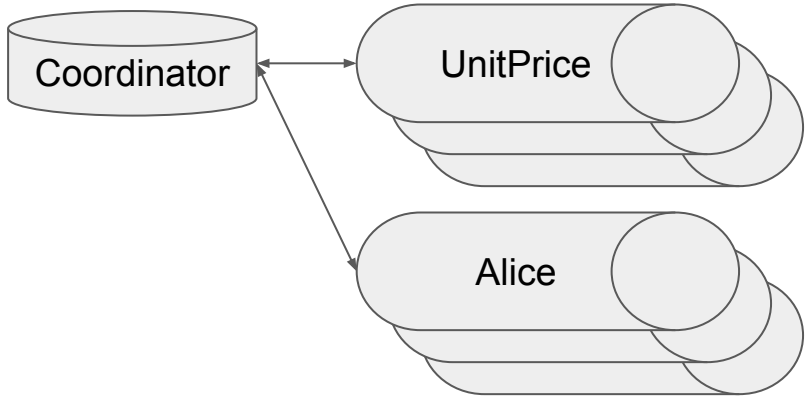




Durable shards via Paxos

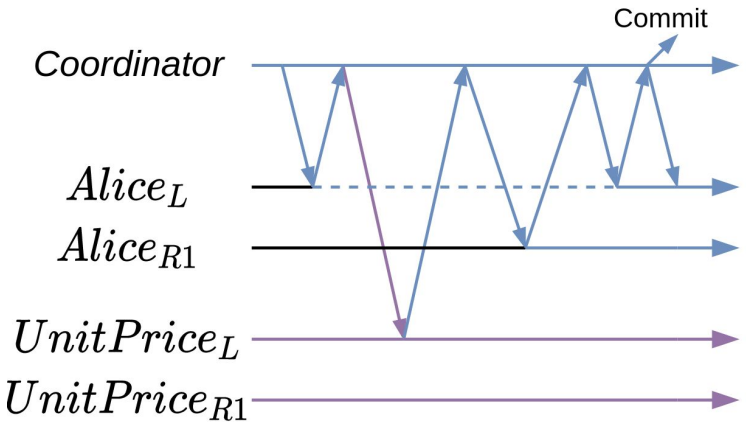
	Shard size	Write (msg)	Read (msg)	Commit (RTT)	Abort (RTT)
2PC	1	3	3	1	1
2PC + Paxos	**2f+1**	3 + **8f**	3 + **8f**	**2**	**2**

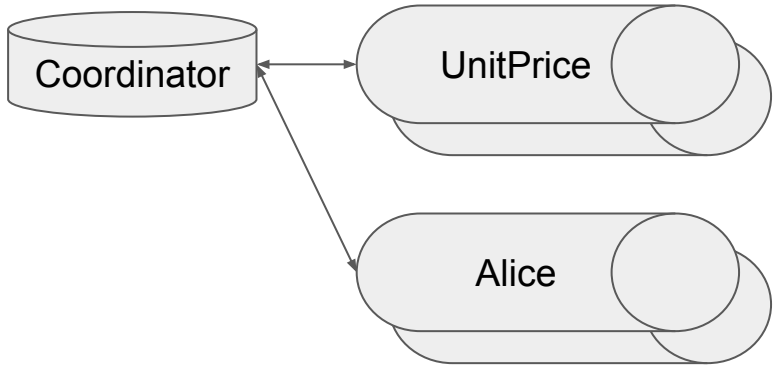




FaRM - Optimised 2PC + Paxos

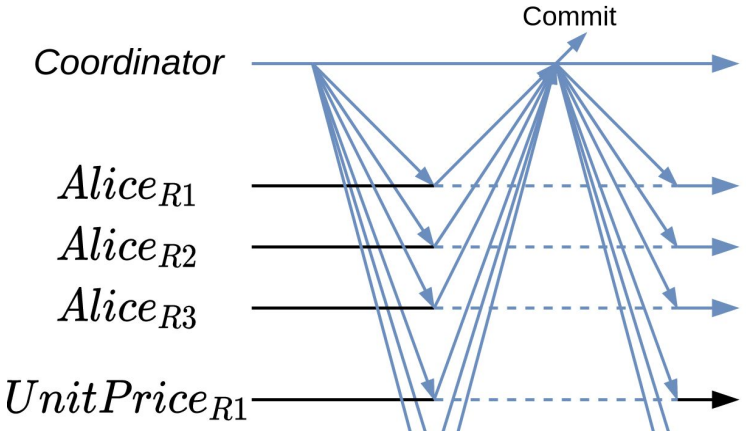
	Shard size	Write (msg)	Read (msg)	Commit (RTT)	Abort (RTT)
2PC	1	3	3	1	1
2PC + Paxos	2f+1	3 + 8f	3 + 8f	2	2
FaRM	**f+1**	5+**3f**	**2**	**3,4**	1,2

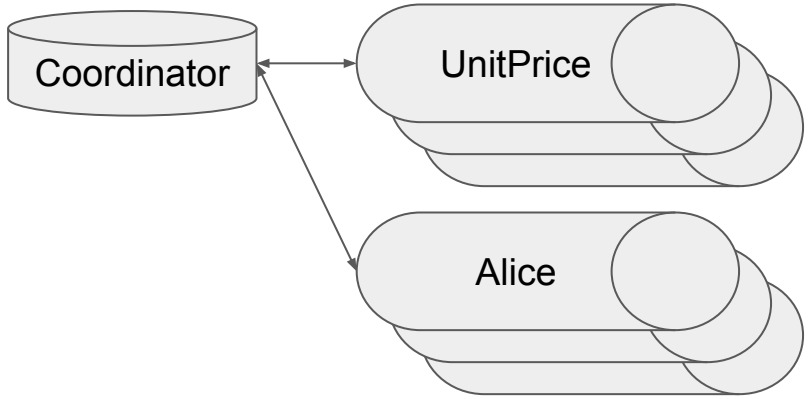




MDCC / Tapir / Meerkat - 2PC + Paxos + FastPaxos

	Shard size	Write (msg)	Read (msg)	Commit (RTT)	Abort (RTT)
2PC	1	3	3	1	1
2PC + Paxos	2f+1	3 + 8f	3 + 8f	2	2
FaRM	f+1	5+3f	2	3,4	1,2
2PC + FastPaxos	2f+1	3+**6f**	**3+6f**	**1,2**	1,2





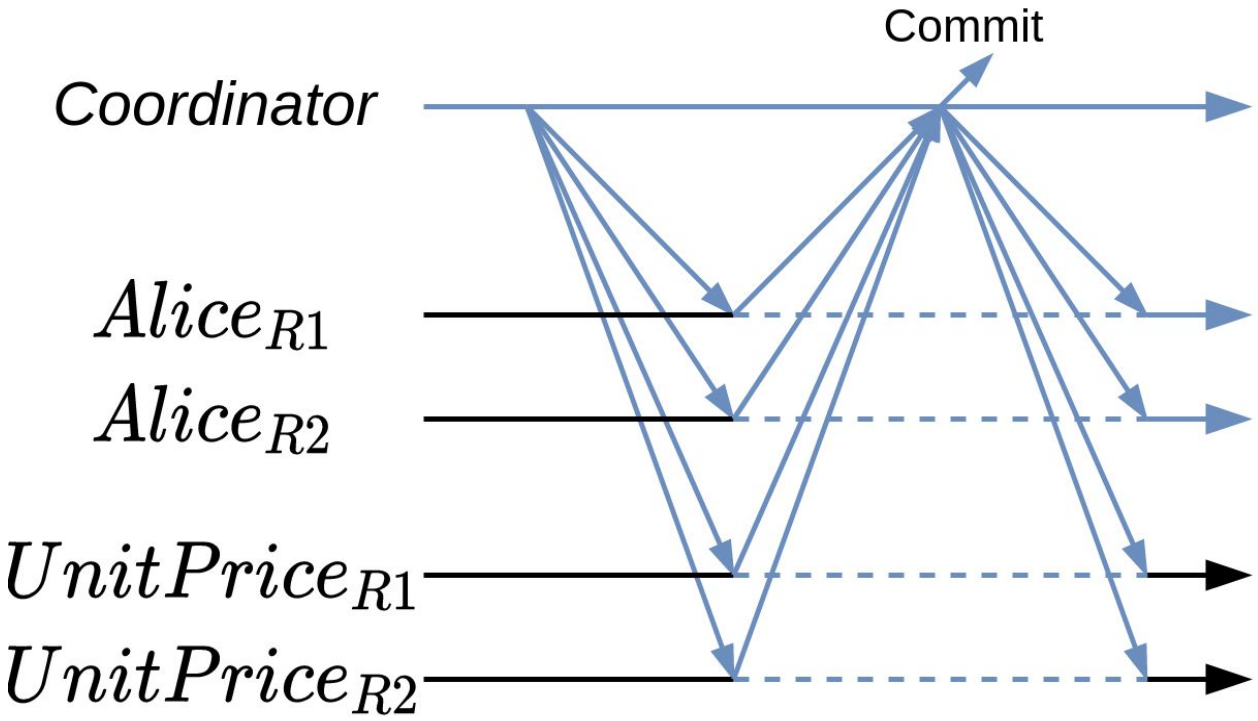
Tradeoff: Low overhead vs low latency

	Shard size	Write (msg)	Read (msg)	Commit (RTT)	Abort (RTT)
FaRM	f+1	5+3f	2	3,4	1,2
2PC + FastPaxos	2f+1	3+6f	3+6f	1,2	1,2

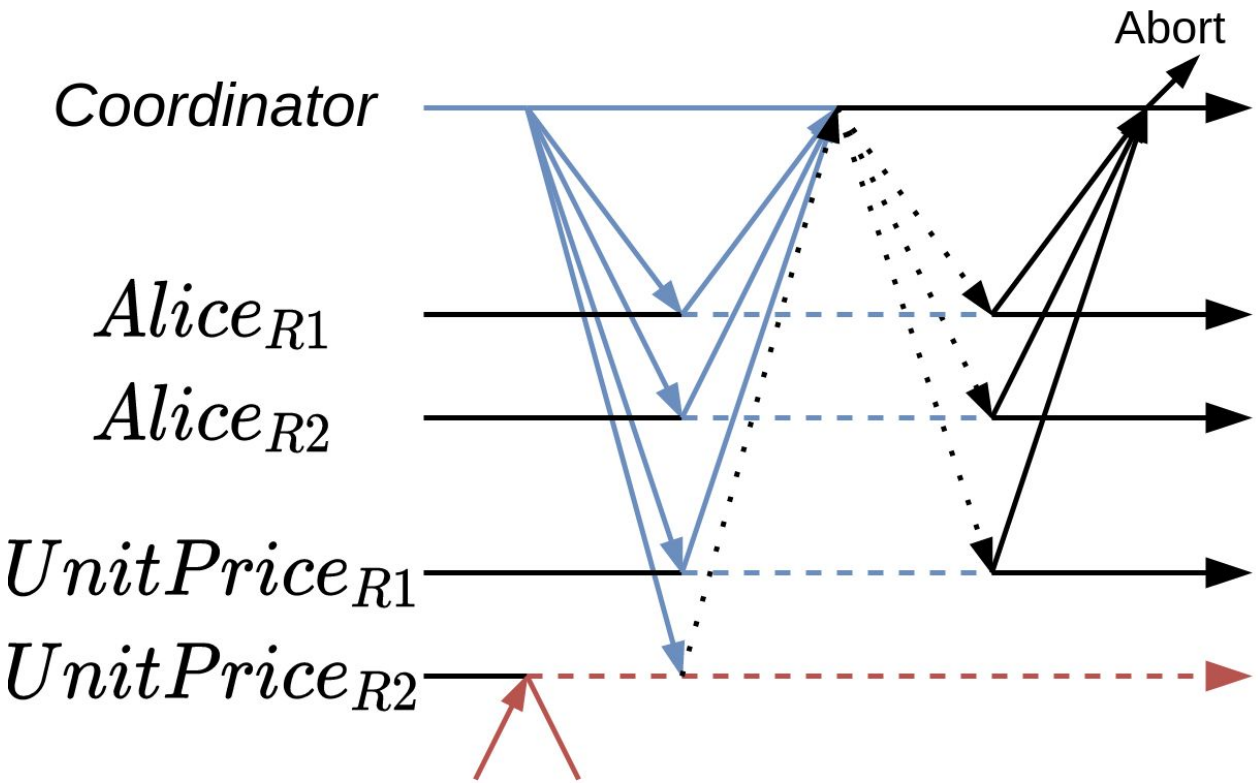
Can we have both?

Yes! - ish

Unanimous 2PC (U2PC) generalises 2PC

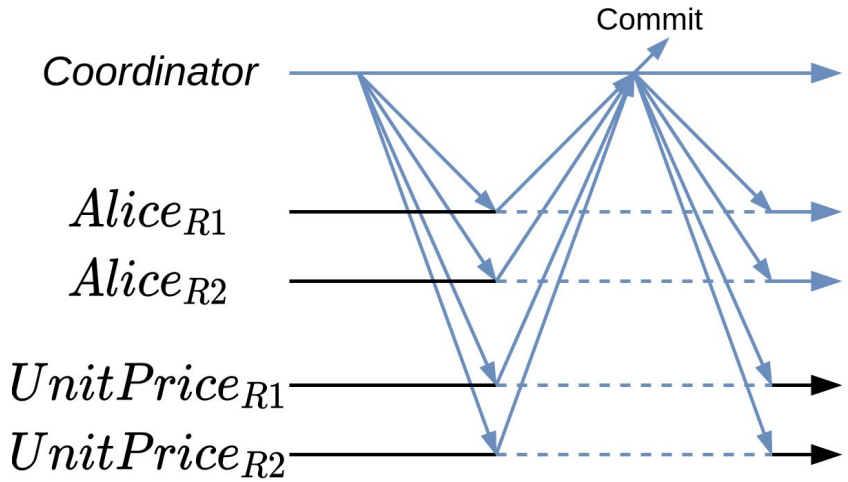


U2PC abort after unlocking one shard



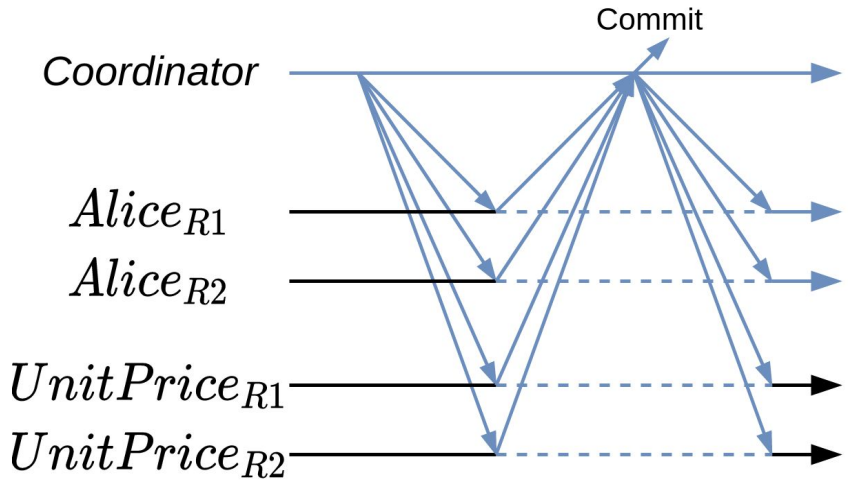
Transaction recovery from replica logs after stop the world

If **any** replica has unlocked	Preserve abort / commit



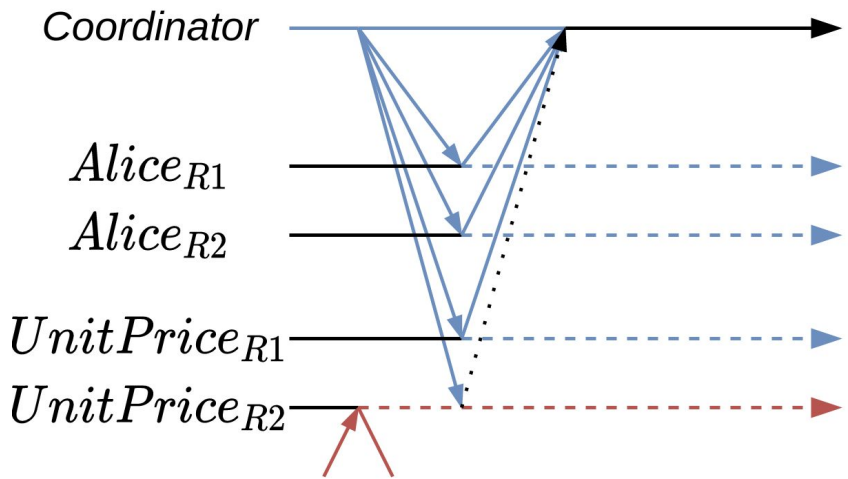
Transaction recovery from replica logs after stop the world

If **any** replica has unlocked	Preserve abort / commit
If **all** replicas are locked	Commit



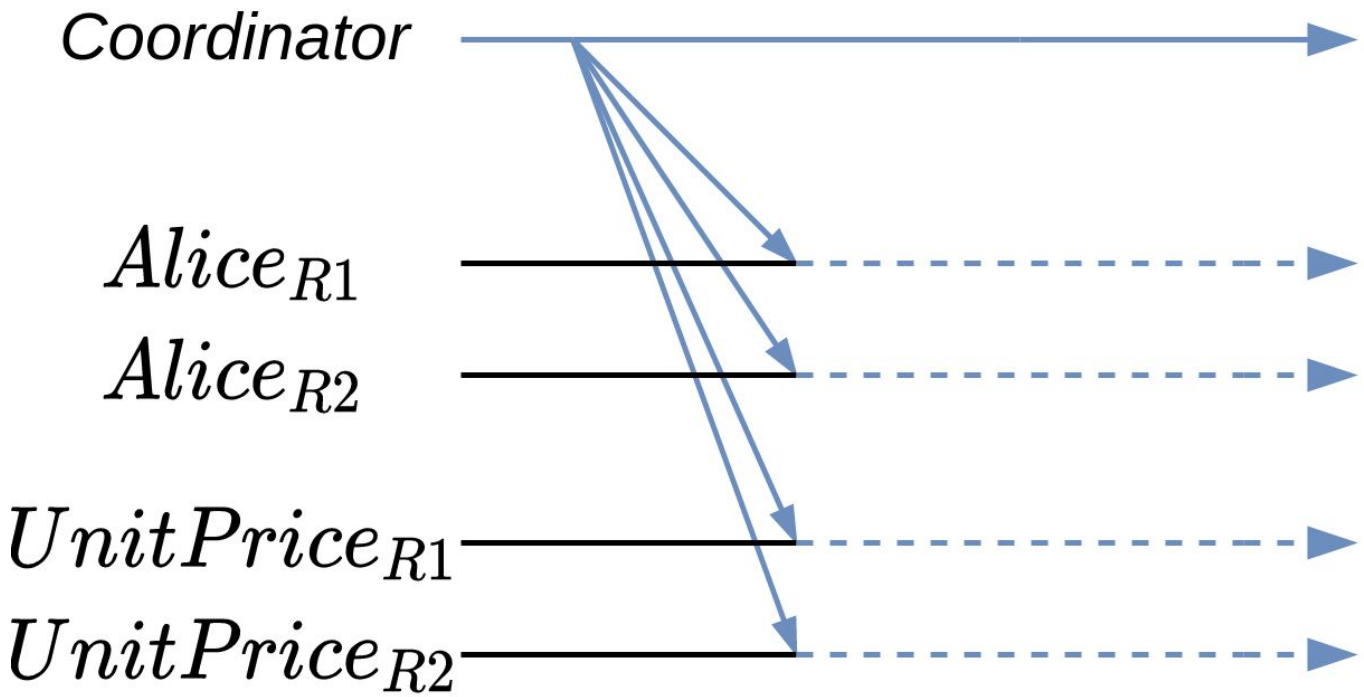
Transaction recovery from replica logs after stop the world

If **any** replica has unlocked	Preserve abort / commit
If **all** replicas are locked	Commit
If **not all** replicas are locked and **none** unlocked	Abort



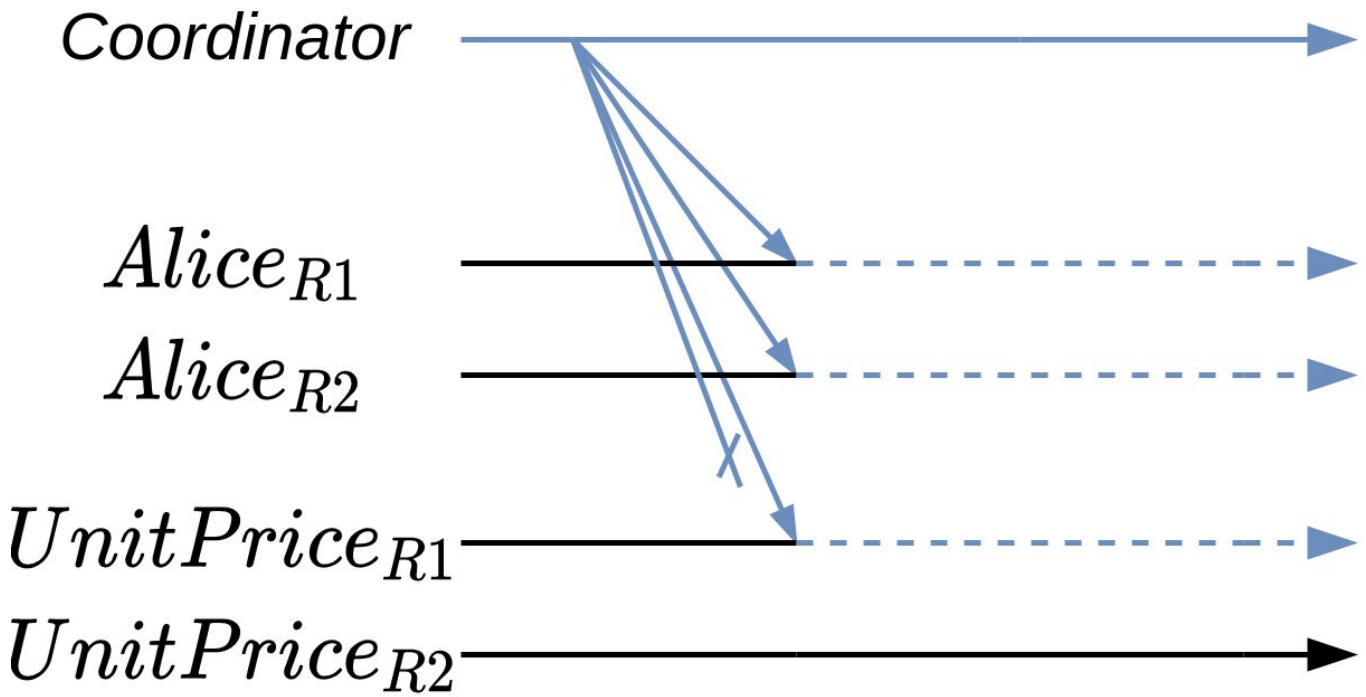
Recovery = Commit

any unlocked	Preserve
all locked	Commit
otherwise	Abort



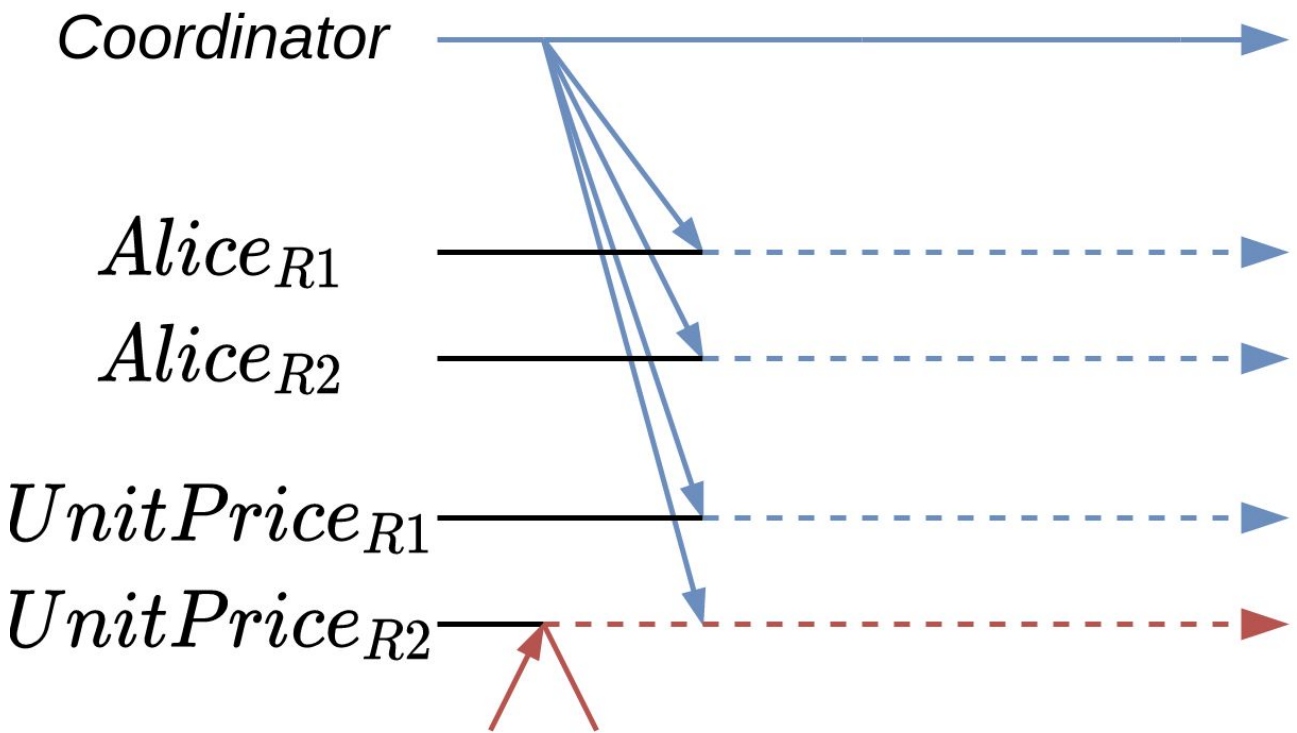
Recovery = Commit or Abort

any unlocked	Preserve
all locked	Commit
otherwise	Abort



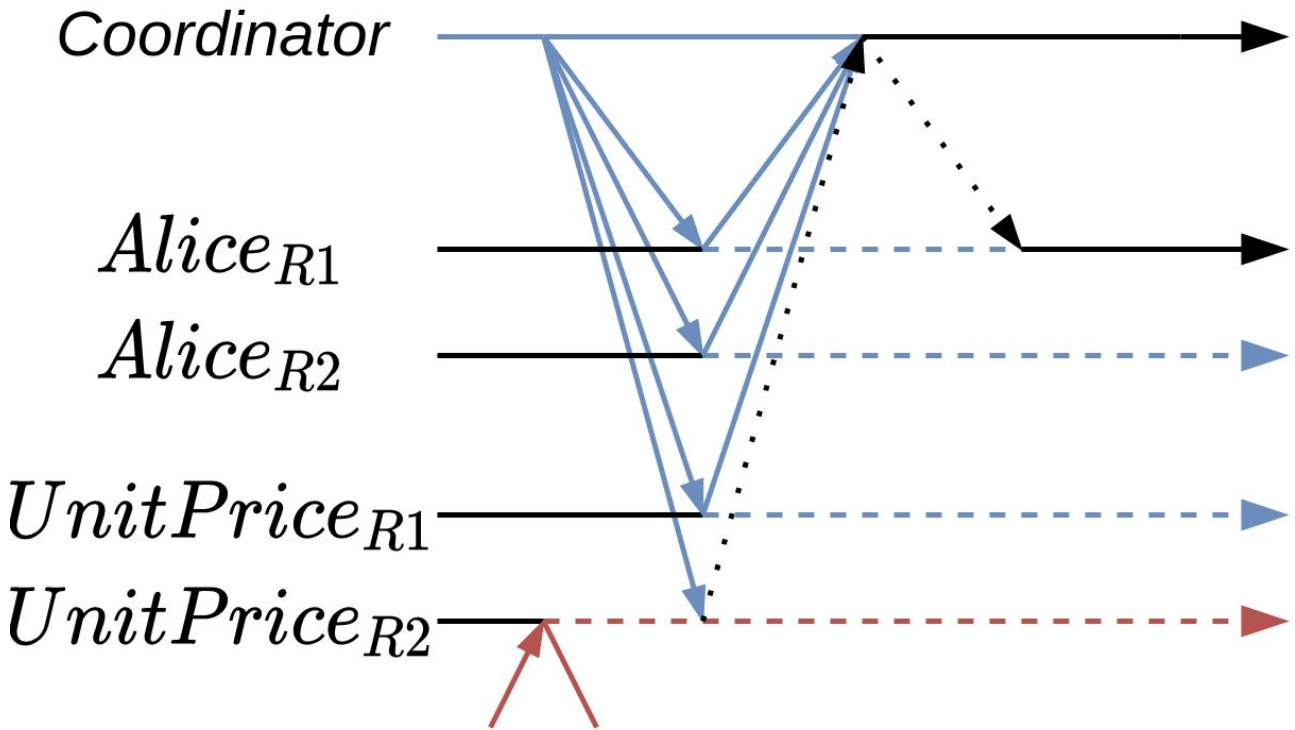
Recovery = Commit or Abort

any unlocked	Preserve
all locked	Commit
otherwise	Abort



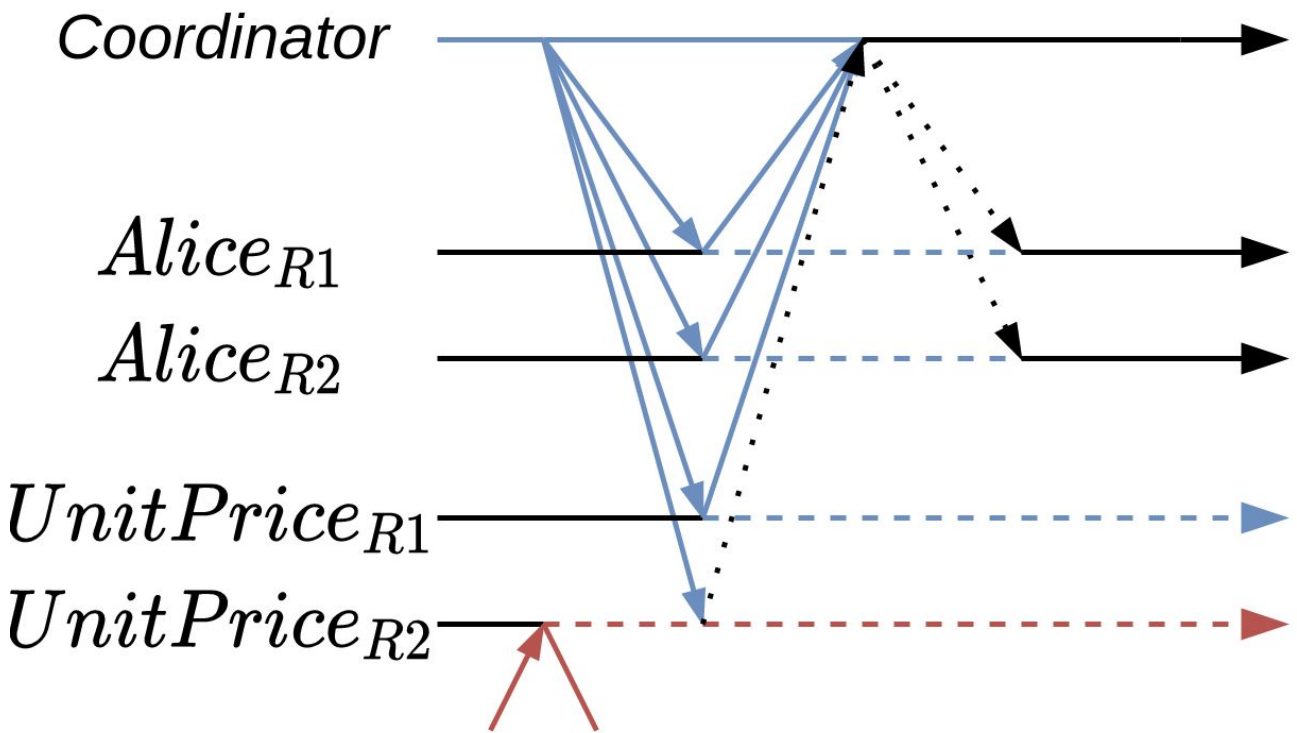
Recovery = Commit or Abort

any unlocked	Preserve
all locked	Commit
otherwise	Abort



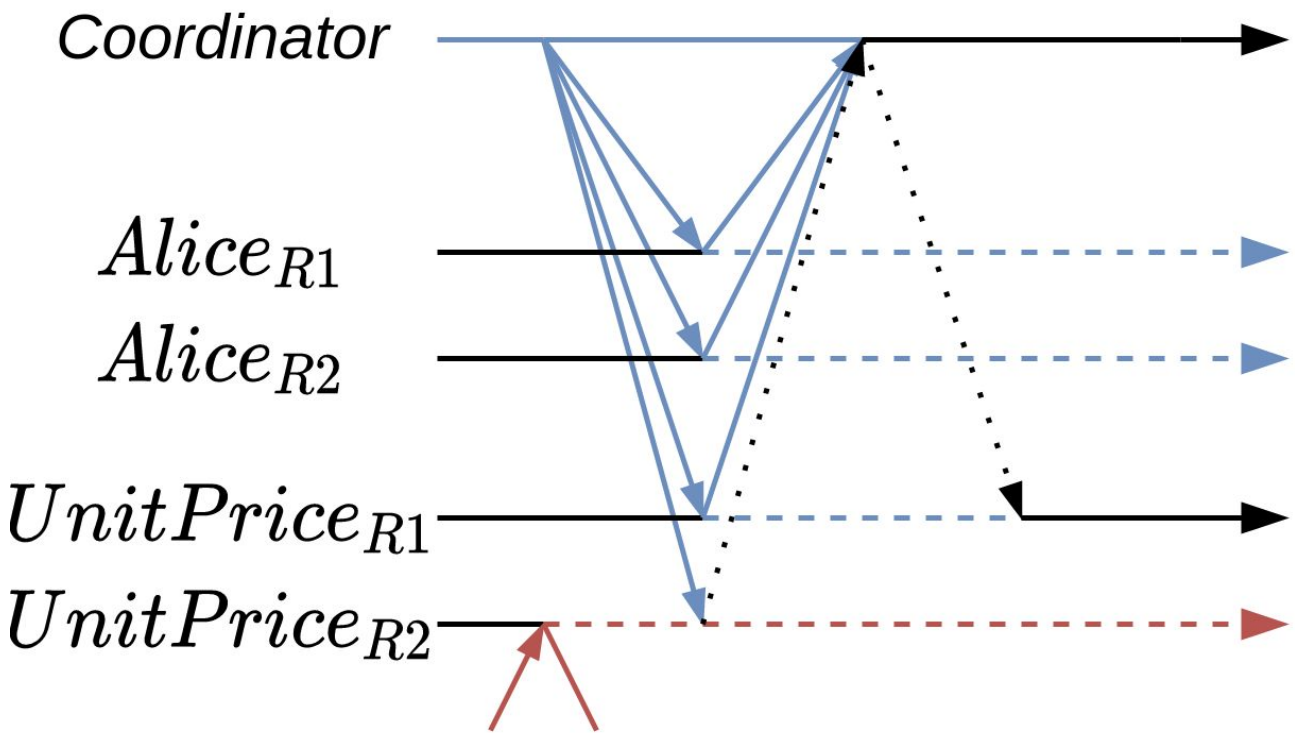
Recovery = Abort

any unlocked	Preserve
all locked	Commit
otherwise	Abort

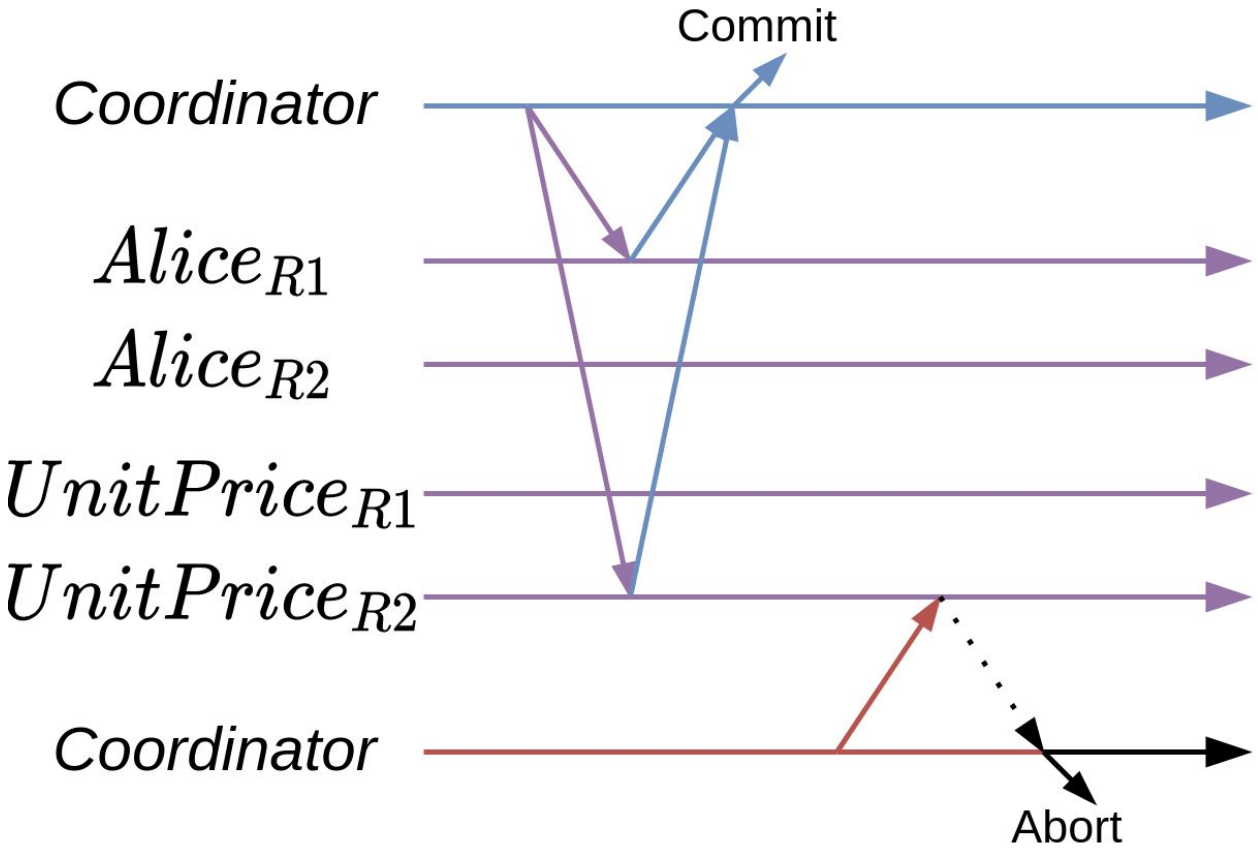


Recovery = Abort

any unlocked	Preserve
all locked	Commit
otherwise	Abort

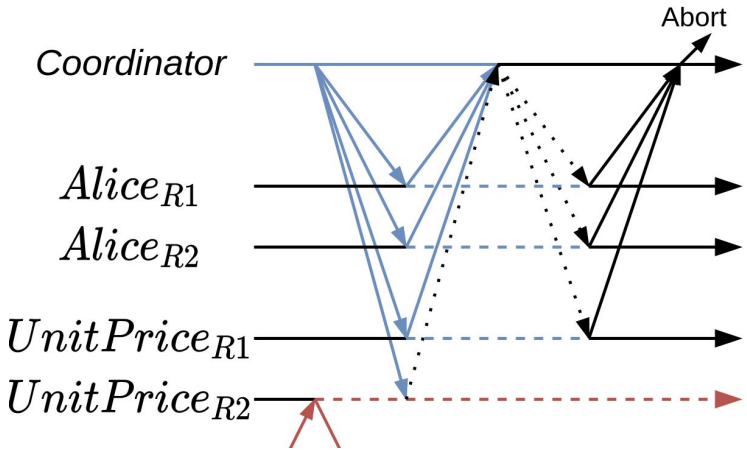


U2PC read-only transactions



Low overhead and low latency

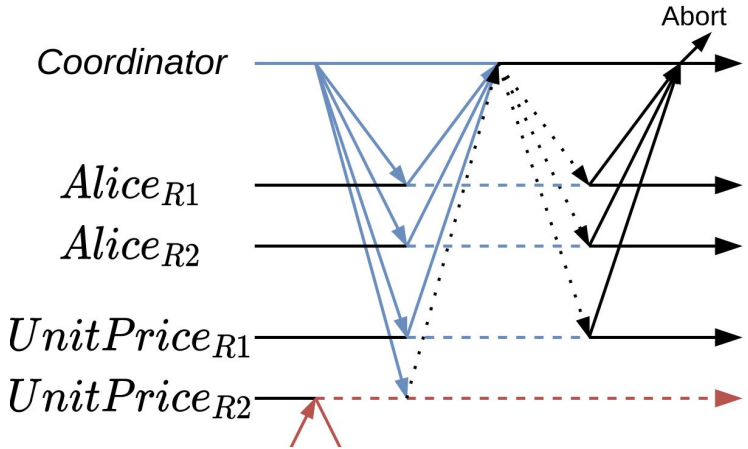
	Shard size	Write (msg)	Read (msg)	Commit (RTT)	Abort (RTT)
FaRM	f+1	5+3f	2	3,4	1,2
2PC + FastPaxos	2f+1	3+6f	3+6f	1,2	1,2
U2PC	f+1	3+3f	2, 3+3f	1	1,2



Intermission - Any questions?

Next: optimisations (?)

	Shard size	Write (msg)	Read (msg)	Commit (RTT)	Abort (RTT)
FaRM	f+1	5+3f	2	3,4	1,2
2PC + FastPaxos	2f+1	3+6f	3+6f	1,2	1,2
U2PC	f+1	3+3f	2, 3+3f	1	1,2



chris.jensen@cl.cam.ac.uk

github.com/Cjen1/u2pc-tla/

Also chat to me about post PhD work

U2PC as a drop in 2PC replacement

- At commit coordinator must know:
  - Read and modified shards
  - Written values to shards
- Then replicate each shard F+1 times

Optimisation
Run FaRM and U2PC in the same cluster
Choose protocol per transaction

- FaRM leader locks override U2PC locks
  - Safe since U2PC requires the leader lock to progress

- Tradeoff space is non-trivial
  - ReadWrite: U2PC << FaRM
  - Read-Only: U2PC = FaRM
  - Write-Only: U2PC > FaRM
  - Contention: U2PC >> FaRM

	Write (msg)	Read (msg)
FaRM	5+3f	2
U2PC	3+3f	2, 3+3f

Optimisation
Better/Different locks to reduce aborts

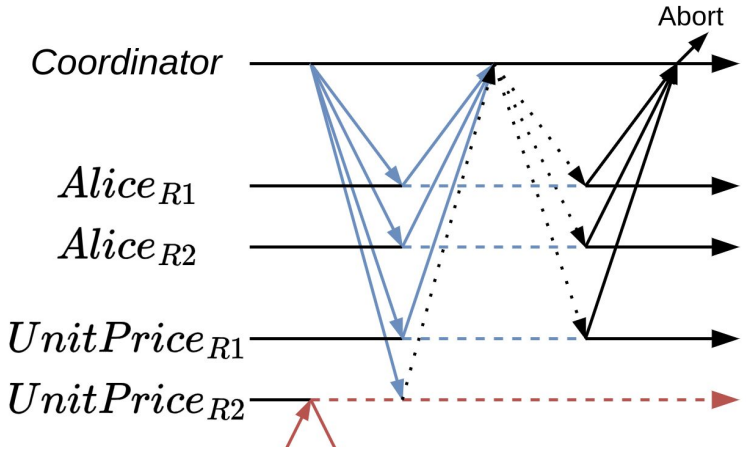
- Fancier locks
  - MRSW
  - Lower consistency level
- Domain specific:
  - Allocation based schemes: Current - inflight > 0

Optimisation?
DrTM: lock during execution for free commit phase

- When `txn.Read` or `txn.Write`, acquire lock
- On last `txn.Read` or `txn.Write`, log as finalised
- Recovery commit if **all** locked and **any** finalised
- Higher contention due to longer lock period

Any questions?

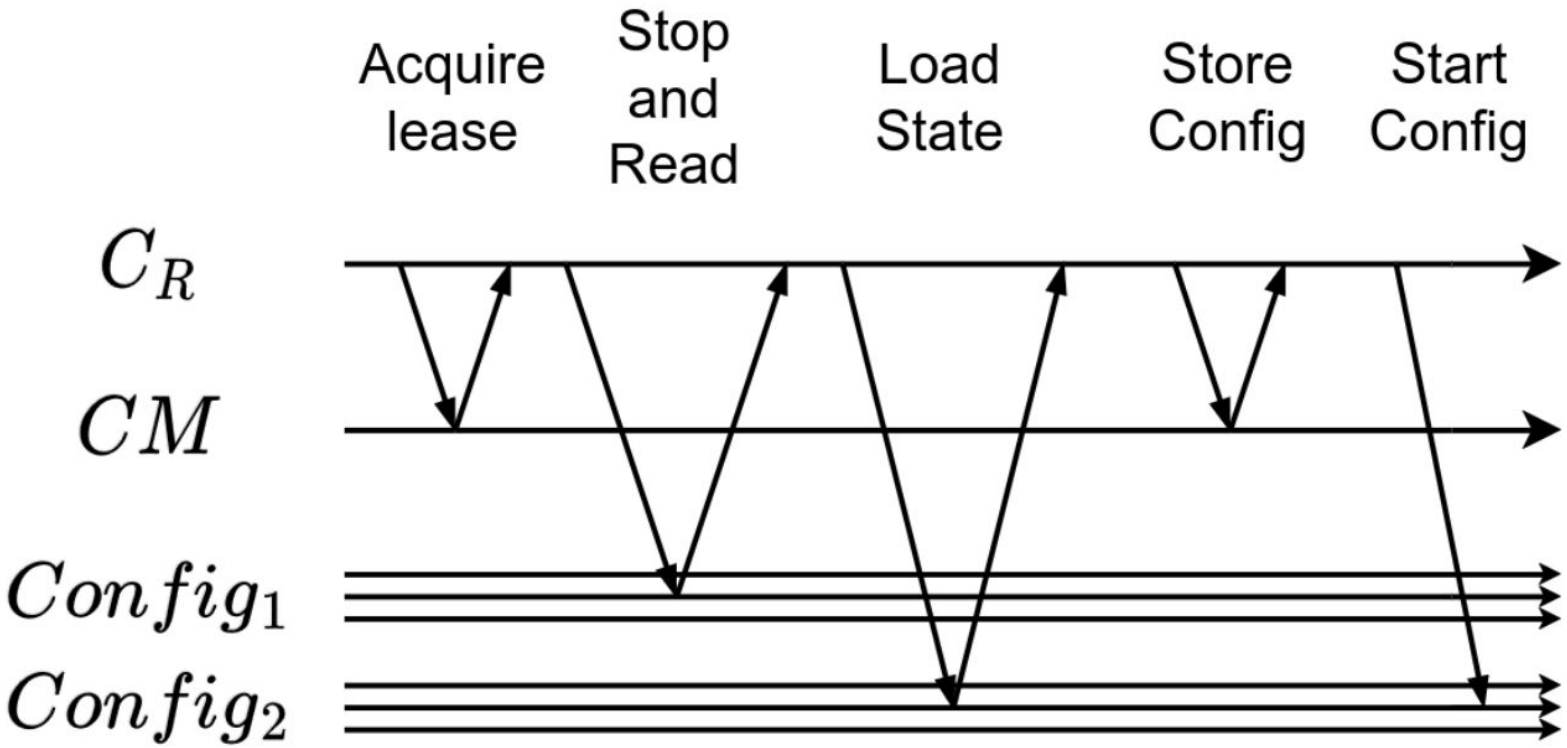
	Shard size	Write (msg)	Read (msg)	Commit (RTT)	Abort (RTT)
FaRM	f+1	5+3f	2	3,4	1,2
2PC + FastPaxos	2f+1	3+6f	3+6f	1,2	1,2
U2PC	f+1	3+3f	2, 3+3f	1	1,2



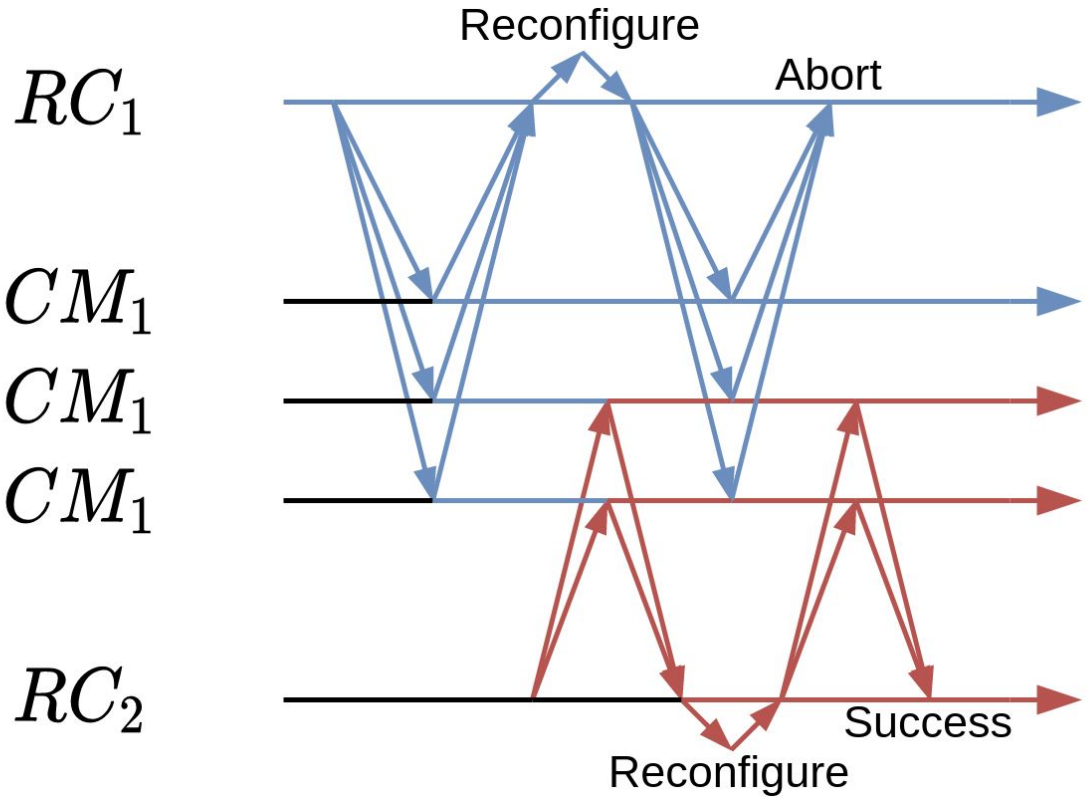
chris.jensen@cl.cam.ac.uk
github.com/Cjen1/u2pc-tla/

Also chat to me about post PhD work

Recovery protocol



Configuration Manager - Virtually synchronous leases



Term = (epoch, uuid)