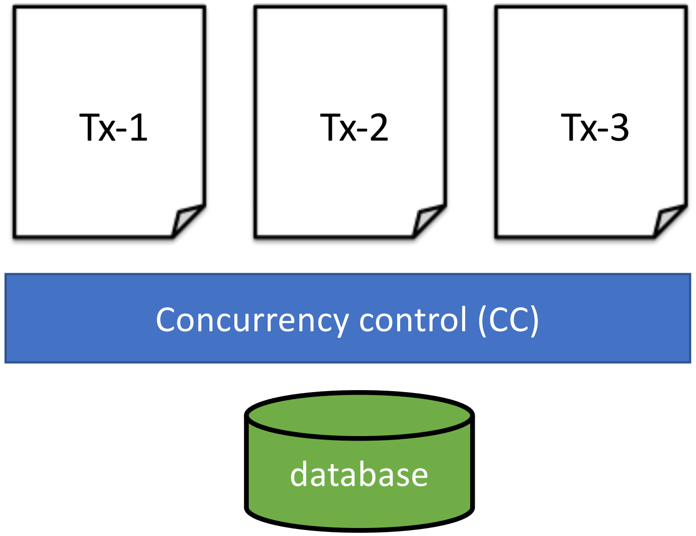
POLYJUICE: High-Performance Transactions via Learned Concurrency Control

Jiachen Wang, **DING DING**, Huan Wang, Conrad Christensen,
Zhaoguo Wang, Haibo Chen and Jinyang Li

dding@nyu.edu

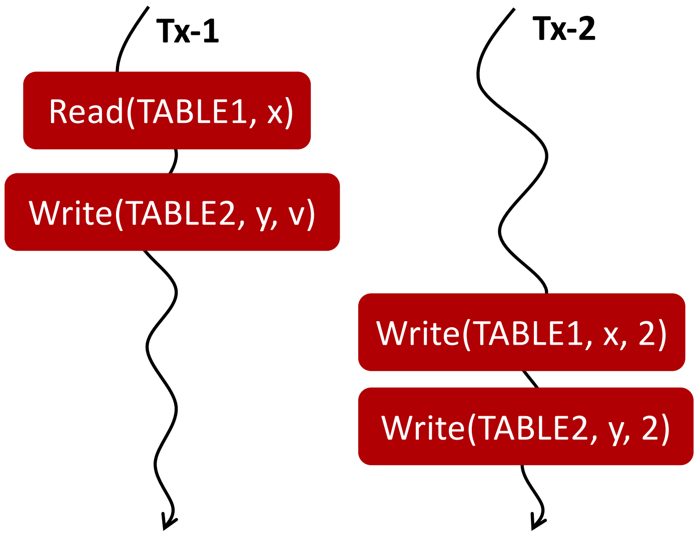
Concurrency control ensures tx serializability



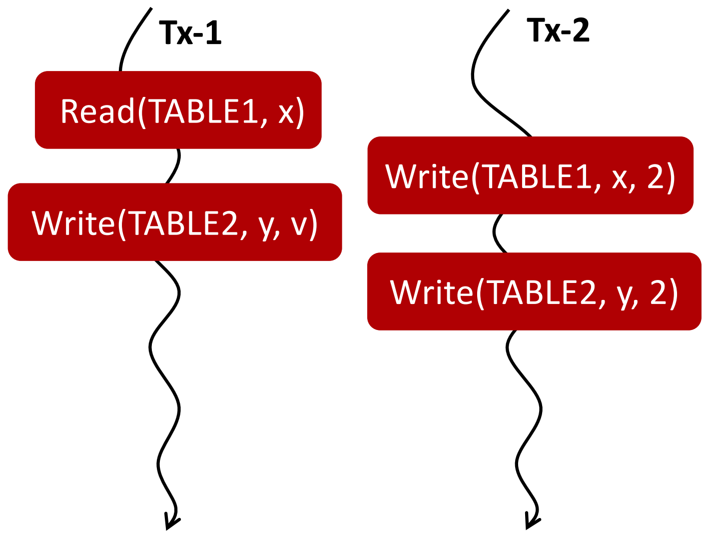
- Transactions provides the abstraction of ACID.
- Concurrency control (CC) ensures Isolation (serializability).

CC is crucial to database performance

- CC acts like a scheduler by controling how concurrent executions interleave.
- Maximize interleaving --> better performance.



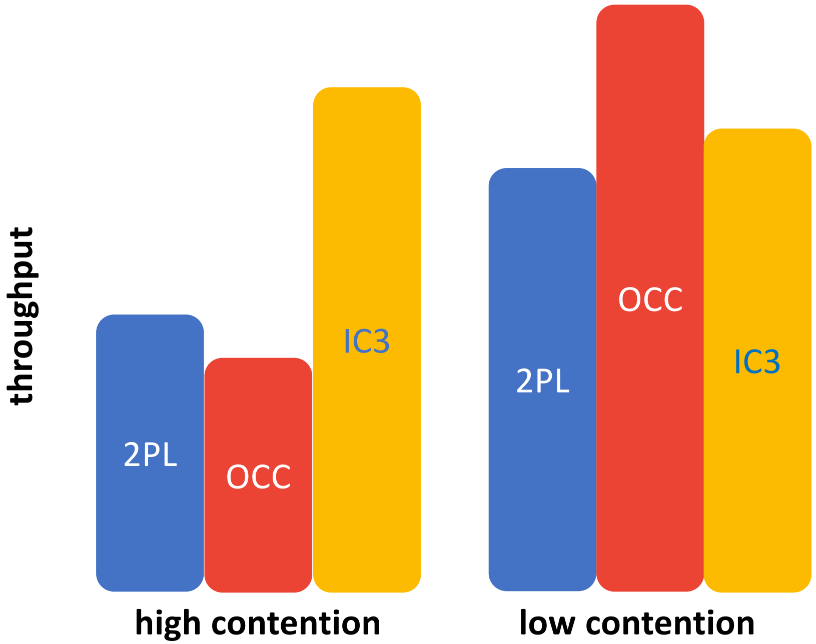
One interleaving



Another more efficient interleaving

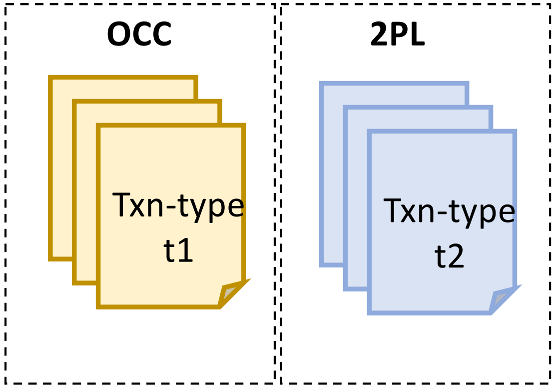
No one CC algorithm fits all

- Some CC performs better than others for a specific workload.

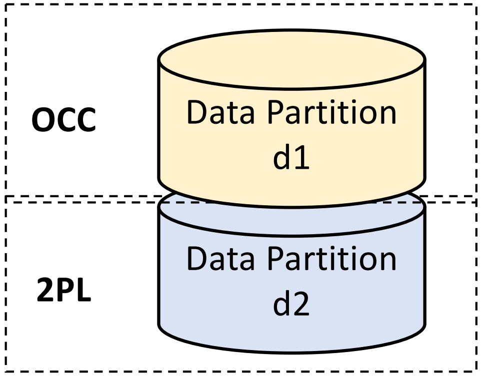


Federated CC?

- A coarse-grained approach to combine a few known CC algorithms.
- Weaknesses ☹
  - Cumbersome: requires manually partitioning of the workload.
  - Suboptimal: uses a single CC within each partition.



Tebaldi [SIGMOD'17]



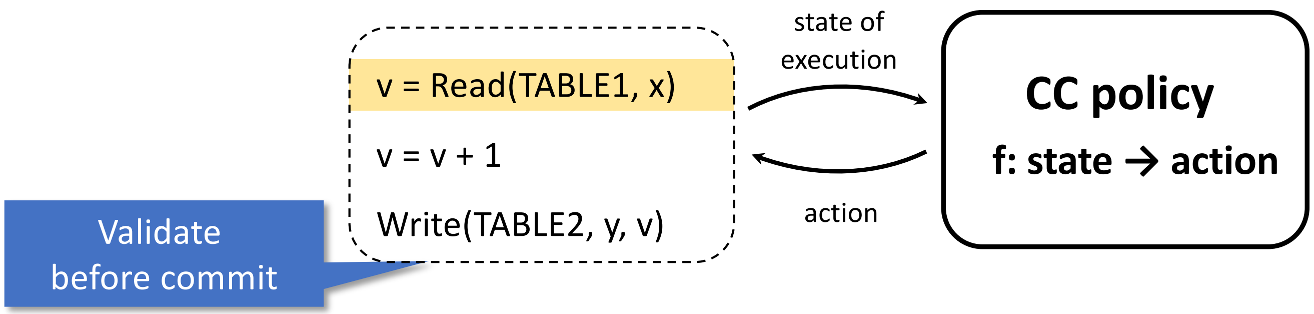
CormCC [ATC' 18]

Our approach:
CC as a fine-grained learning task

- Model CC as a policy function, inspired by reinforcement learning.

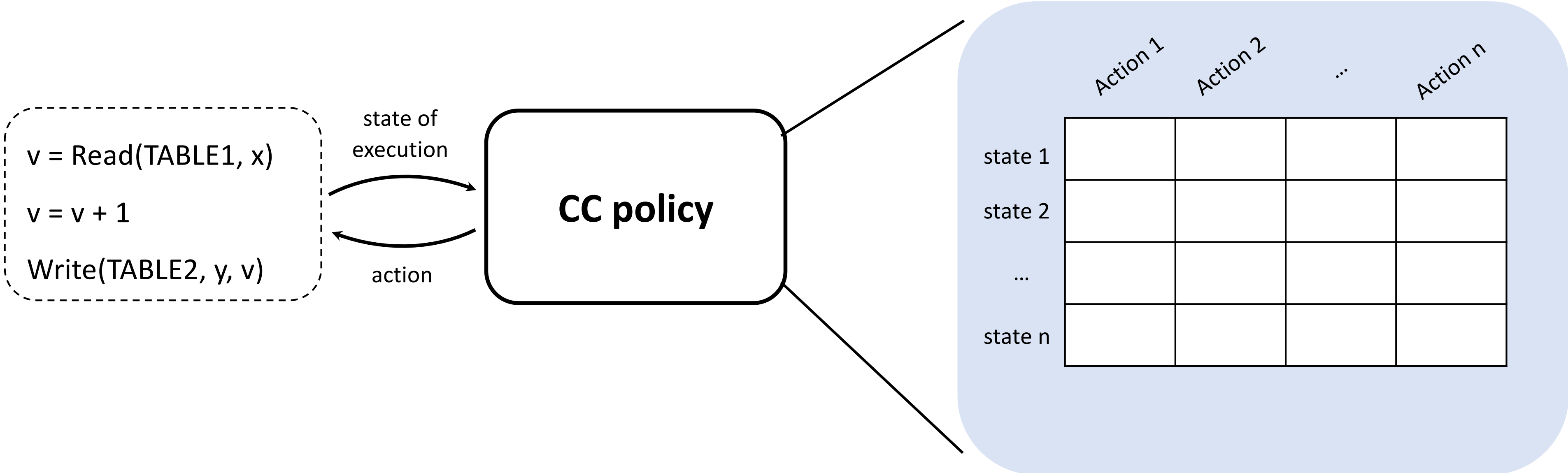
  f: state → action

- Ensure correctness separately by validation.

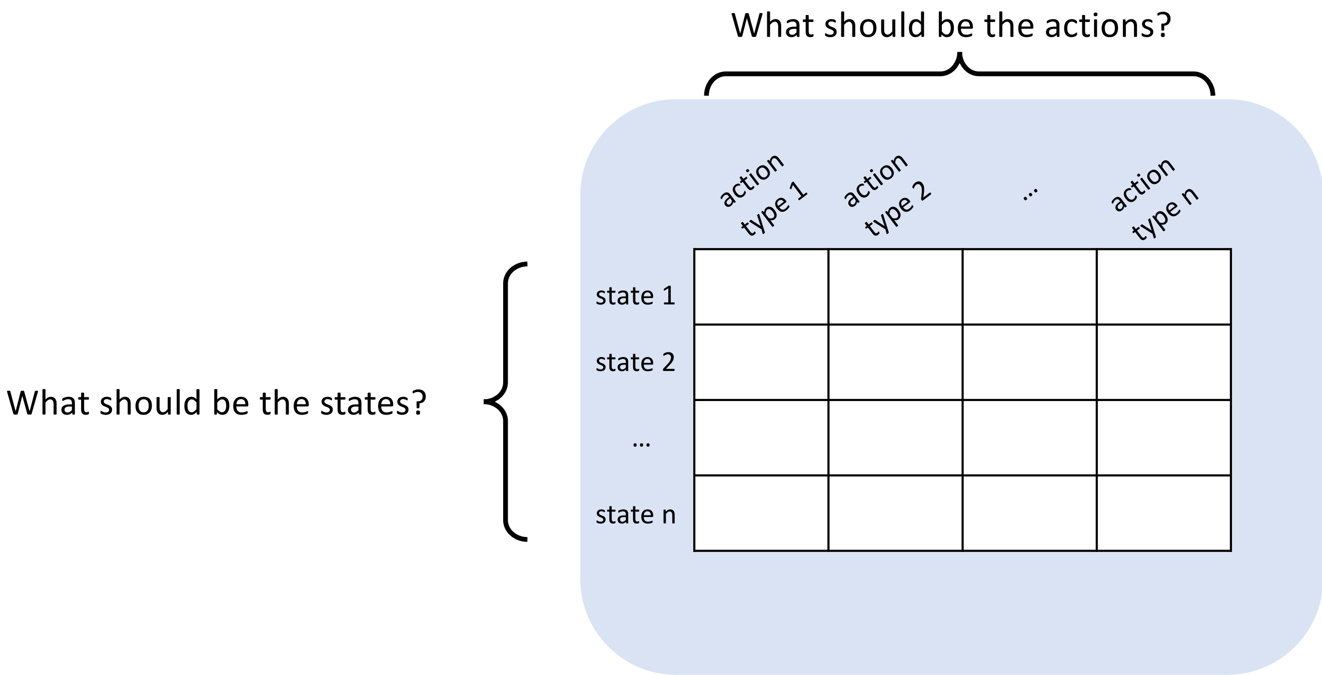


Polyjuice: CC as a policy table

- Represent a CC policy as **a table.**



Challenge: designing the policy table



Goal: design should be able to encode existing CC algorithms.

Polyjuice: state space

Goal

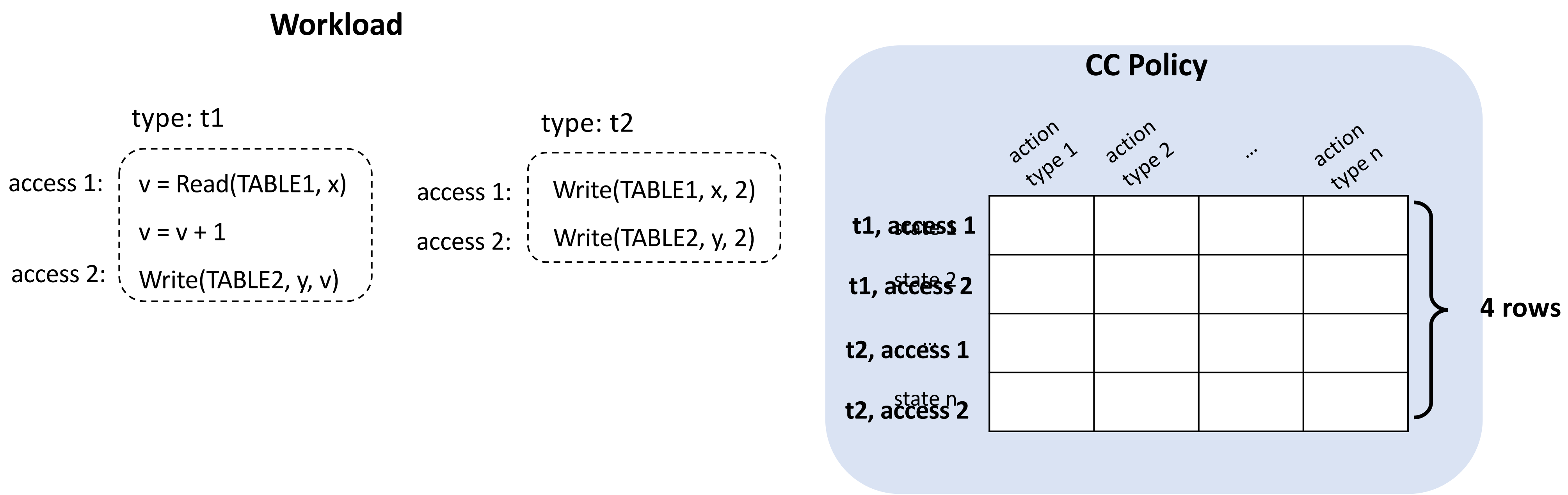
Differentiate state that require different CC actions.

Solution

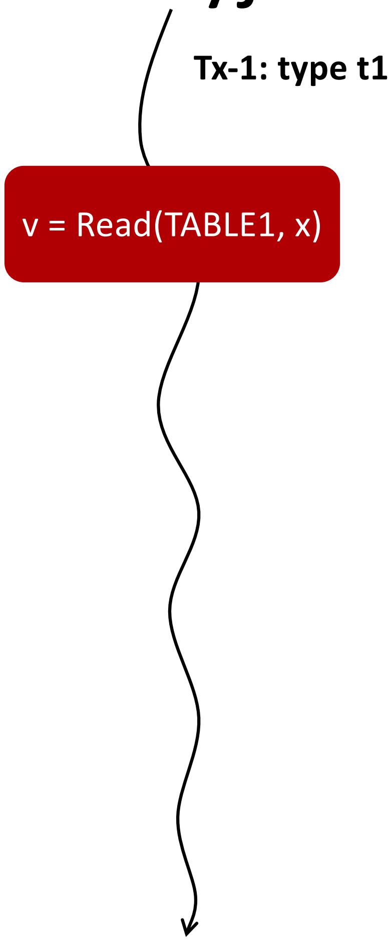
State consists of:

1. The type of transaction being executed.
2. The data access of the transaction being executed.

Polyjuice: state space



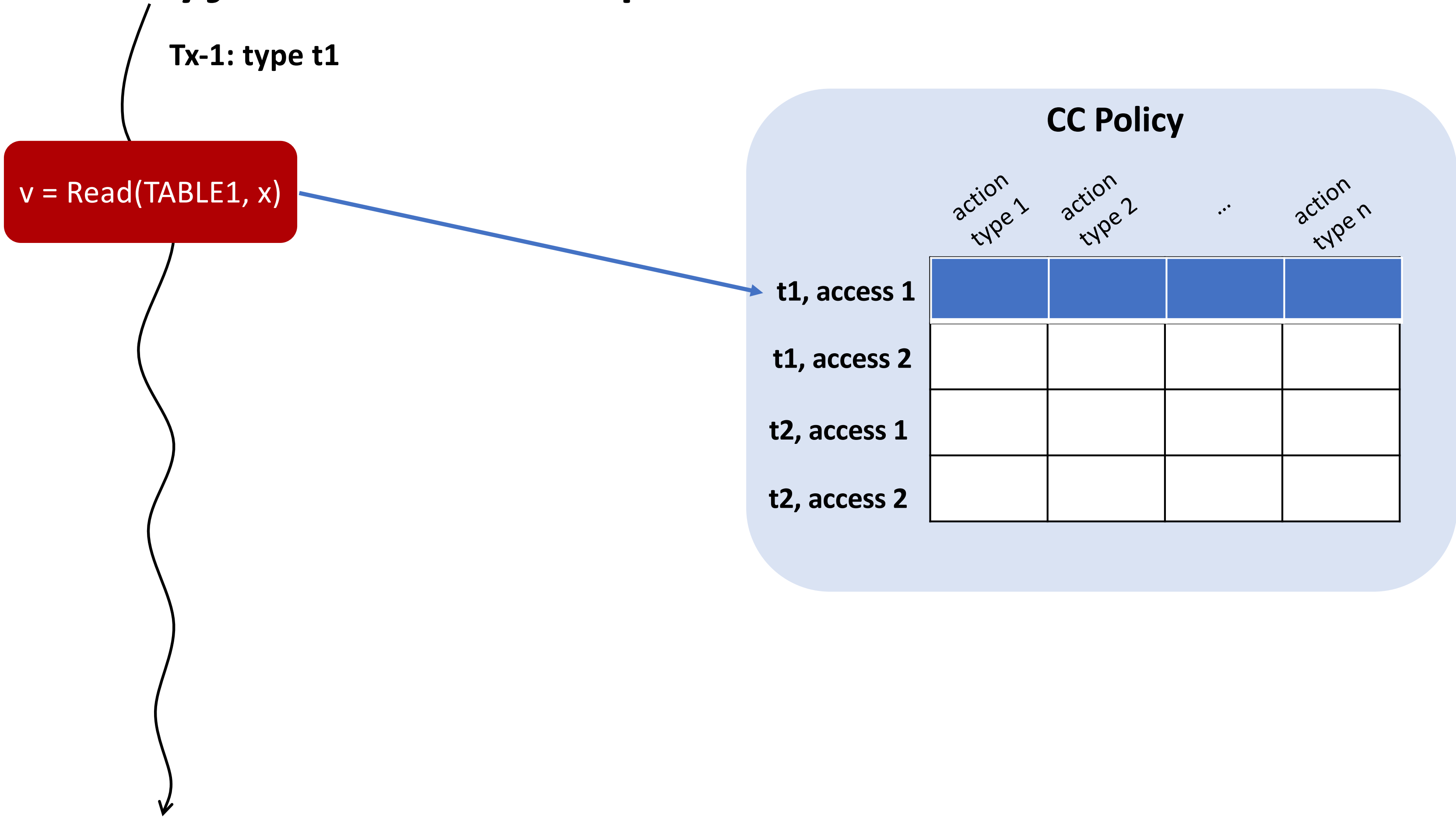
Polyjuice: state space



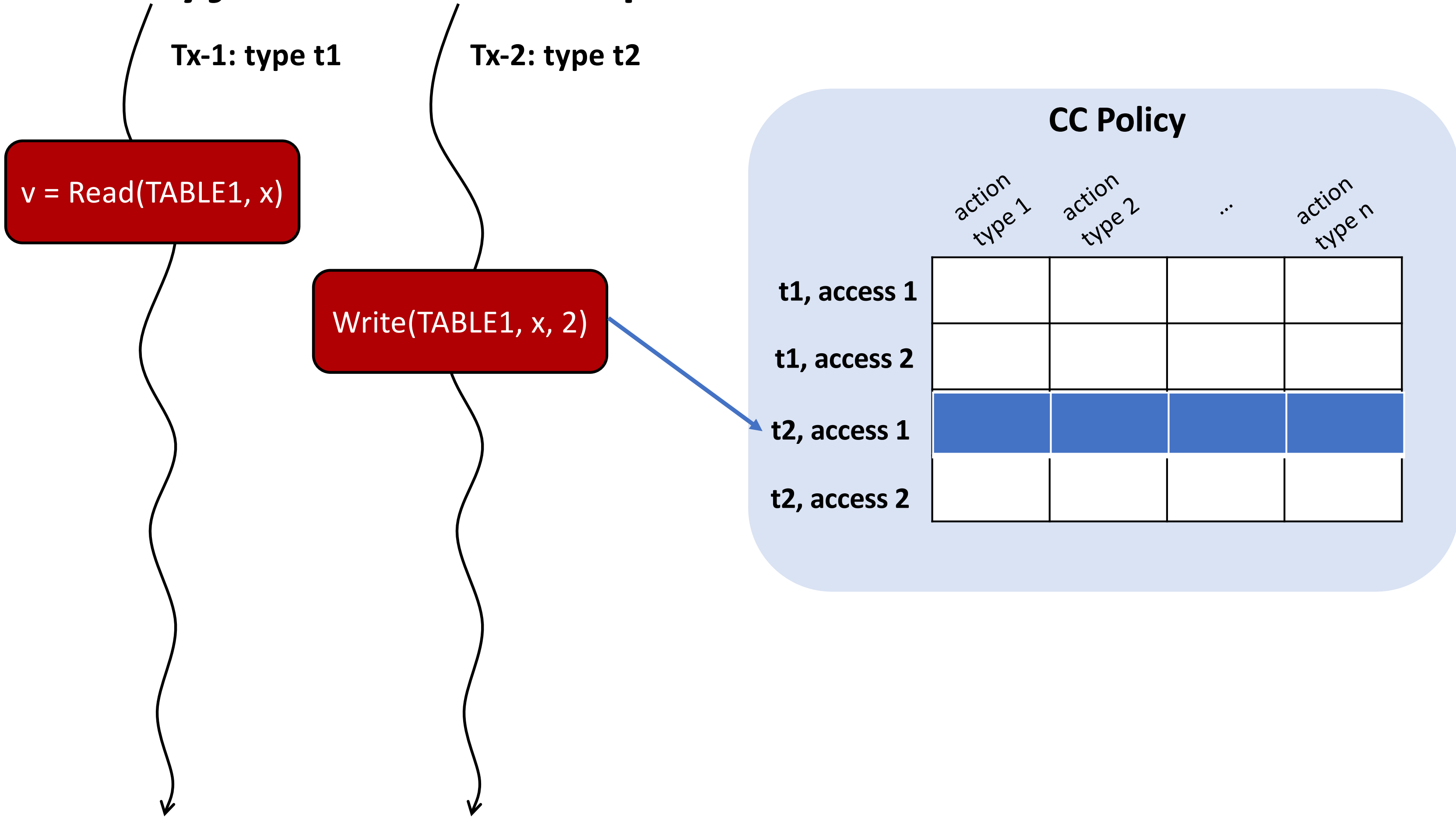
CC Policy

	action type 1	action type 2	...	action type n
t1, access 1				
t1, access 2				
t2, access 1				
t2, access 2				

Polyjuice: state space



Polyjuice: state space

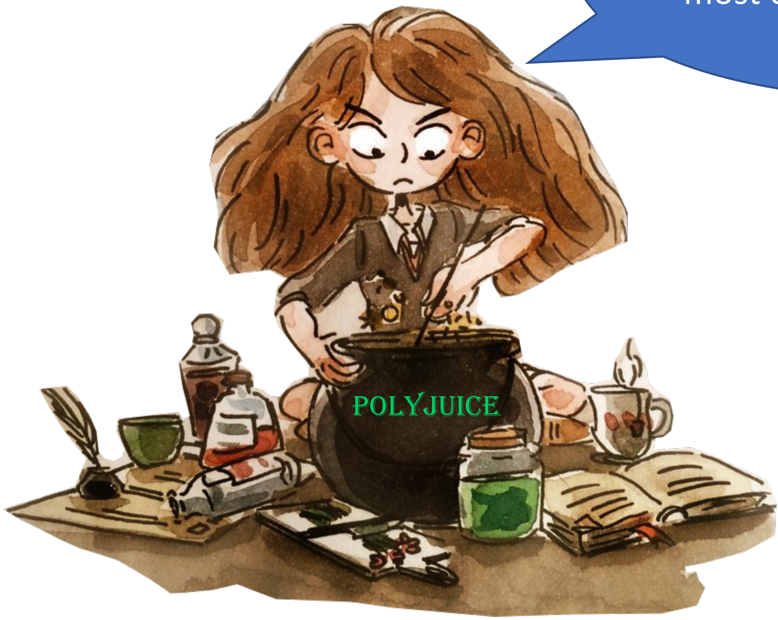


Polyjuice: action space

Goal

Exert control
on the interleaving.

Action space should be able to encode most existing CC algorithms.



Polyjuice: action space

Goal

Exert control
on the interleaving.

Solution

Expose these knobs of control:

- Whether to wait, and how long?
- Which versions of data to read?
- Whether to make a dirty write visible?
- Whether to validate now, prior to commit?

Wait action choices

- No wait.

Used by OCC.

- Wait until dependent transactions commit.

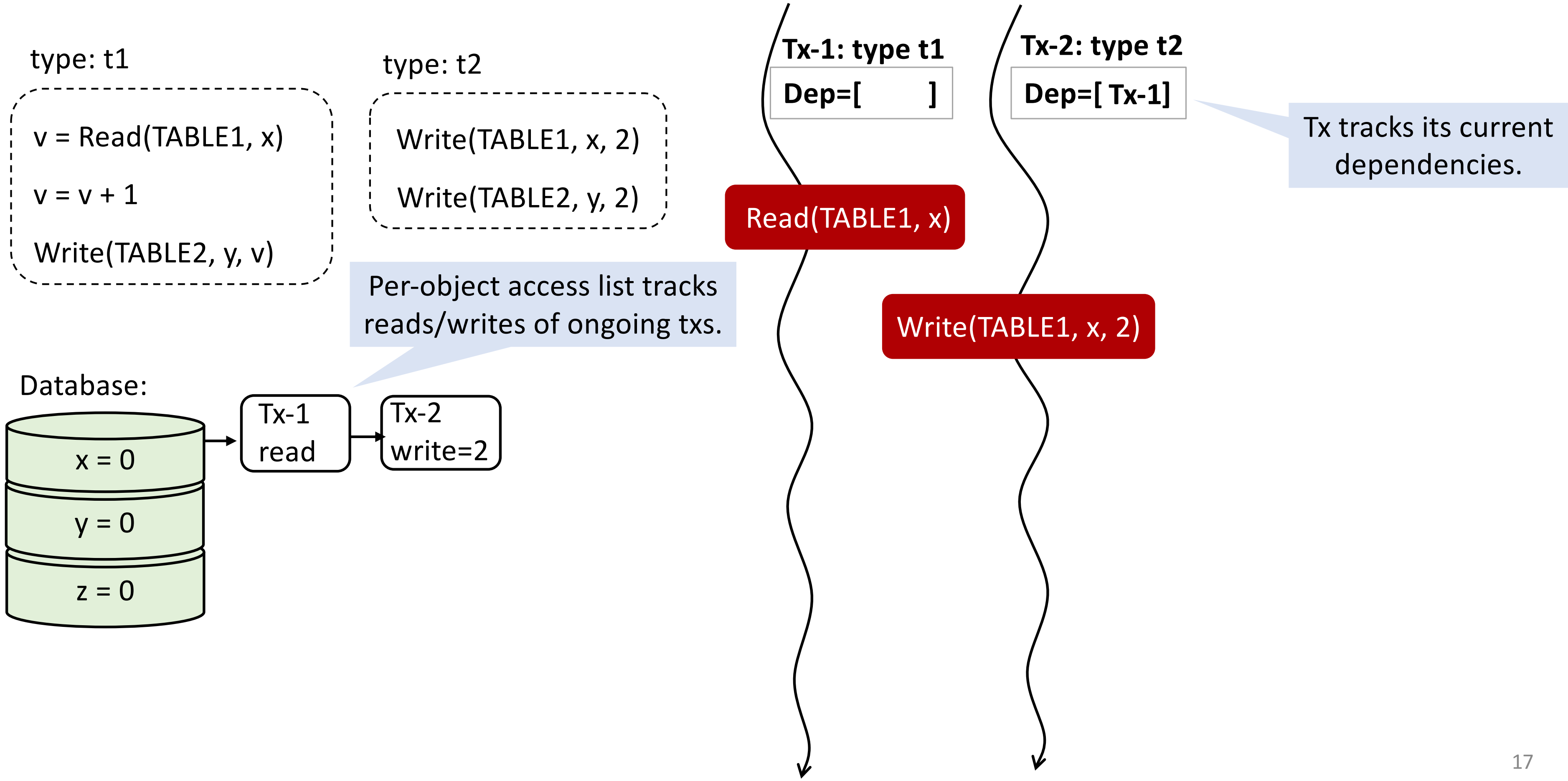
Used by 2PL.

- Wait until dependent transactions finish execution up to some point.

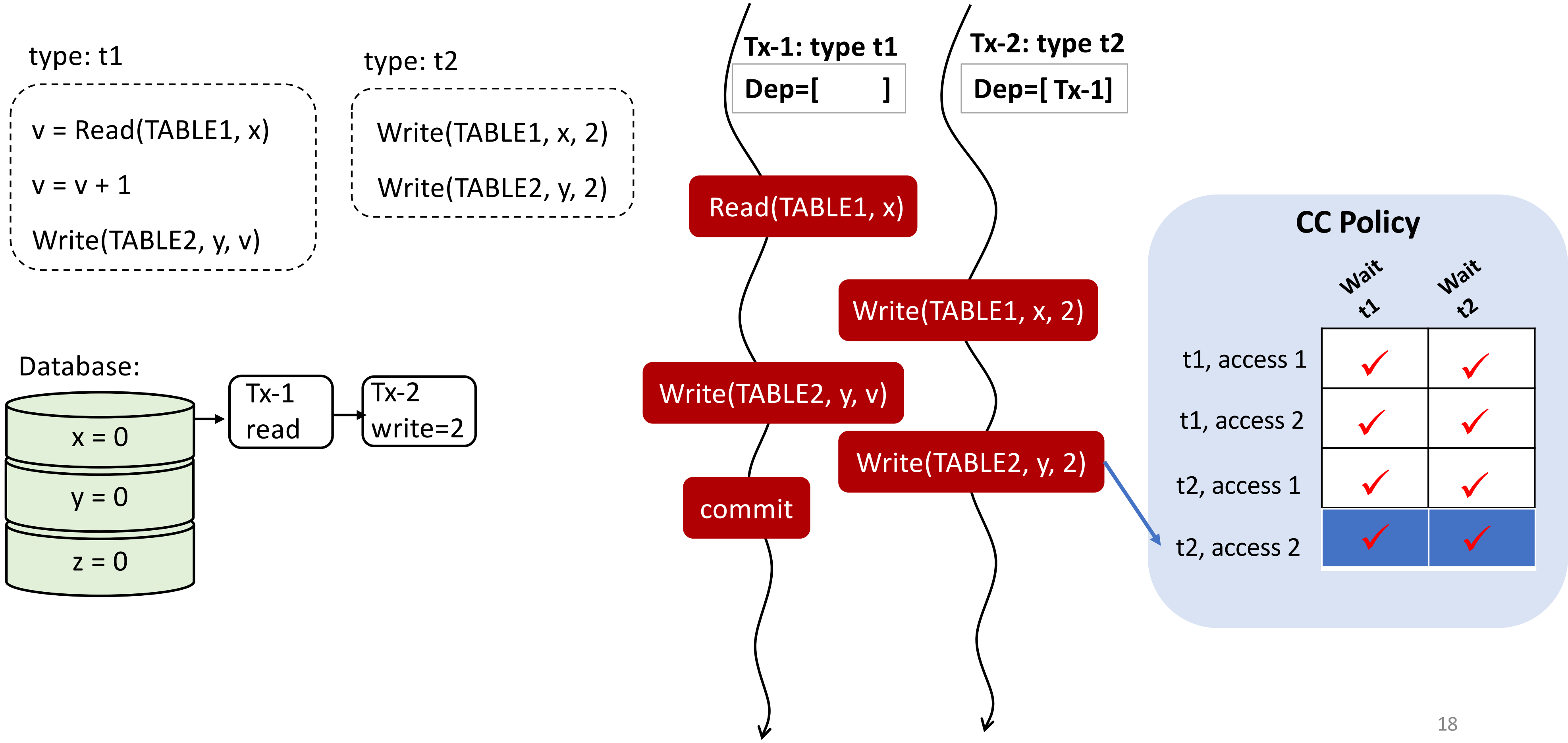
Used by IC3[SIGMOD'16], Callas RP[SOSP'15], DRP[Eurosys'19].

How to realize different wait choices in one implementation?

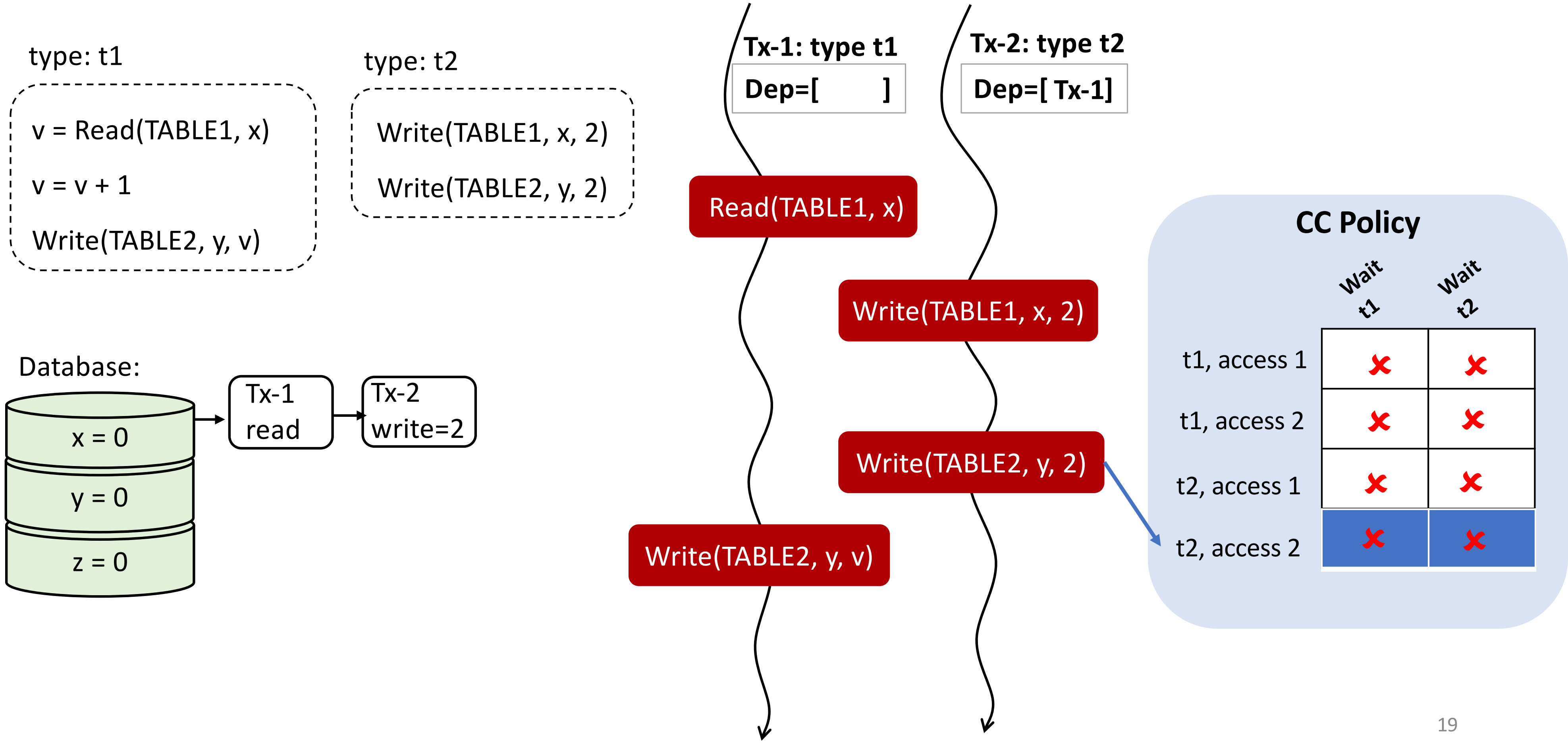
Wait choices: wait commit



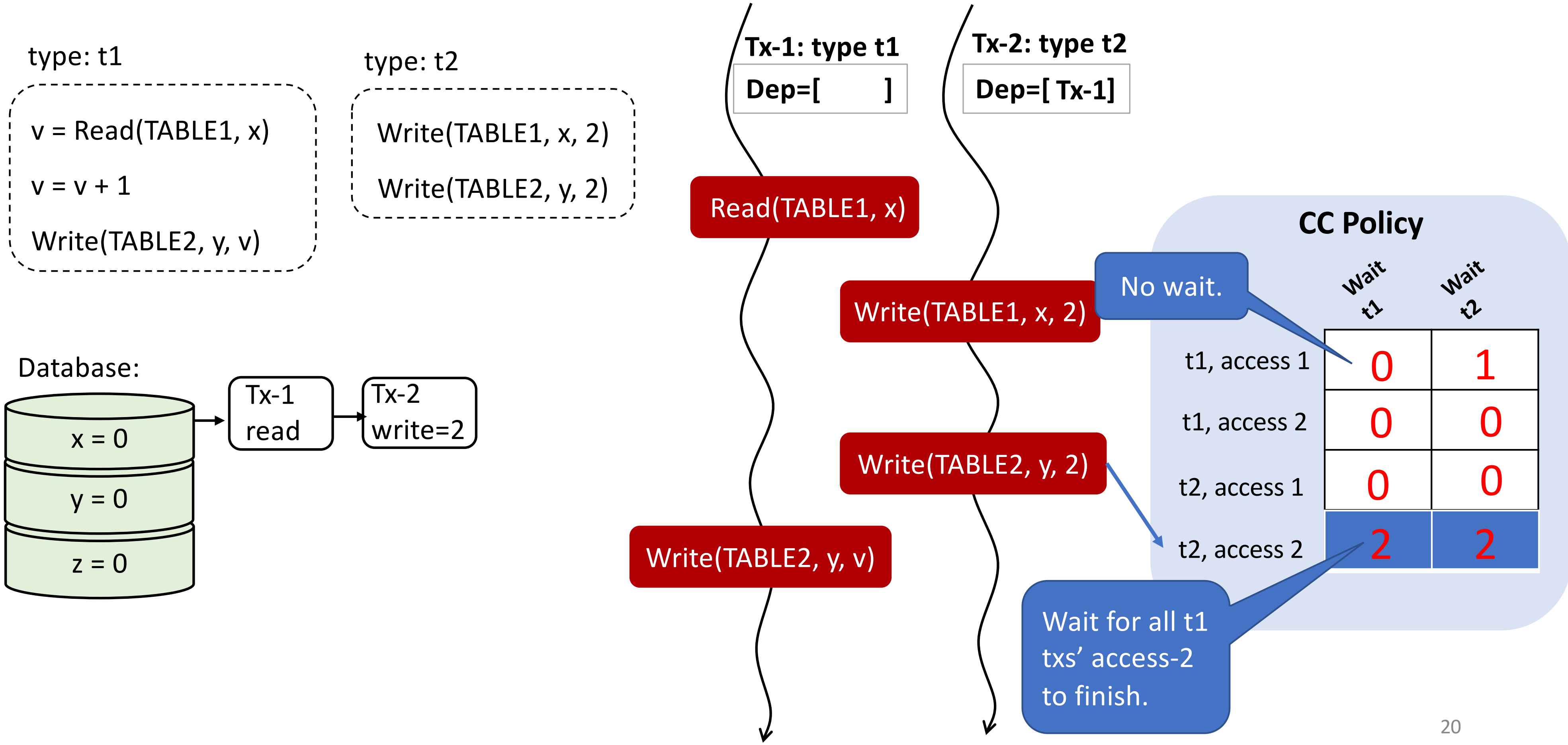
Wait choices: wait commit



Wait choices: no-wait



Wait choices: fine-grained wait



Read action: 2 choices

- Read latest committed version.
  - Polyjuice is a single-version database.
- Read the latest published dirty version.

Used by OCC.

Used by IC3[SIGMOD'16], Callas RP[SOSP'15], DRP[Eurosys'19].

Write action: 2 choices

- Keep the dirty write in the private buffer.
- Publish the dirty write.
  - Cumulative: all buffered are published.

Validation: 2 choices

- Whether or not to validate after this access.
- There is always a final validation prior to commit.
- Polyjuice’s validation is based on Silo[SOSP’13]'s protocol.

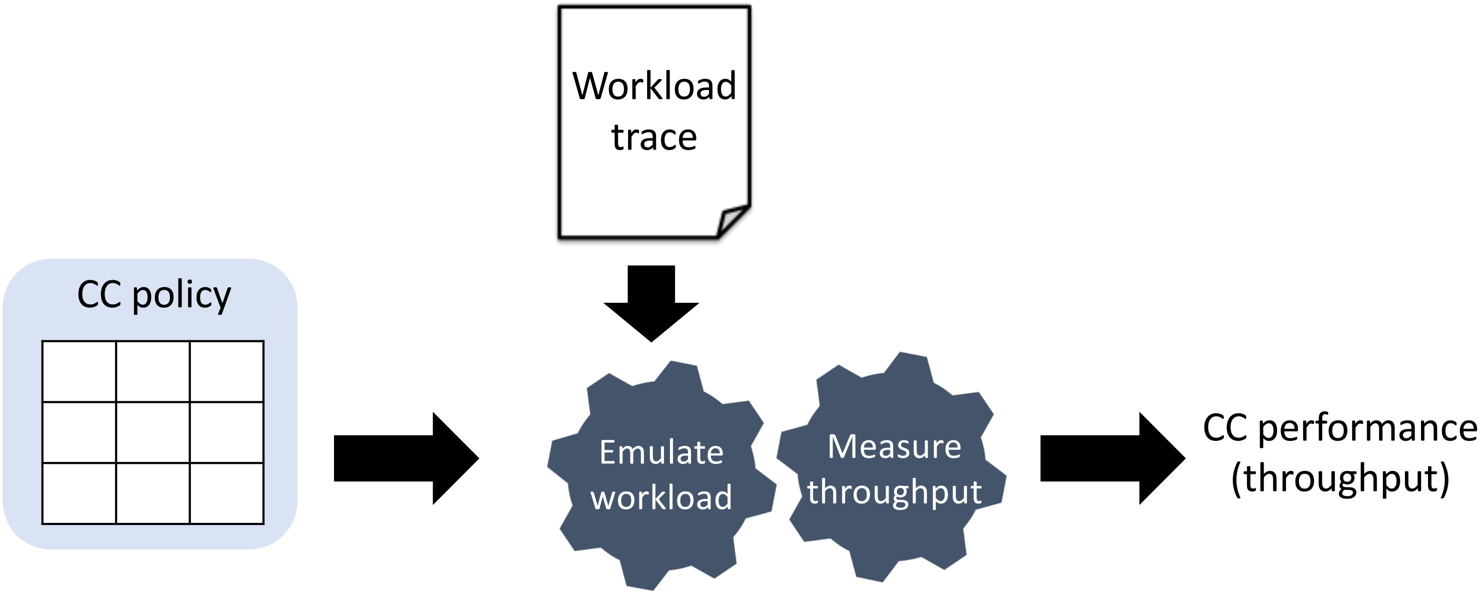
Polyjuice: state and action space

CC Policy

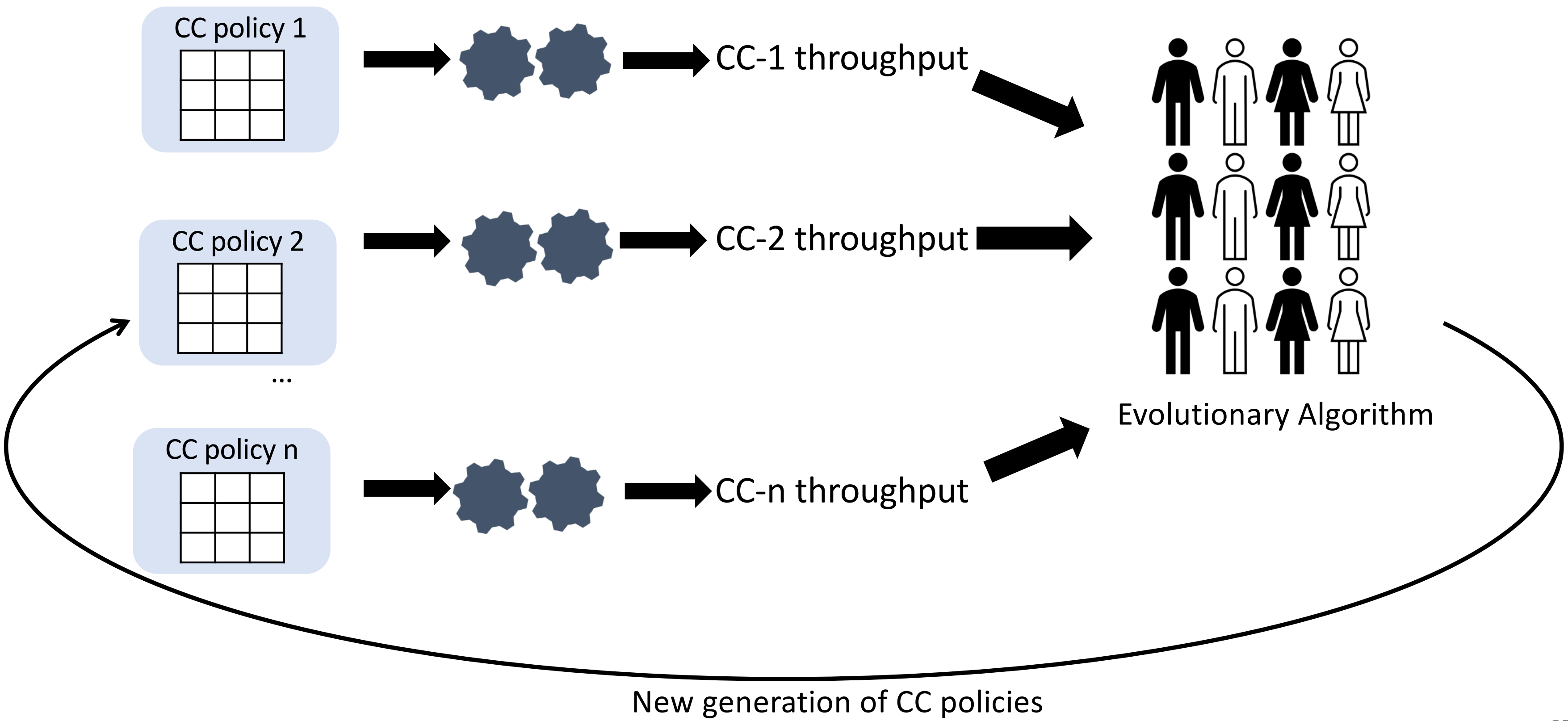
	wait t1	wait …	wait tn	read-dirty?	write-visible?	validate-early?
t1, access 1						
t1, access 2	0	…	2	✗	✓	✓
…						
tn, access m						

- Can encode most existing CC algorithms.
- Enable novel interleavings not permitted by existing CCs.

Optimize policy for a given workload



Optimize policy for a given workload

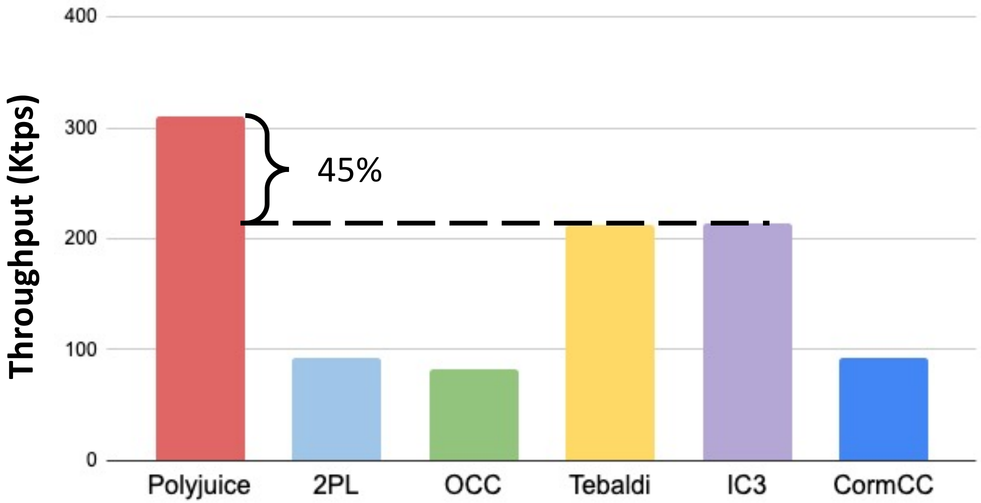


Evaluation

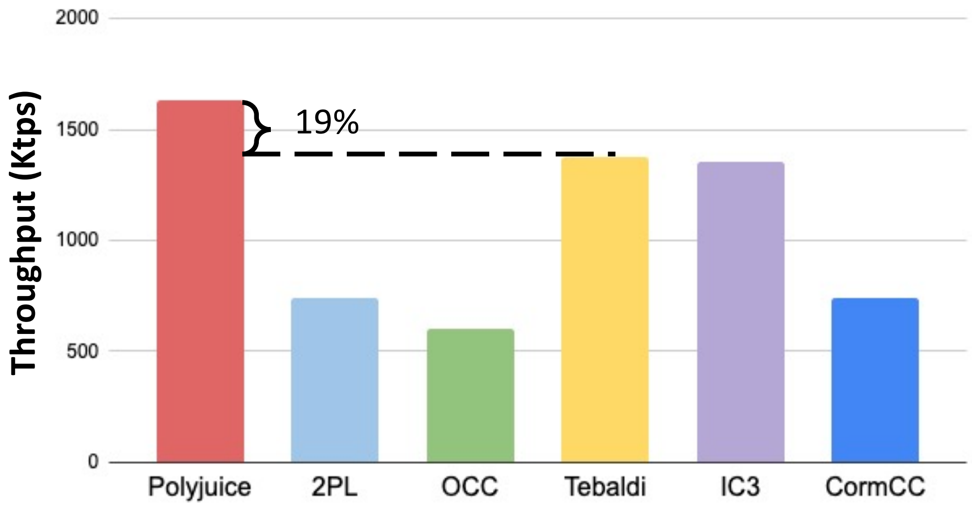
- How does Polyjuice compare to tradition and federated CC?
- Can Polyjuice find novel interleavings?

Polyjuice outperforms existing CCs under high and moderate contention

- Setup: 48-threads, TPC-C (3 read-write txns)

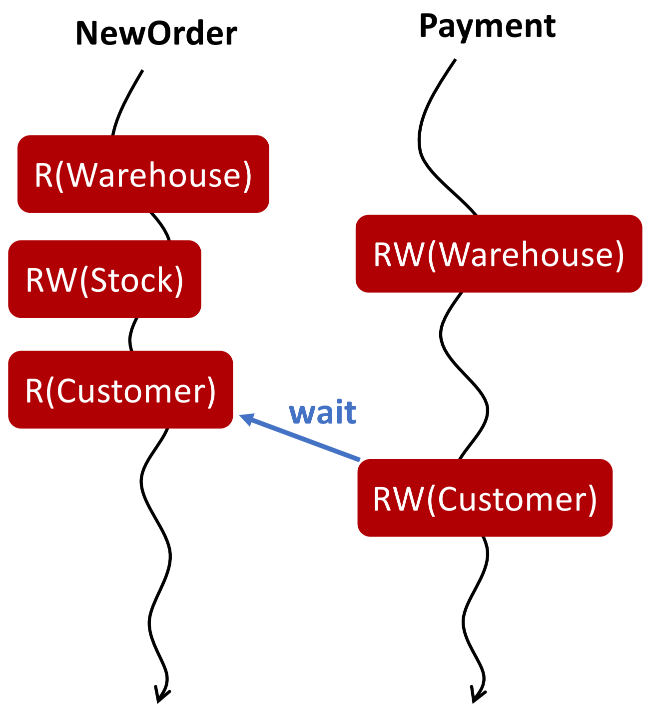


1 warehouse (high contention)

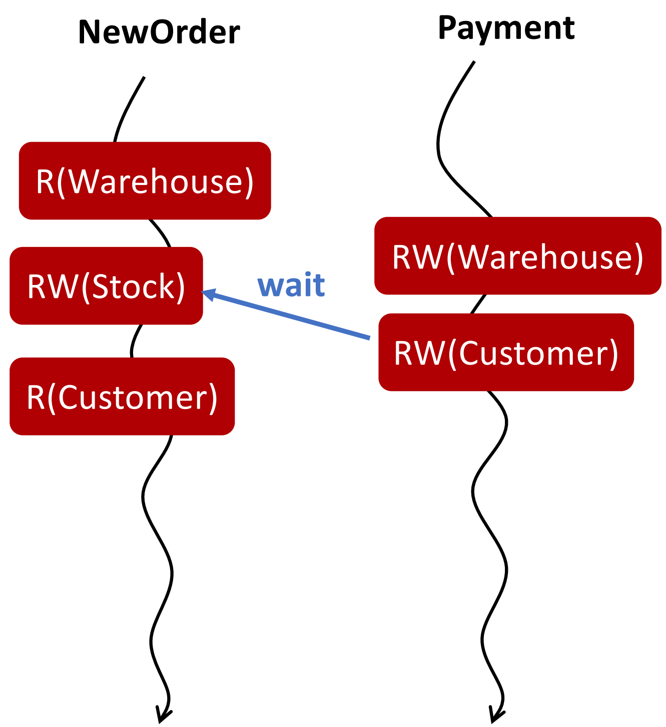


8 warehouse (moderate contention)

Polyjuice finds a more efficient interleaving



IC3's interleaving



Polyjuice's interleaving

Conclusion

- We model CC as a learning task that optimizes policy for a workload
- Polyjuice's policy table design can:
  - encode existing CCs
  - allow new interleaving
- Polyjuice can outperform existing fixed and federated CC

https://github.com/derFischer/Polyjuice