All about Eve: Execute-Verify Replication for Multi-Core Servers

Manos Kapritsos, Yang Wang, Vivien Quema,
Allen Clement, Lorenzo Alvisi, Mike Dahlin

Dependability



Databases

key	value
firstName	Bugs
lastName	Bunny
location	Earth

Key-value stores

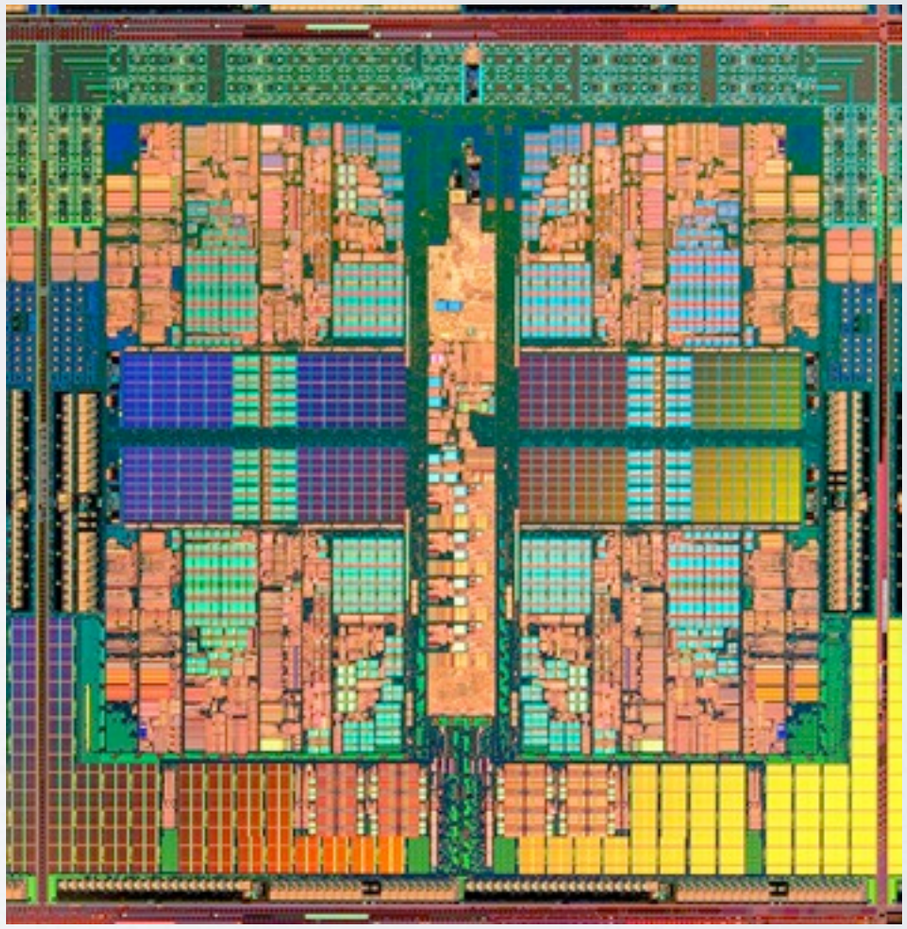


Coordination & locking



File servers

Multi-core



Dependability  Multi-core

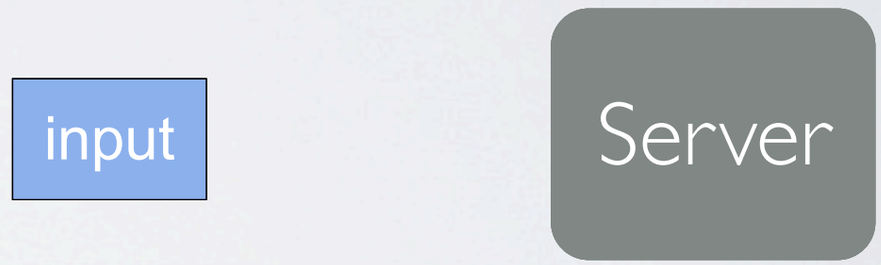
How do we build dependable ~~multithreaded~~ services?

Answer:
State Machine Replication

STATE MACHINE REPLICATION (SMR)

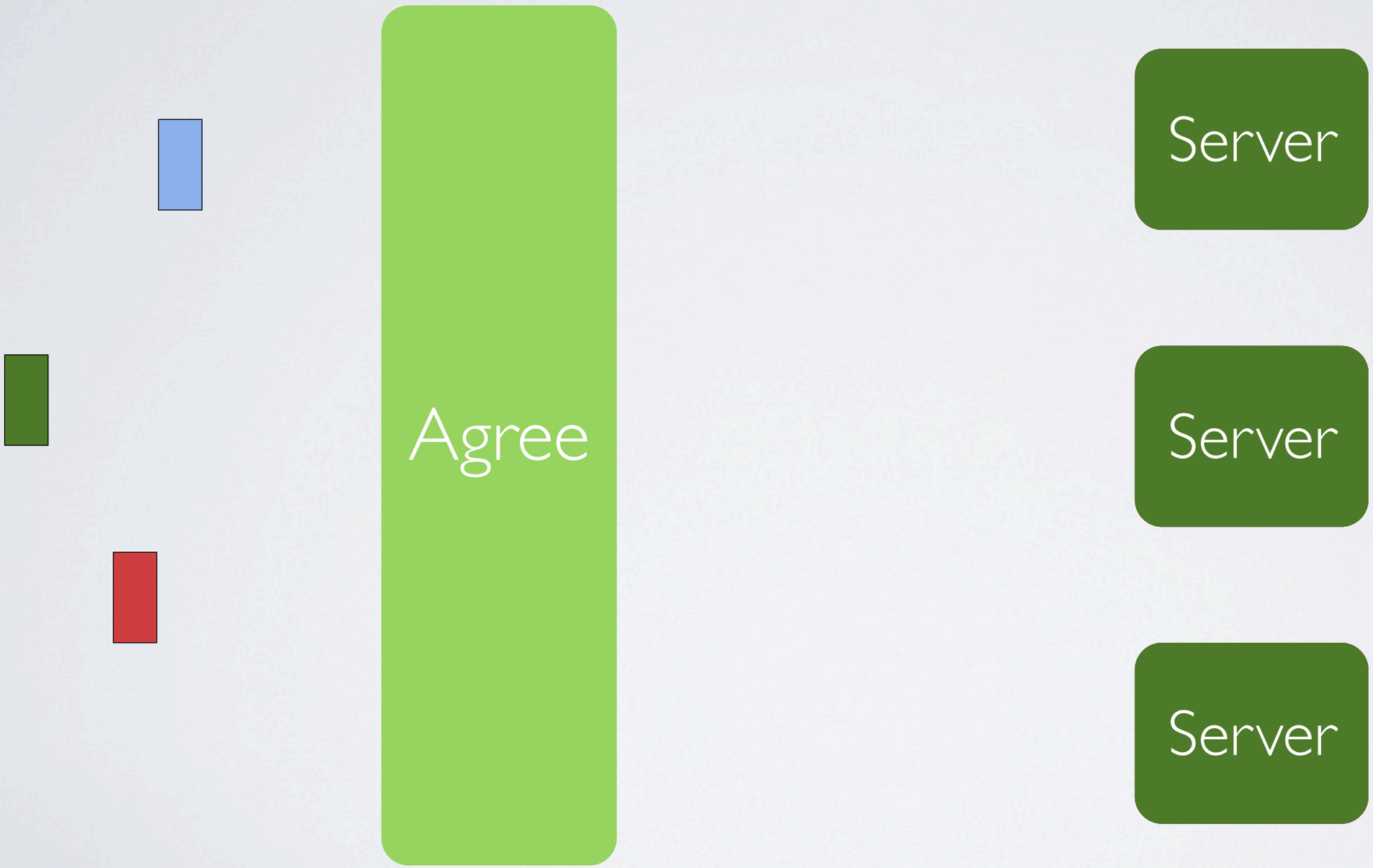
Ingredients: a service

1. Implement service as a deterministic state machine
2. Replicate
3. Provide all replicas with the same input



Guarantee: all correct replicas will produce the same output

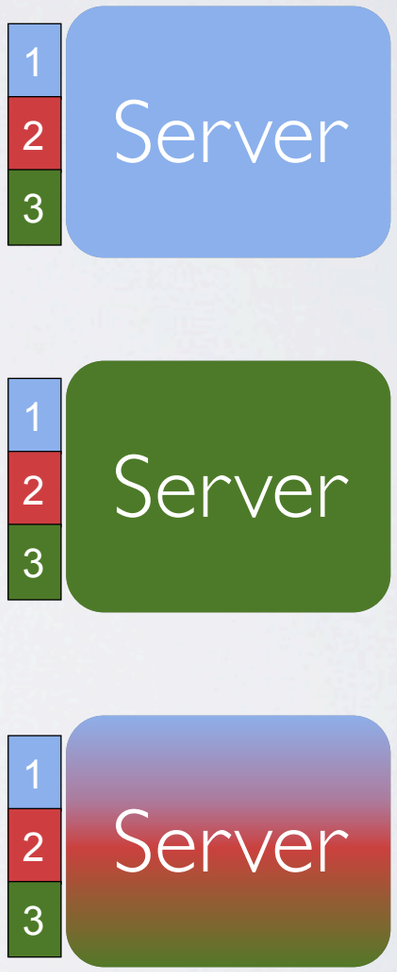
SMR IMPLEMENTATION



How do we build dependable ~~multithreaded~~ services?



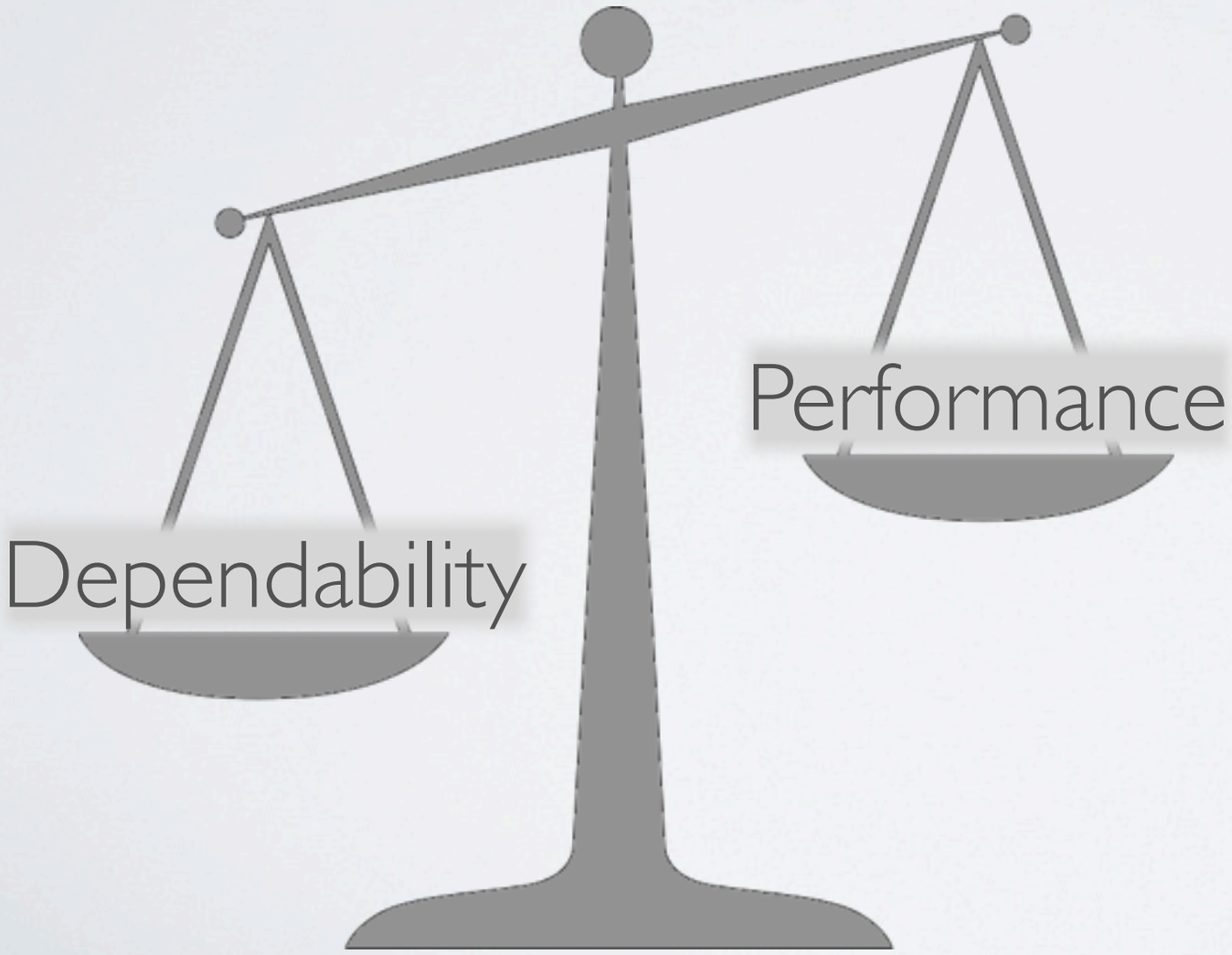
Maybe use deterministic multithreading?



Nope. Won't support modern replication protocols



How do we build dependable multithreaded services?



Server

Server

Server

Eve

State machine replication with multithreaded execution

Outline

Motivation

Insight

Mechanisms

Architecture

Evaluation

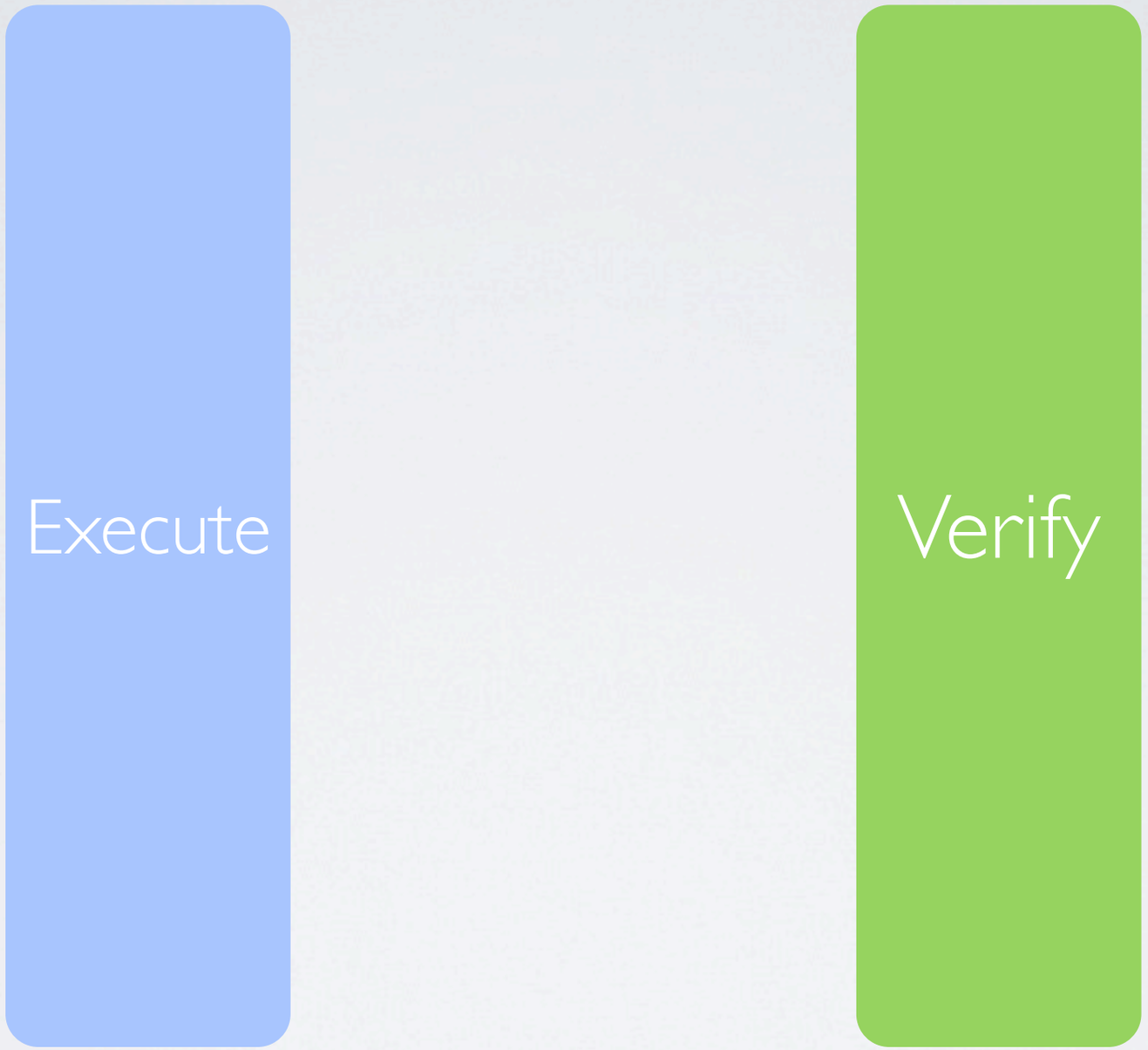
SMR requires replica convergence

Agree

Execute

Agree-Execute enforces sequential execution

EXECUTE-VERIFY



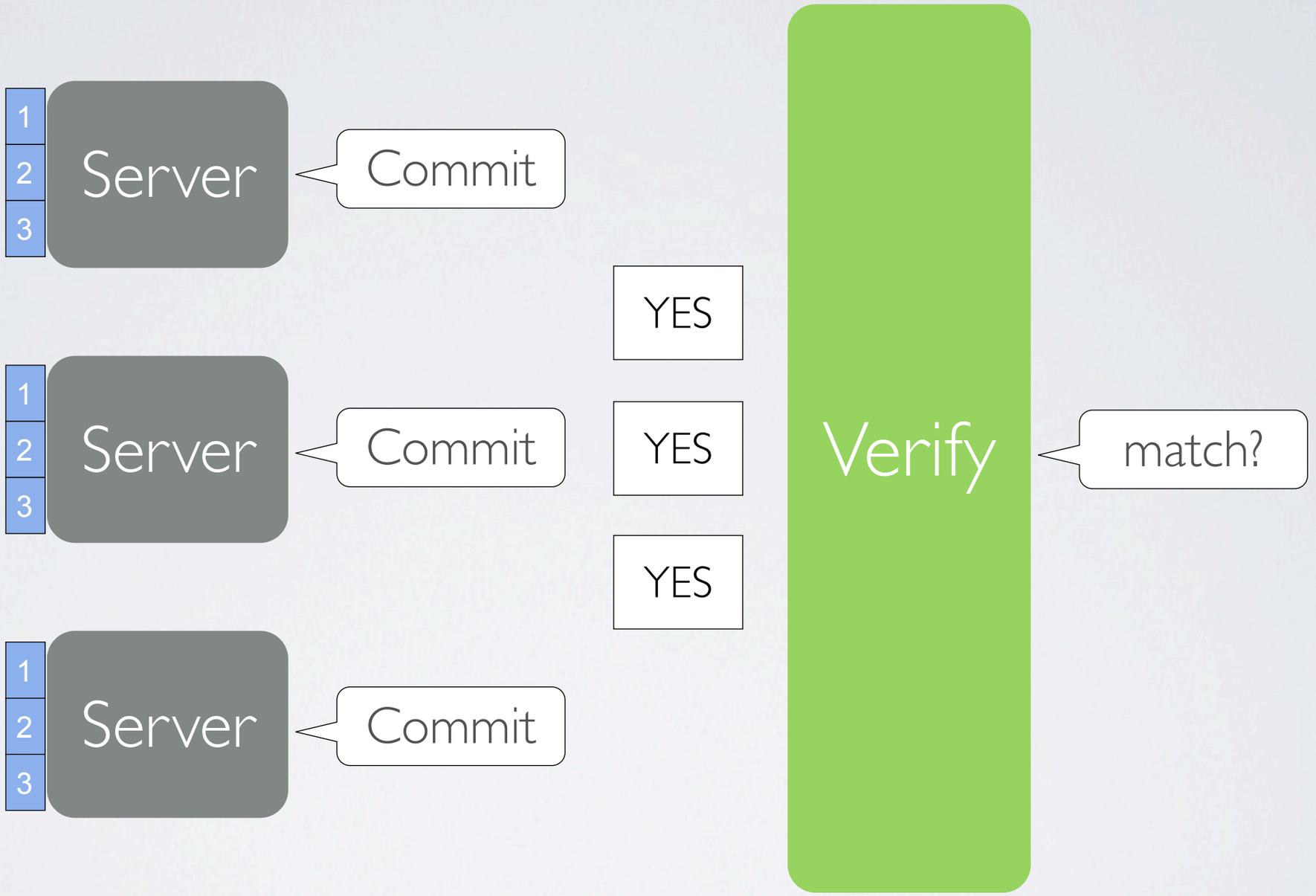
First execute...

(multithreaded and without agreeing on the order)

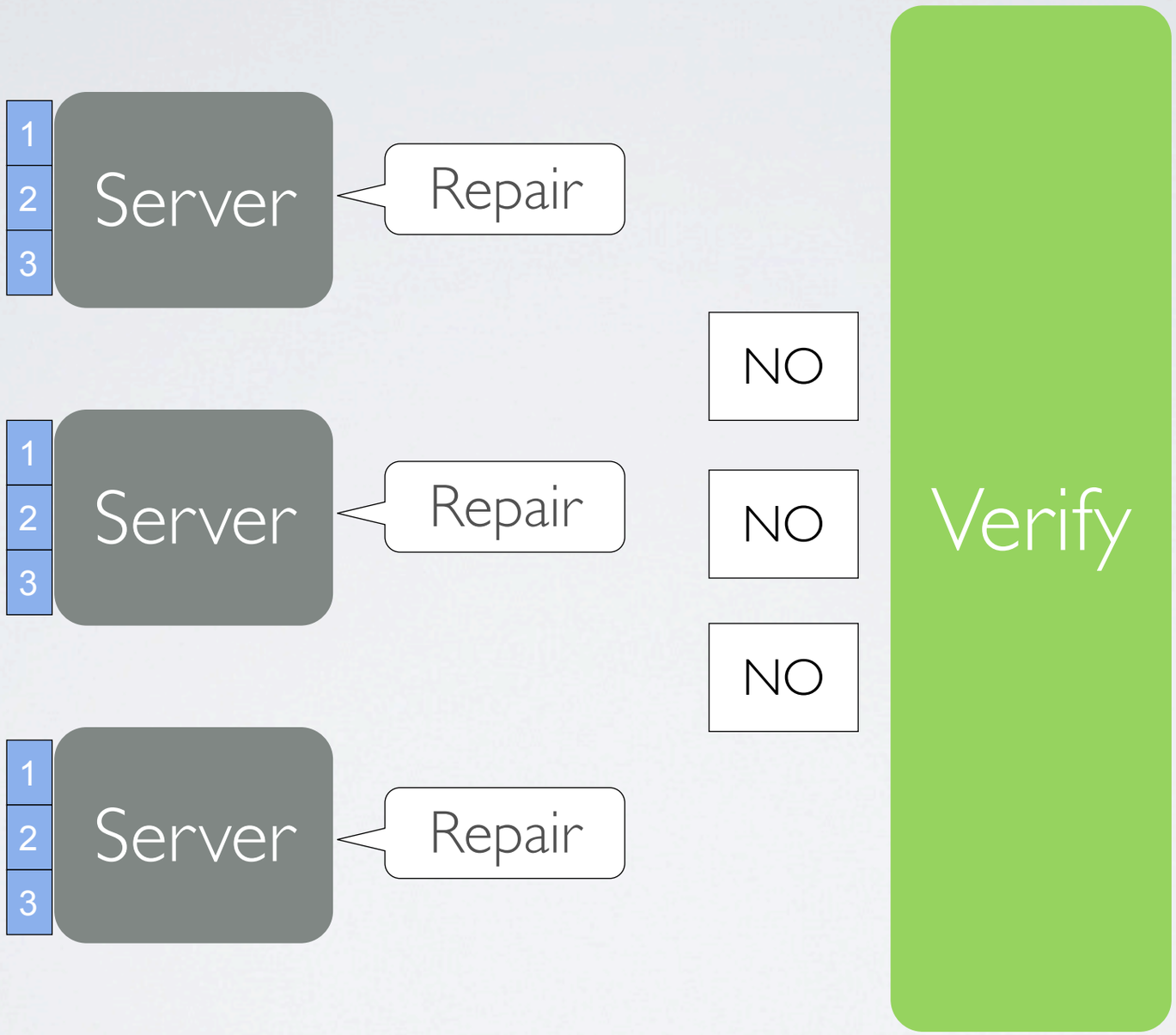
...then verify

(that replicas agree on the outcome)

ON CONVERGENCE



ON DIVERGENCE



Repair: rollback and re-execute sequentially

Outline

Motivation

Insight

Mechanisms

Architecture

Evaluation

Eve's logic at a glance

Frequent

```
if (converged)
    commit
else
    repair divergence
```

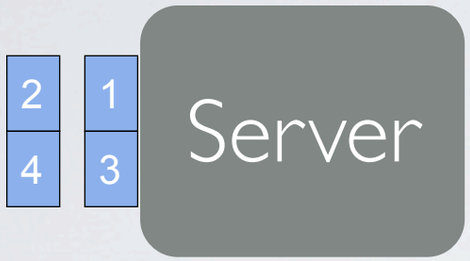
Uncommon

1. Make divergence uncommon

2. Detect divergence efficiently

3. Repair divergence efficiently

MAKING DIVERGENCE UNCOMMON




```
if (converged)
    commit
else
    repair divergence
```





Idea: identify commutative requests

Mixer: group together commutative requests

- Execute requests within a group in parallel

Mixer is a hint, not an oracle

EXAMPLE: TPC-W MIXER

Transaction	Read tables	Write tables
getBestSellers	item, author, order_line	
doCart	item	shopping_cart_line, shopping_cart
doBuyConfirm	customer, address	order_line, item, cc_xacts, shopping_cart_line

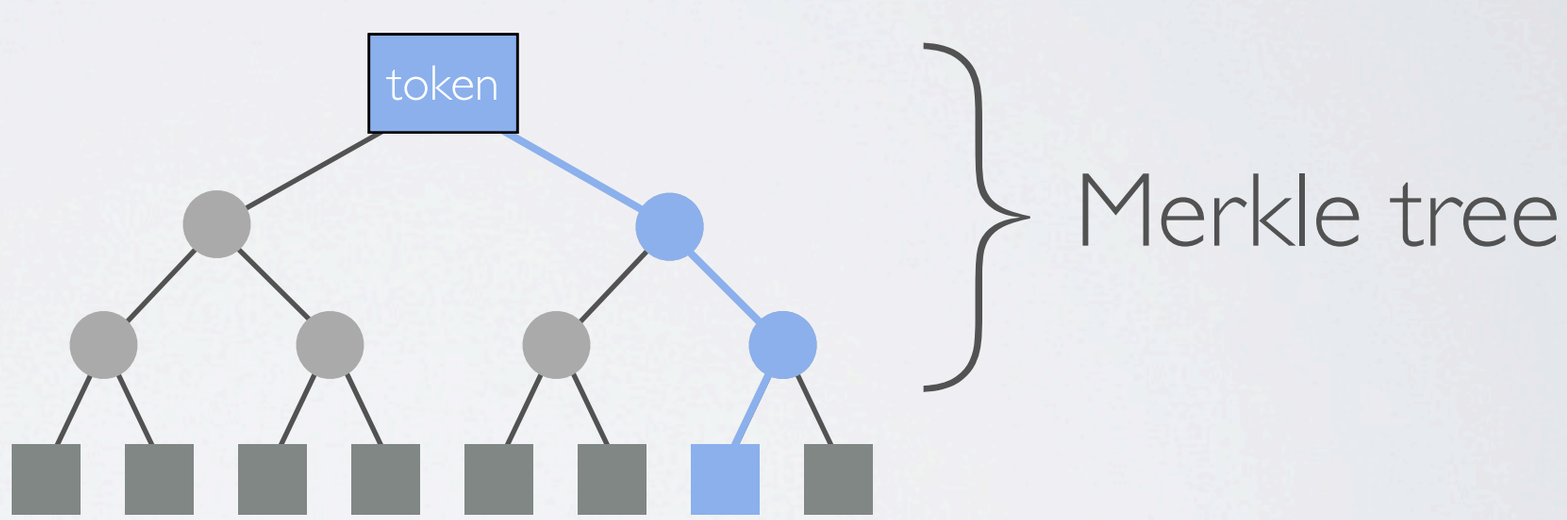
3 frequent transactions of the TPC-W browsing workload

EFFICIENT DIVERGENCE DETECTION

Need to compare application states & responses frequently

```
if (converged)
    commit
else
    repair divergence
```

Application state



GROWING DETERMINISTIC MERKLE TREES

Idea: postpone adding objects until token generation

Ensure that all replicas add objects in the same order

Requests are ordered: **requestID**

Single thread per request: **objectSeqNumber**

(requestID,objectSeqNumber): unique and sortable

Optimization: leverage deterministic order of references

EFFICIENT DIVERGENCE REPAIR

```
if (converged)
    commit
else
    repair divergence
```

Need to rollback application states after every divergence



```
if (converged)
    commit
else
    repair divergence
```

1. Make divergence uncommon

Mixer

2. Detect divergence efficiently

Merkle tree

3. Repair divergence efficiently

Copy-on-Write

Outline

Motivation

Insight

Mechanisms

Architecture

Evaluation

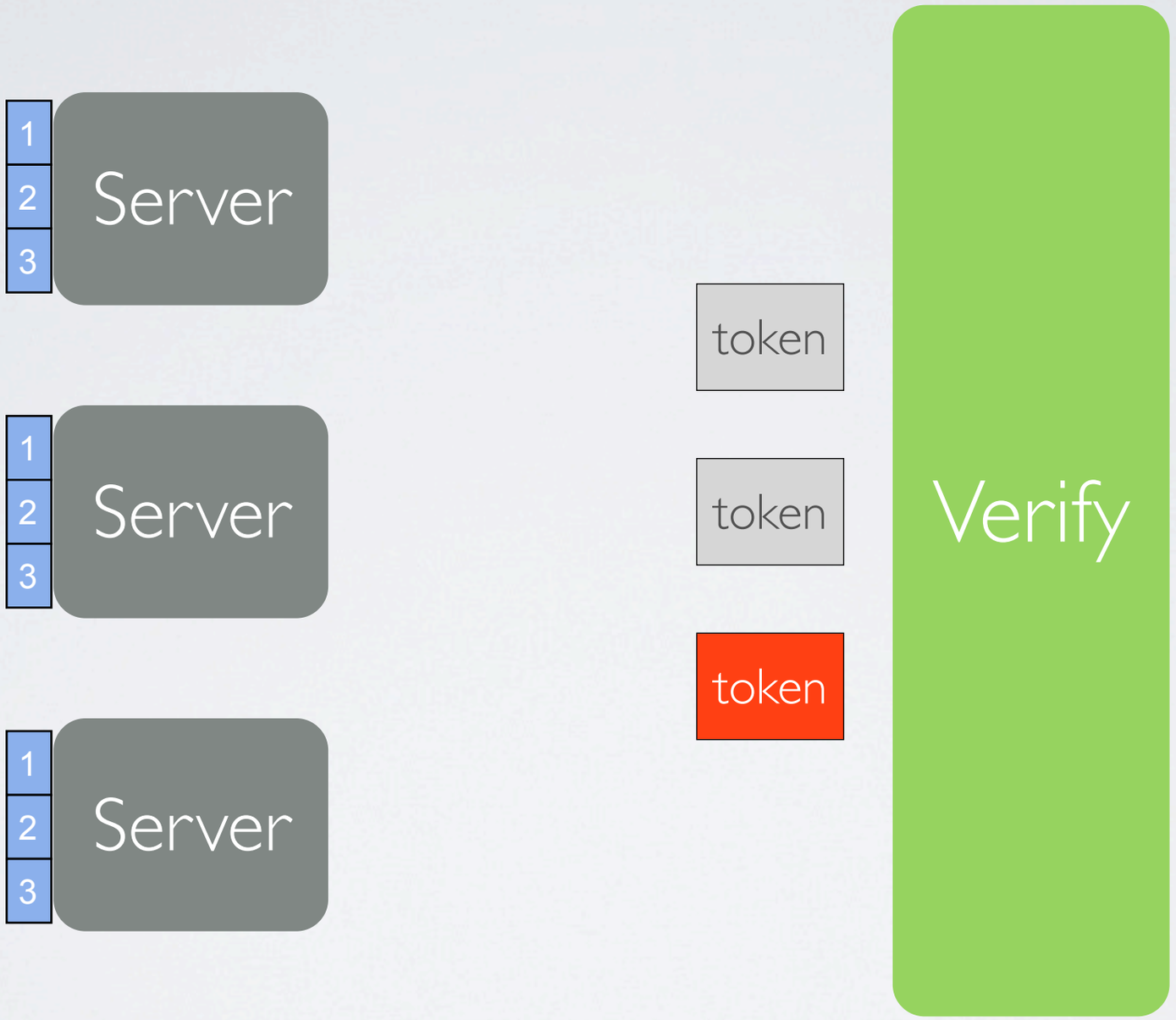
Dependability ❤ Performance

Independent execution ❤ Non-deterministic order of requests

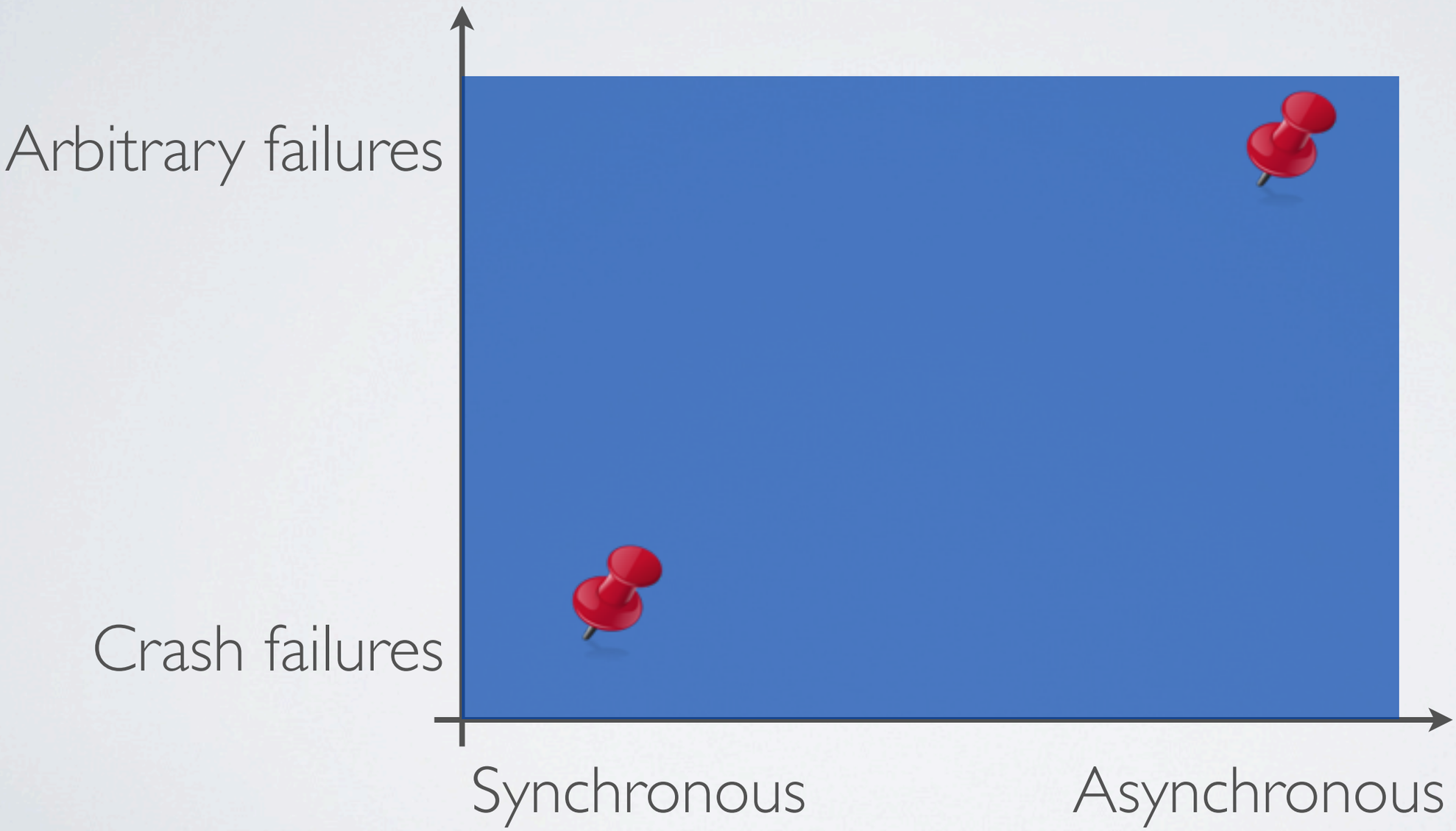
Replication of multithreaded services

Bonus: mask concurrency bugs

MASKING CONCURRENCY BUGS



EXECUTE-VERIFY: AN ARCHITECTURAL CHANGE

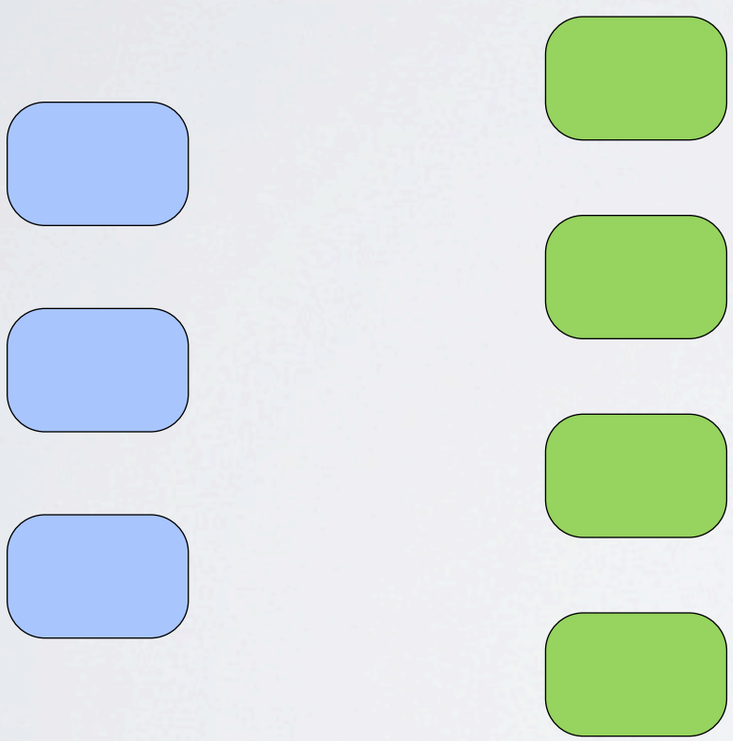


CONFIGURATIONS

Asynchronous BFT

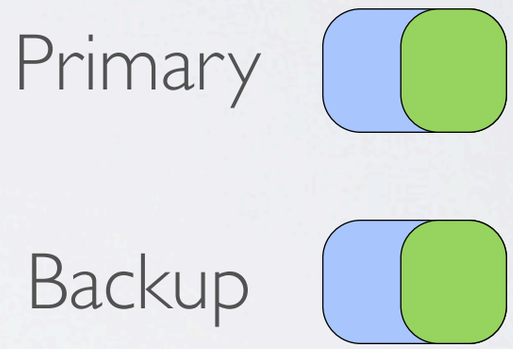
Execution

Verification



Tolerates 1 arbitrary fault

Synchronous primary-backup



Tolerates 1 omission fault

EVALUATION

What is the performance benefit of Eve compared to traditional SMR systems?

How does the quality of the mixer affect Eve's performance?

EXPERIMENTAL SETUP

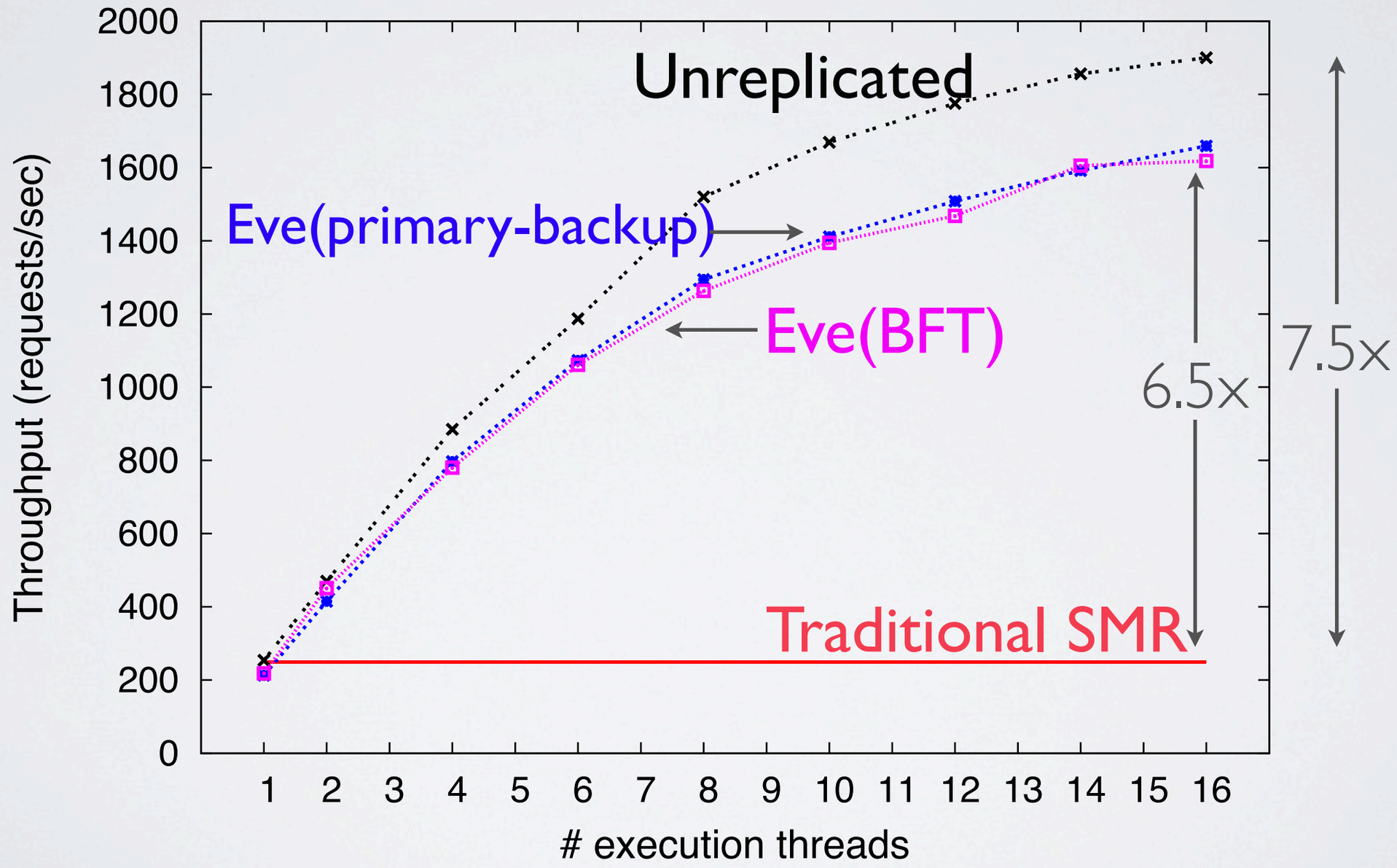
Emulab testbed deployment

- Execution replicas: 16 cores

Applications

- H2 Database Engine (TPC-W benchmark)
- Key-value store (Microbenchmarks)

Application: H2 Database Engine
Workload: TPC-W (browsing)



IMPACT OF THE MIXER

Application: Key-value store

Number of key-value pairs

- Determines available parallelism

Mixer Quality

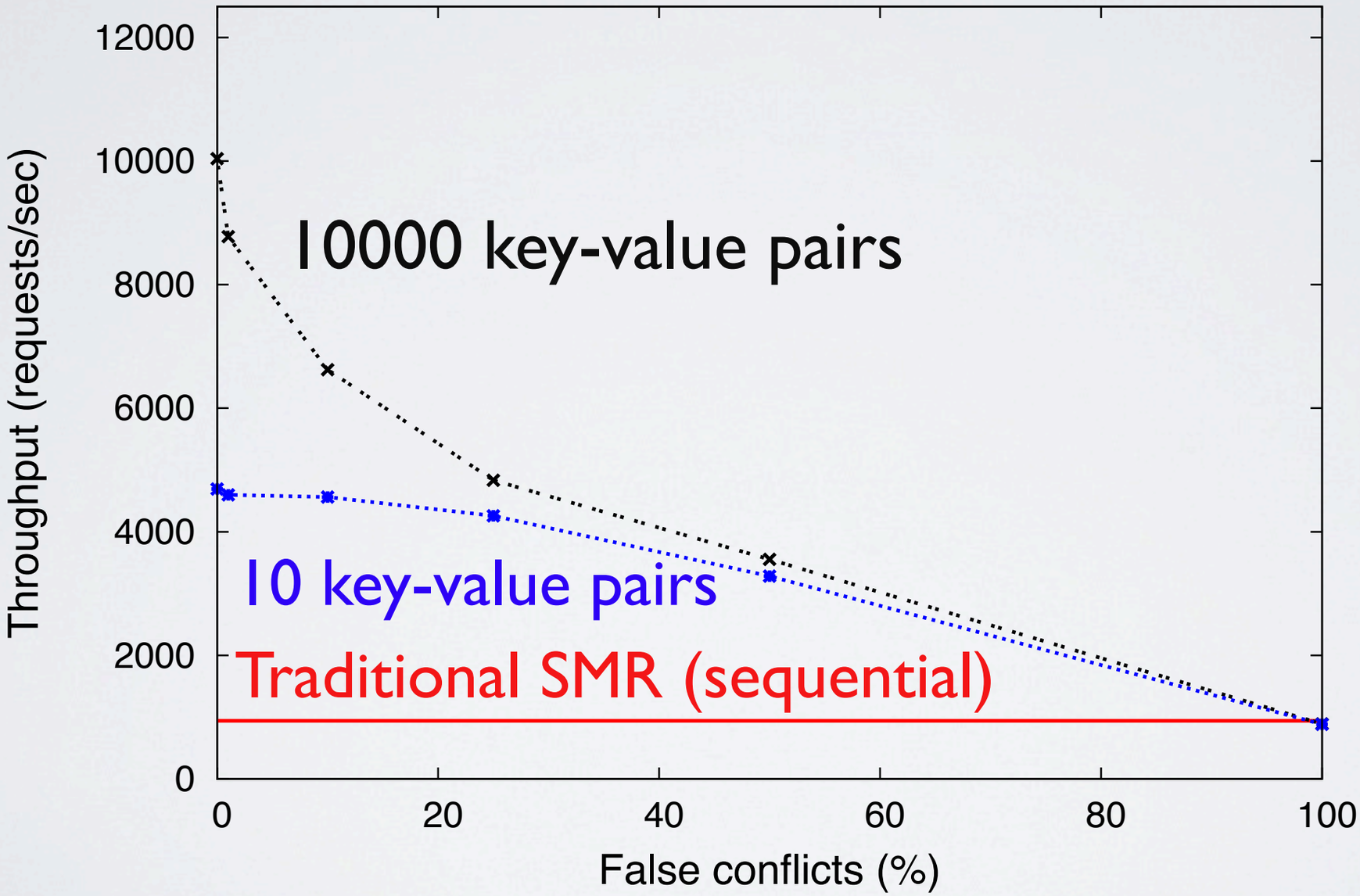
False conflicts: misclassify non-conflicting requests as conflicting

- Reduces parallelism

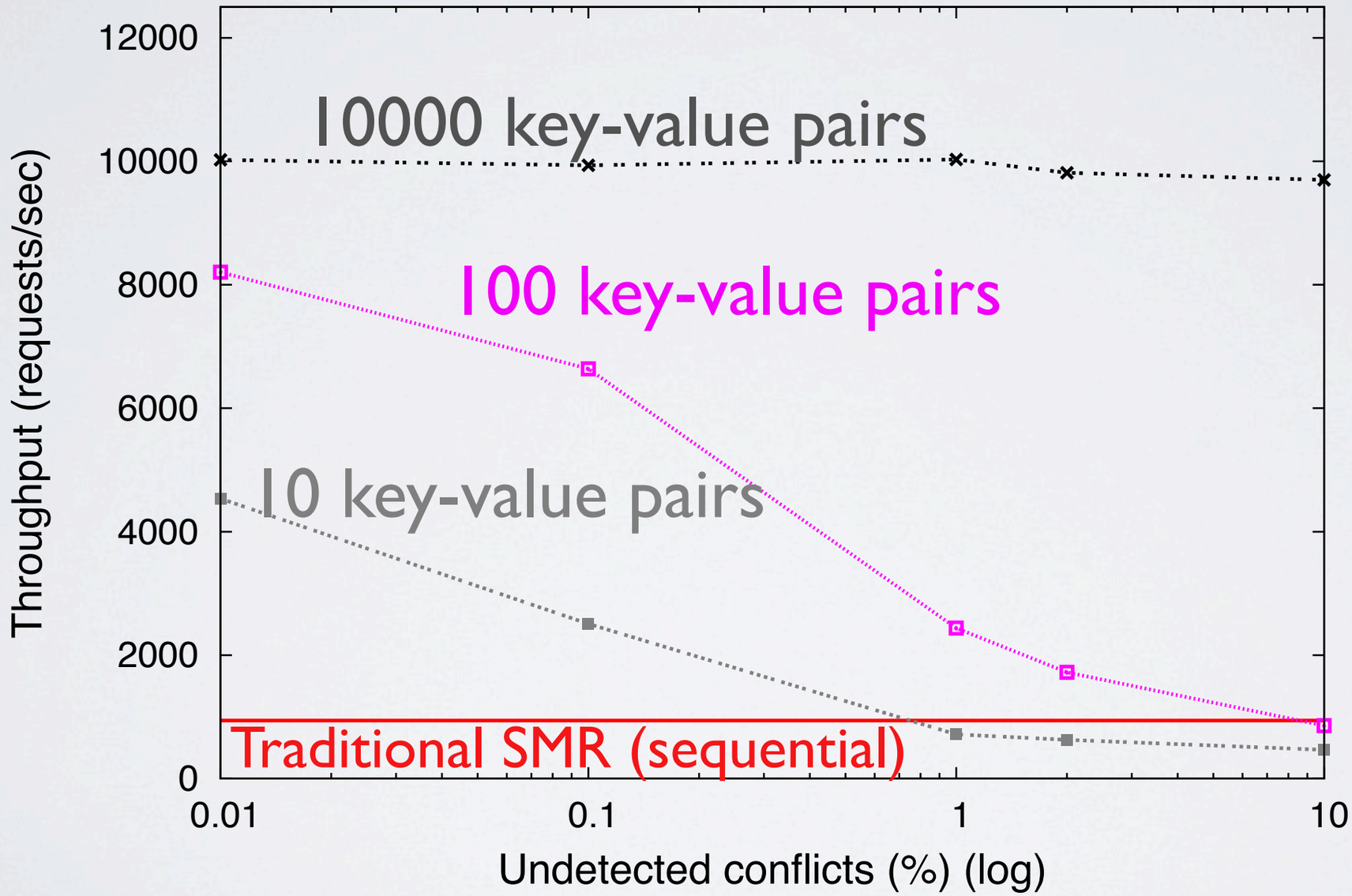
Undetected conflicts: misclassify conflicting requests as non-conflicting

- Can introduce divergence

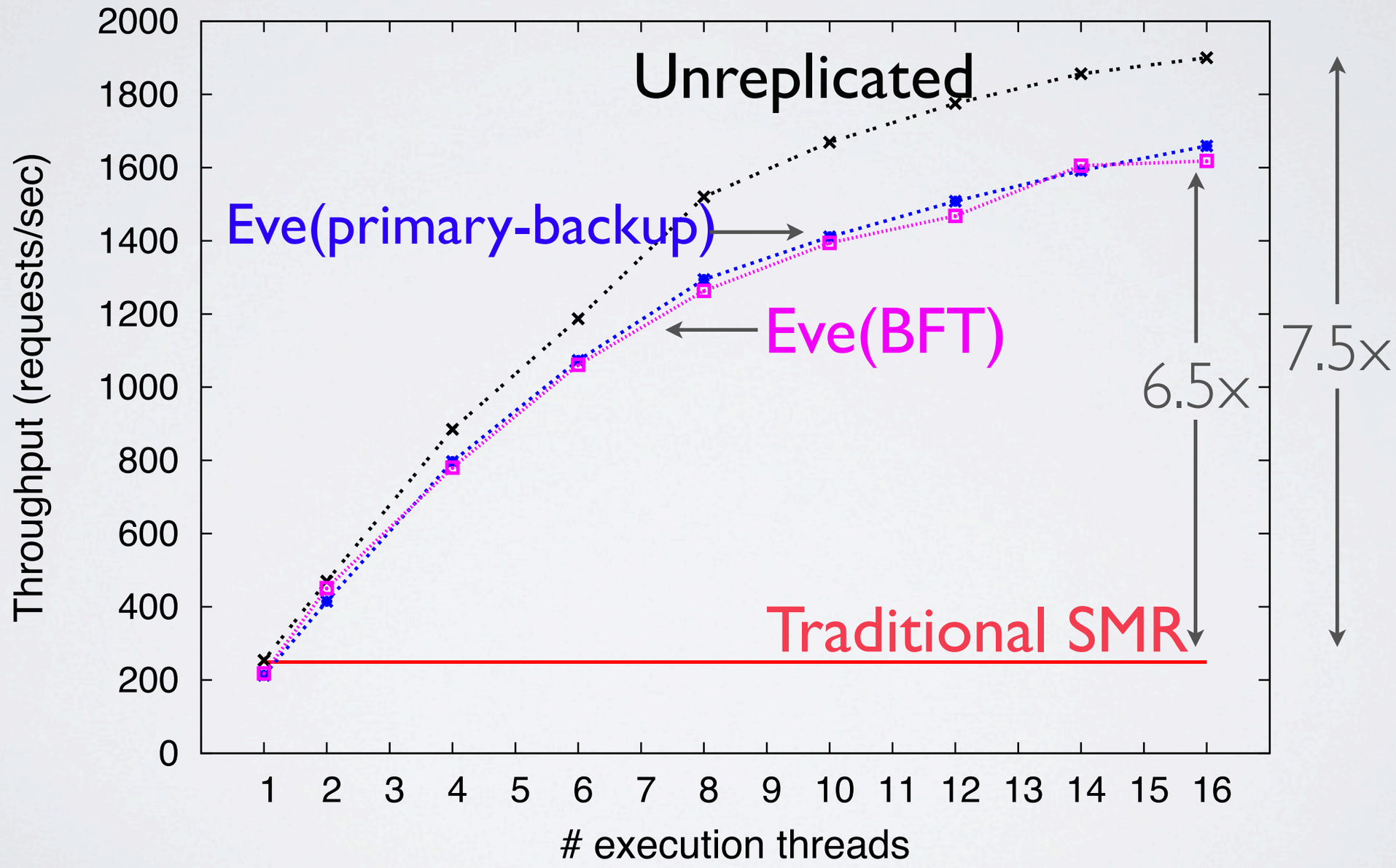
FALSE CONFLICTS REDUCE THE AVAILABLE PARALLELISM



UNDETECTED CONFLICTS CAUSE DIVERGENCE AND ROLLBACKS



TPC-W EXPERIMENTS: NO ROLLBACKS OBSERVED



CONCLUSION

Replication and multithreading are
not mutually exclusive

Redesign replication:
from agree-execute
to **execute-verify**

