What consistency does your key-value store *actually* provide?

Eric Anderson, Xiaozhou (Steve) Li, Mehul Shah, Joseph Tucek, and Jay Wylie
HP Labs
HotDep 2010
October 3, 2010

Outline

- Key-value stores
- Consistencies
- Checking consistencies
- Algorithms
- Findings

KEY-VALUE STORES

amazon web services™ — Simple Storage Service (S3), Dynamo

Google code labs

Google Storage for Developers

Microsoft® SQL Azure™

Cassandra

Tokyo Cabinet 8192PiB

Project Voldemort
A distributed database.

CONSISTENCIES

amazon web services™

Eventually consistent

Google code labs

Read-your-writes

Quorum-based, multiple levels

Cassandra

Project Voldemort

A distributed database.

Read-repair, vector clocks, hinted hand-off

Tokyo Cabinet 8192PiB

Sequential writes

DO YOU BELIEVE THEM?

Why not?

WHY DO YOU WANT TO KNOW?

Verify SLAs that may contain consistency guarantees

WHY DO YOU WANT TO KNOW?

Choose the one that meets your consistency requirements

Tokyo Cabinet 8192PiB

Microsoft® SQL Azure™

Google code labs

amazon web services™

Cassandra

Project Voldemort
A distributed database.

WHY DO YOU WANT TO KNOW?

Choose a proper service level for own workload

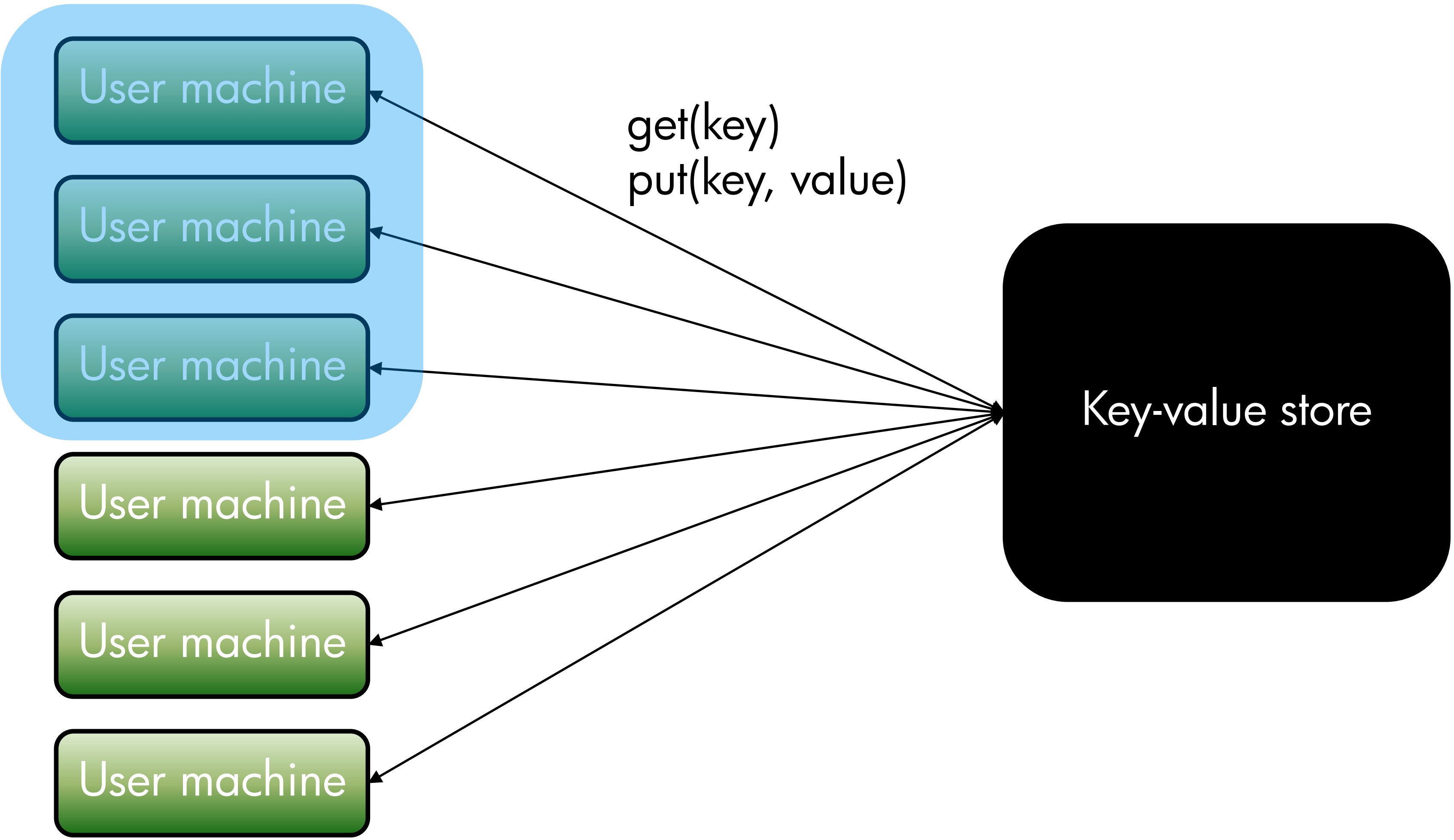
- What you pay is what you get
- What you get depends on your workload
- Tough workloads & failures: Worse than expected / promised
- Benign workload & good operating conditions: Better than minimal guarantee

WHAT CAN A USER DO?

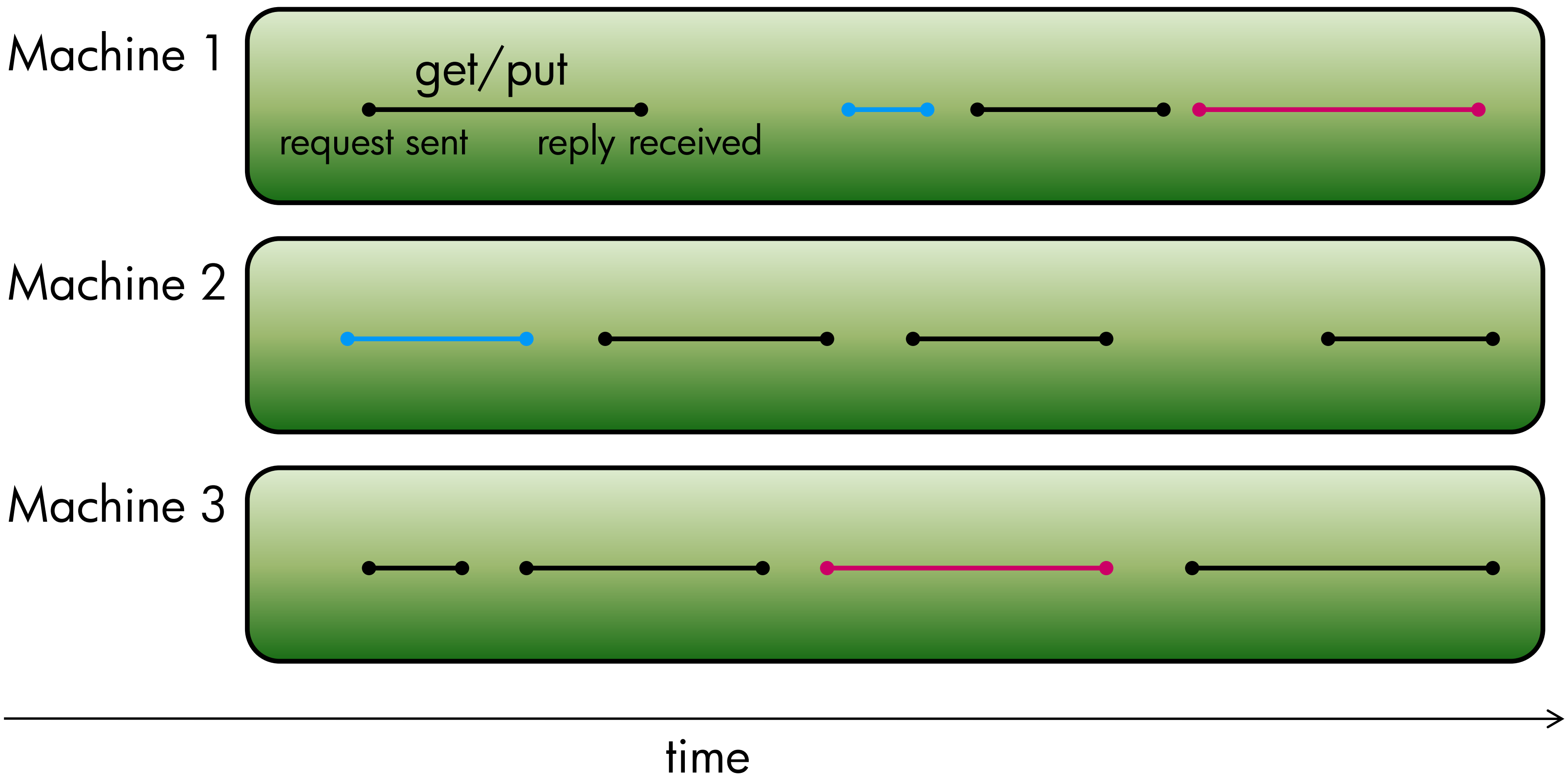
If we know the internal protocols …

WHAT CAN A USER DO?

If we don't know the internal protocols …



CLIENT TRACES



REGISTER-BASED CONSISTENCY

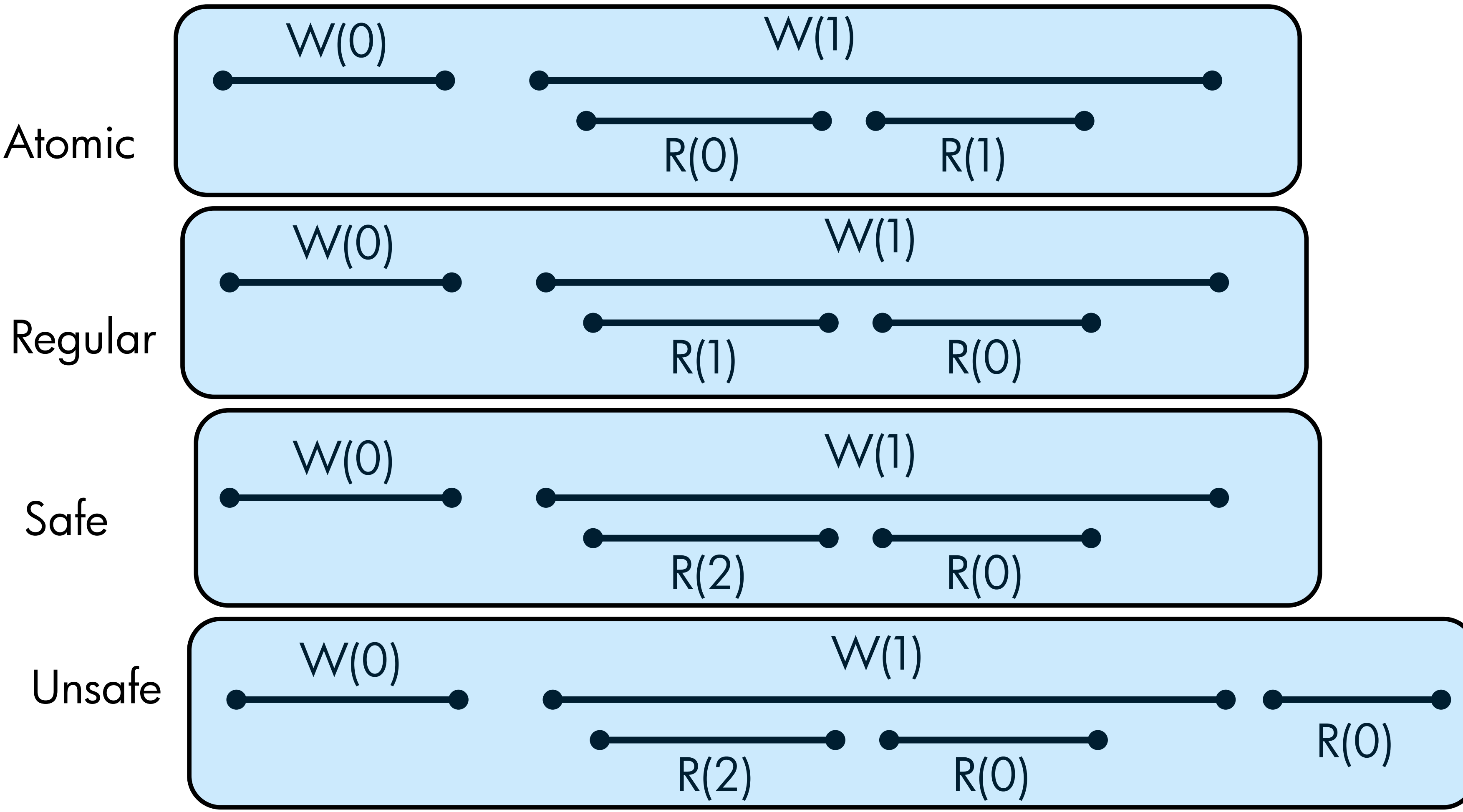
[Lamport, Distributed Computing, 1986]

– Atomic

– Safe

– Regular

ATOMIC/REGULAR/SAFE



OVERALL APPROACH

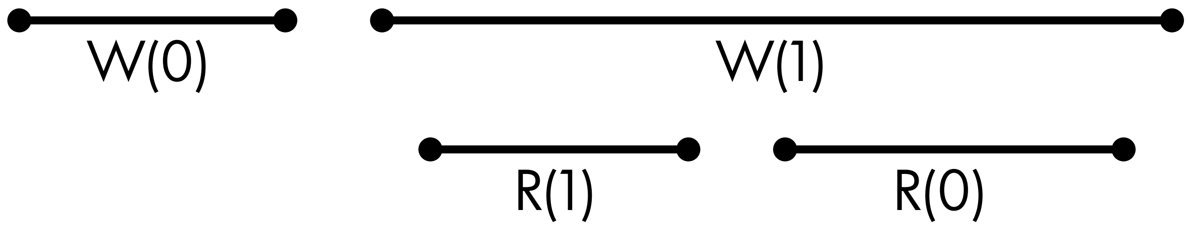
For all three: safe, regular, atomic

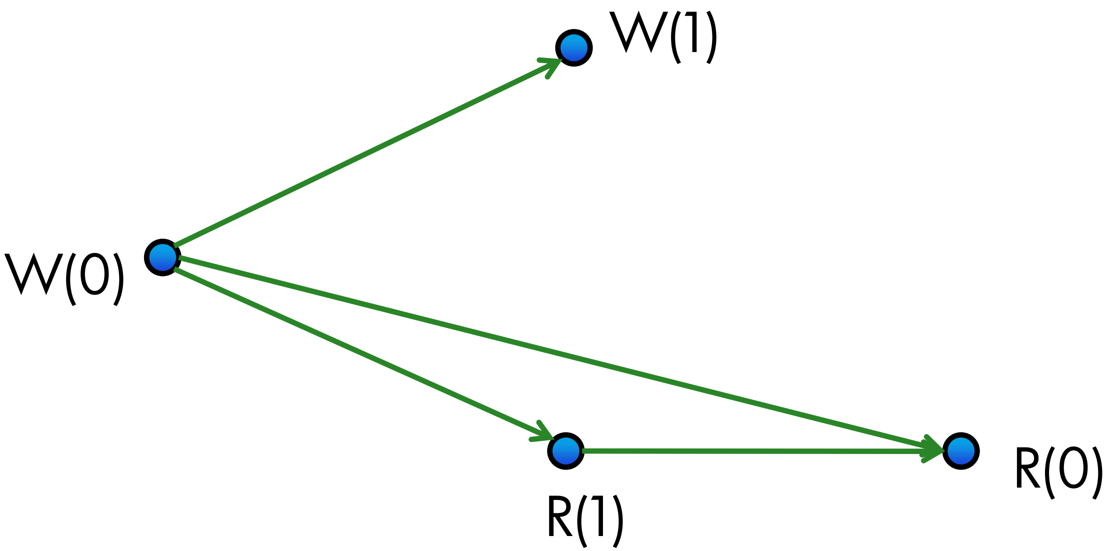
1. Construct a digraph
   - Vertices = operations
   - Edges = precedence
2. Add edges
   - Time
   - Data
   - Hybrid
3. Check if graph is DAG

ASSUMPTIONS

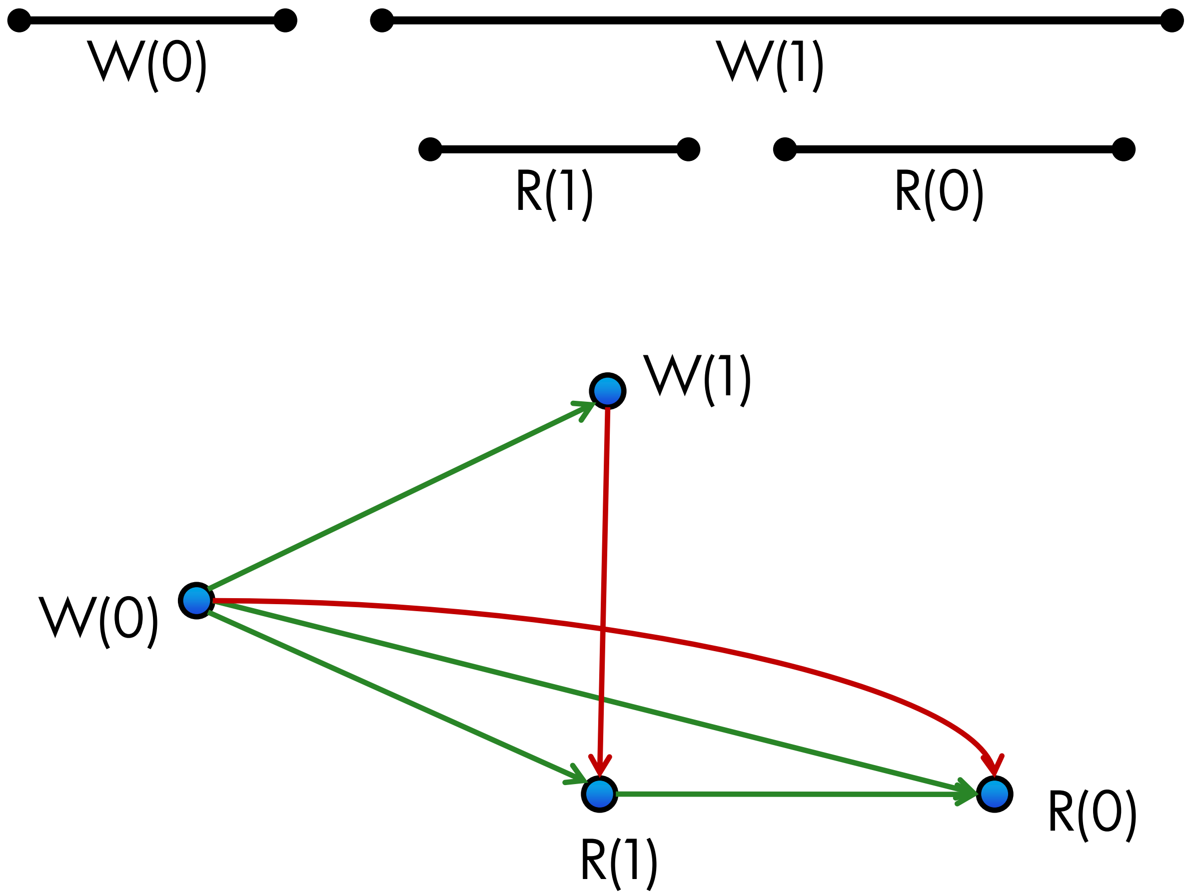
- Client timestamps are reasonably synchronized
- Or they are calibrated during merge
  - Chirp [Anderson et al., MASCOTS, 2009]
- All writes write a distinct value
- There is a default value for each key

ADDING TIME EDGES

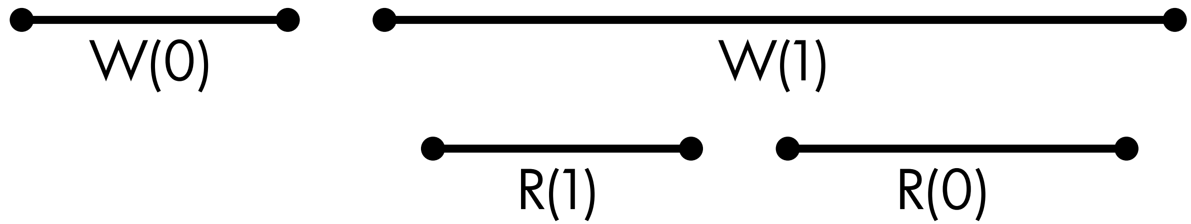


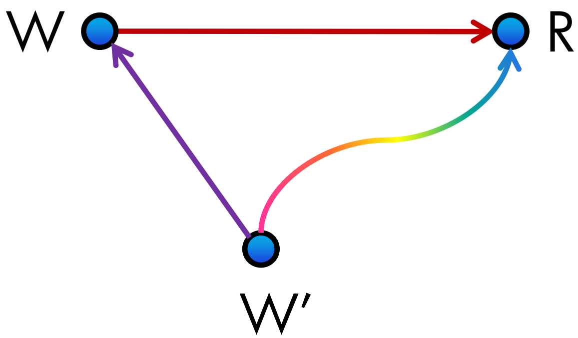


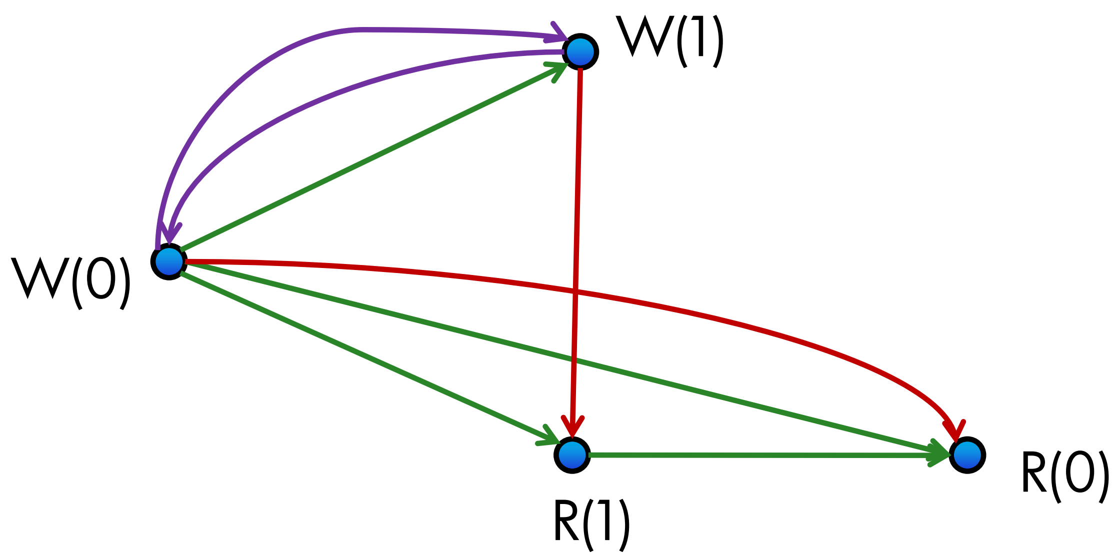
ADDING DATA EDGES



ADDING HYBRID EDGES

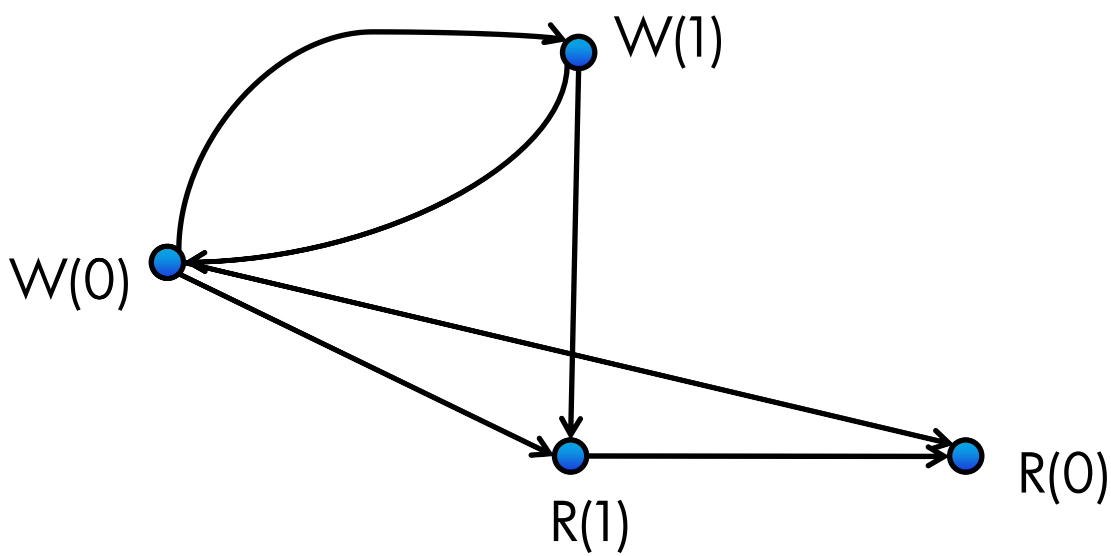






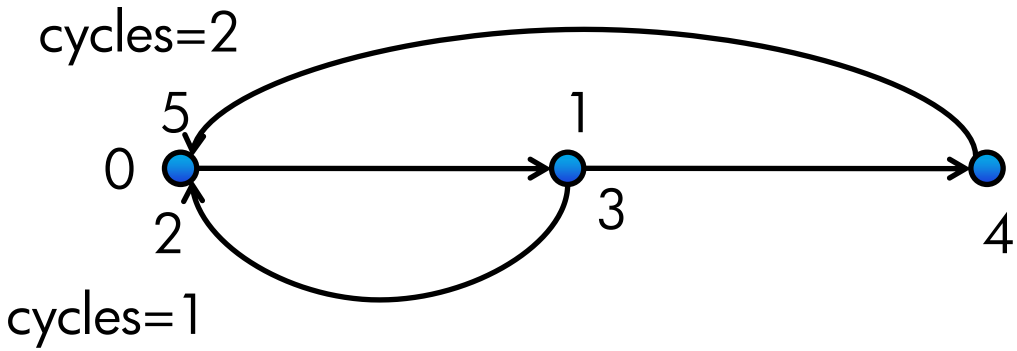
DETECTING CYCLES

DFS



COUNTING VIOLATIONS

Number of cyles found in DFS

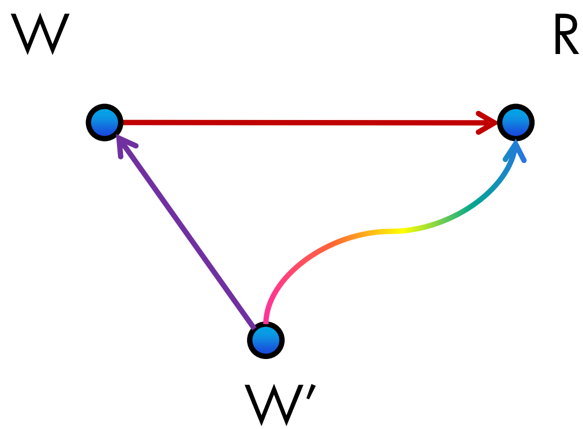


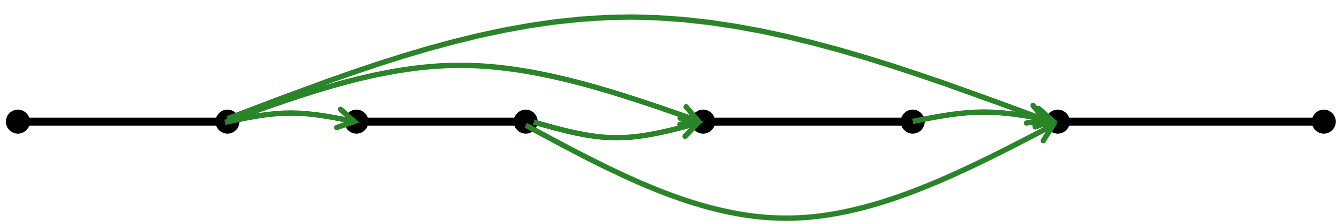
Feedback arc set
Feedback vertex set

CHECKING REGULARITY AND SAFETY

	Atomicity	Regularity	Safety
1	Keep all reads and writes	Remove reads that read a concurrent write's value	Remove all reads that are concurrent with some writes
2	Add time edges		
3	Add data edges		
4	Add hybrid edges		

REDUCING NUMBER OF TIME EDGES





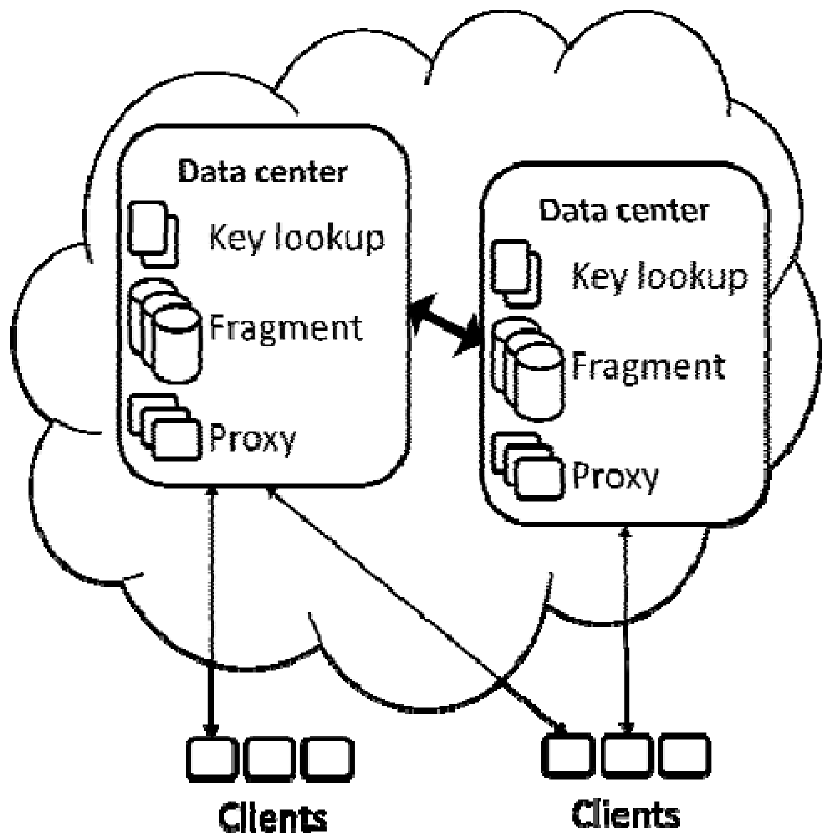
Edges $m=n^2$ even in typical cases; all-pair reachability takes $mn=n^3$ time.

Reduced to $mn=n^2$ time in typical cases.

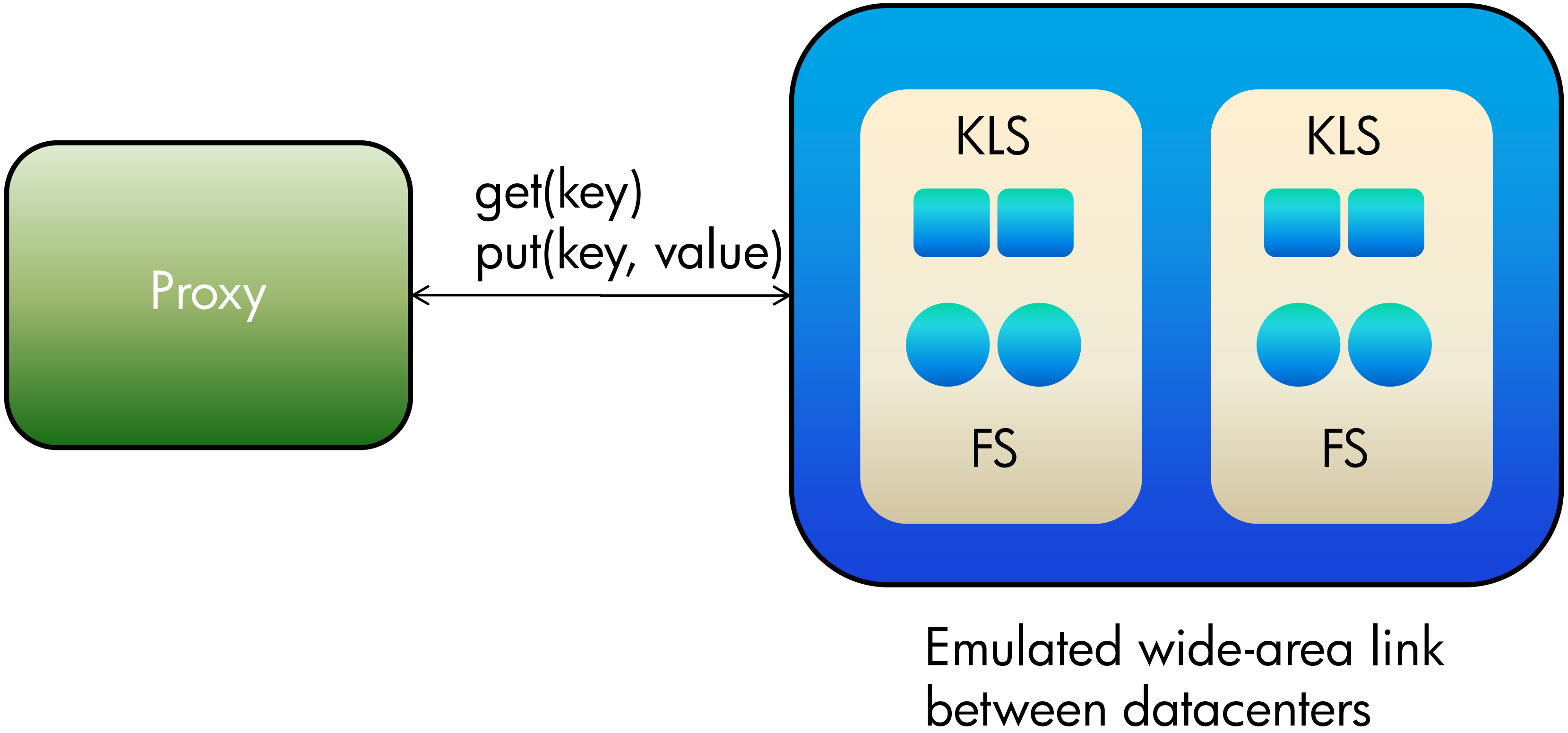
PAHOEHOE

[Anderson et al., DSN, 2010]

- A key-value store prototype
- Erasure-coded
- Multi-datacenter



EXPERIMENT SETUP

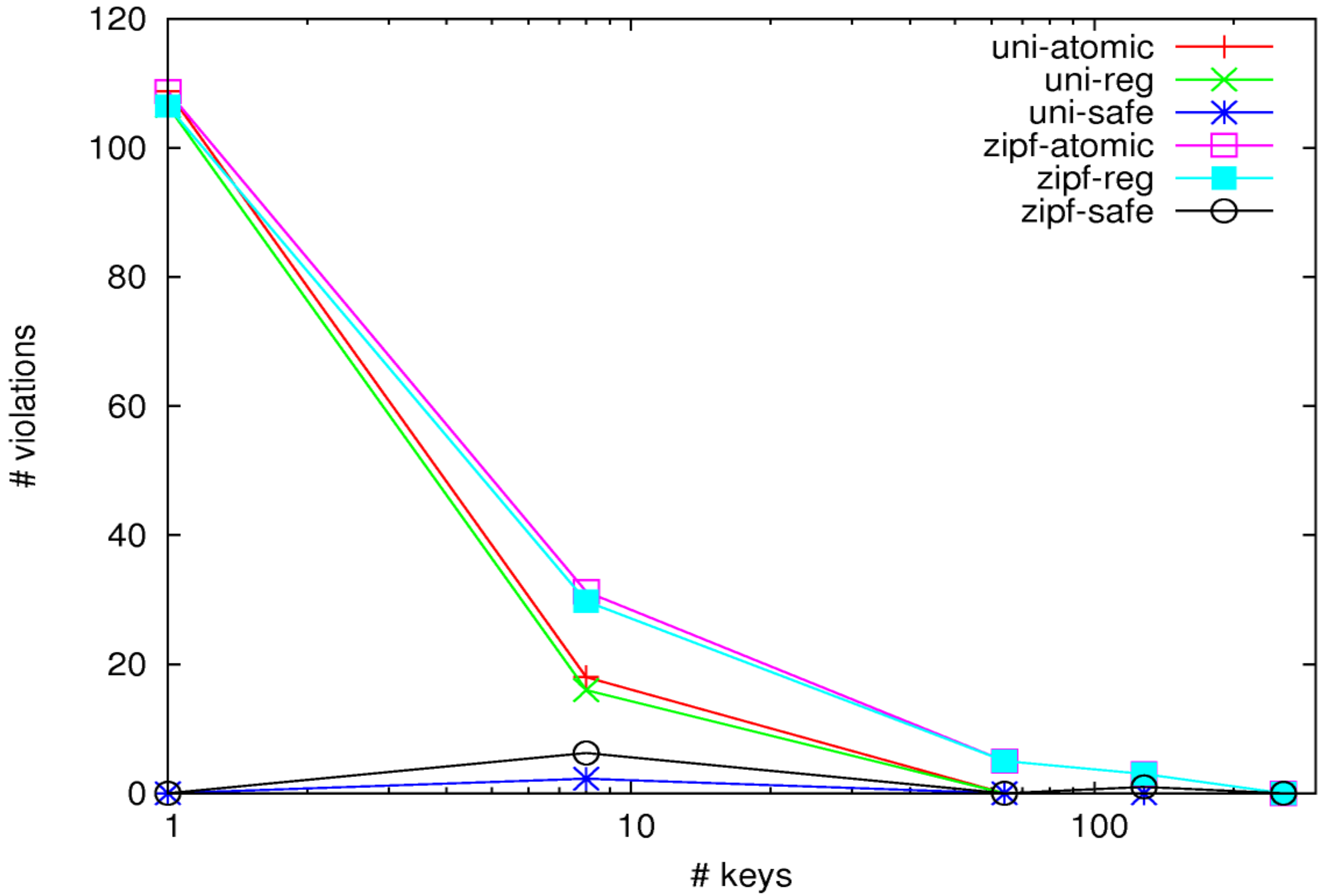


EXPERIMENT SETUP

- Proxy is <u>in</u> data center and shares NTP w/ servers
- 1000 operations
- Similar to YCSB microbenchmark
  - Larger object size: 128KB
  - 40% gets + 60% updates = 70% gets + 30% puts
- Varying
  - Number of keys
  - Number of processes
  - Distribution (uniform, Zipfian)

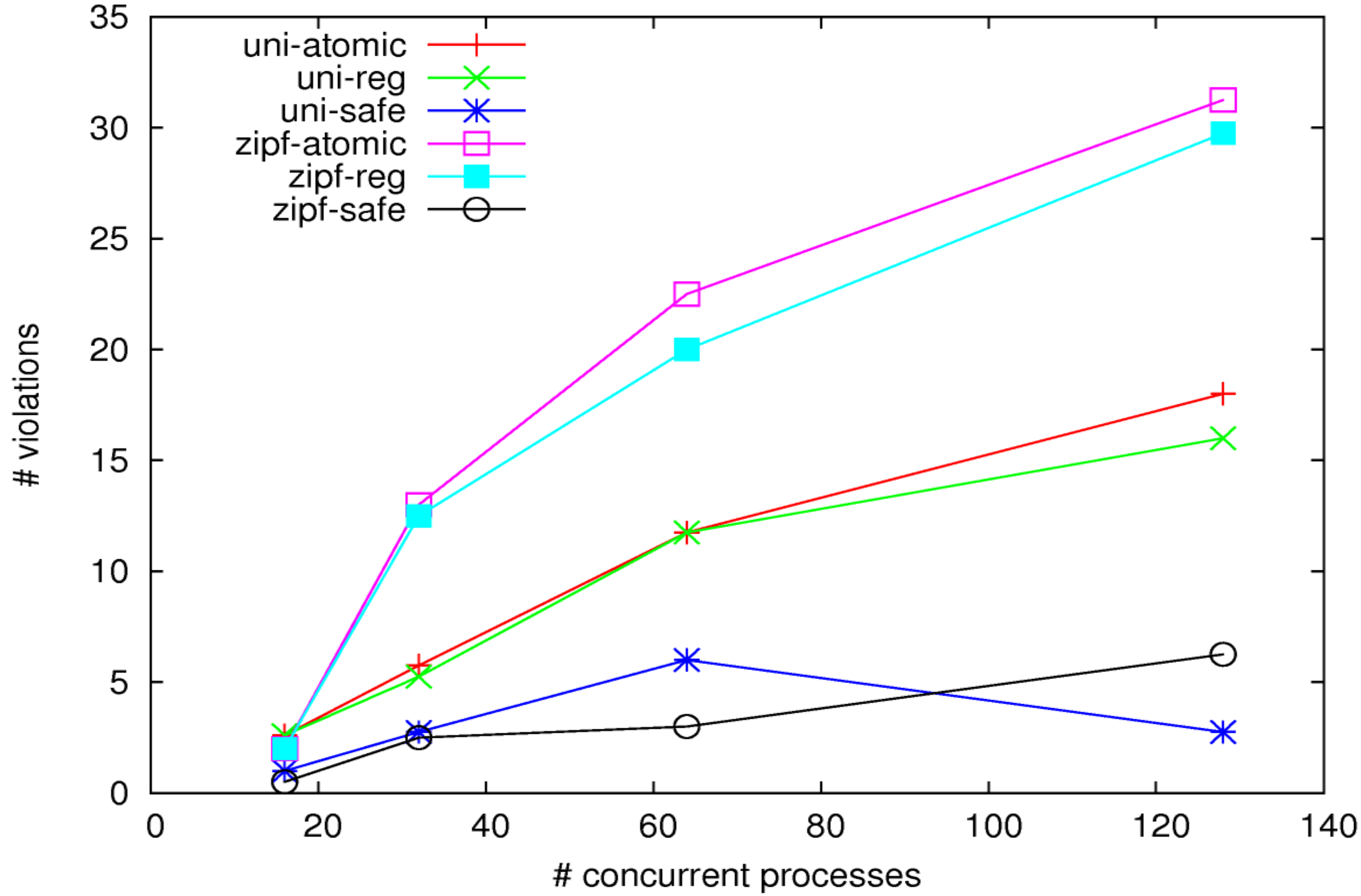
VIOLATIONS VS. KEYS

Concurrency = 128



VIOLATIONS VS. CONCURRENCY

Keys = 8

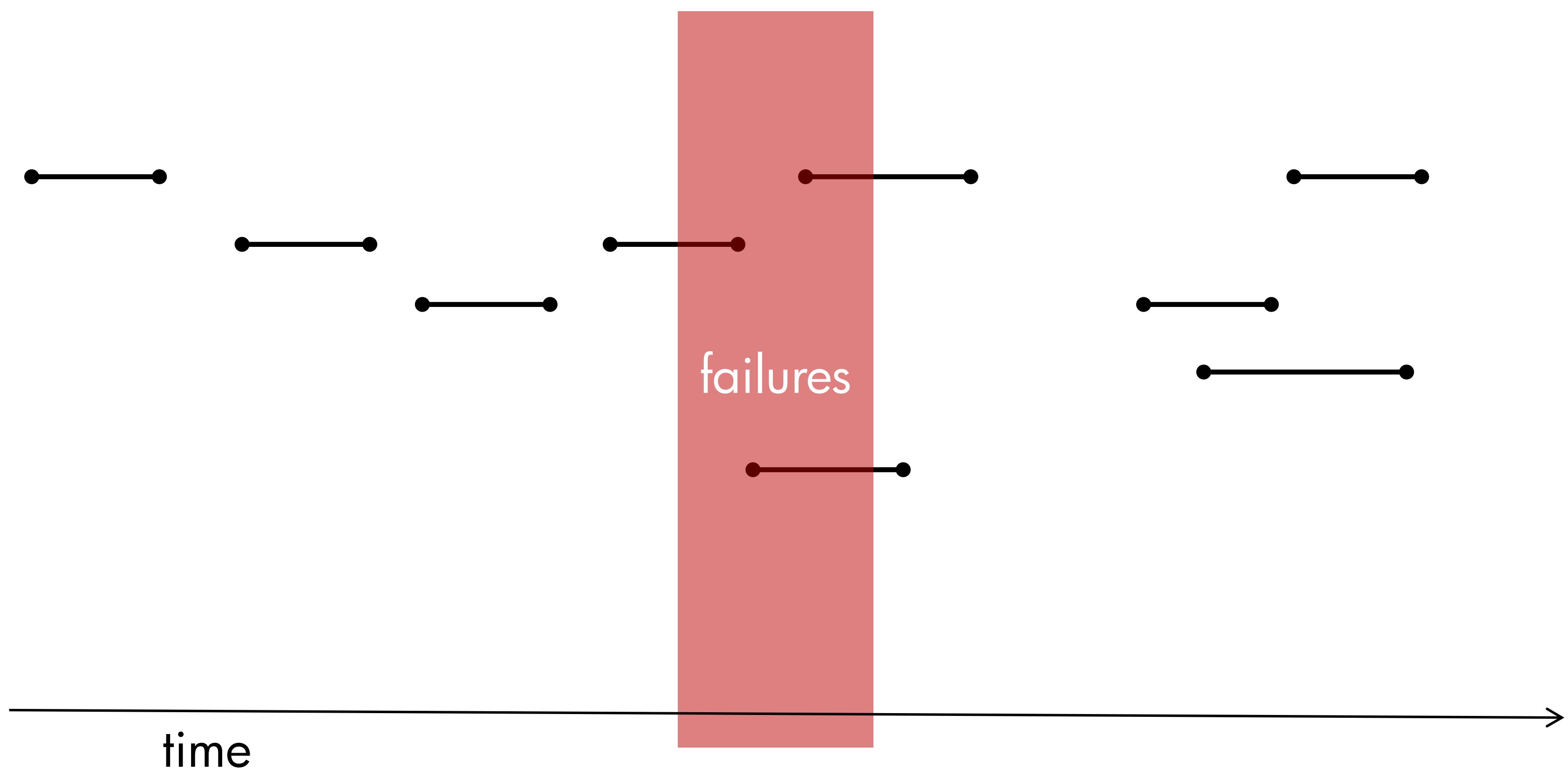


RELATED WORK

[Misra, TOPLAS, 1986]

- Misra's algorithm
- Reasons about values
- Only for atomicity
- Probably can be extended for safety and regularity
- Harder to quantify violation severity

ONLINE CONSISTENCY CHECKING



CONCLUSIONS

- Independent checking useful
- Algorithms for checking three semantics
- Eventually consistent may perform atomically
- Future work
  - Other semantics
  - Implement online checking
  - Monitor key-value stores