Pineapple: Unifying Multi-Paxos and Atomic Shared Registers

Tigran Bantikyan (Northwestern)*
Jonathan Zarnstorff (BoreDM)*
Te-Yen Chou (CMU)
Lewis Tseng (UMass Lowell)
Roberto Palmieri (Lehigh)

*Equal contribution

Storage

Replicated Storage

Replicated Storage

Replicated Storage ⇒

using consensus for strong consistency

Strong Consistency (or Linearizability)

High-level idea:

- All observed operations appear in the same total order

Advantage:

- Easier to develop applications with reduced complexity

Linearizable Replicated Storage

Google:

- GFS, Bigtable, Spanner

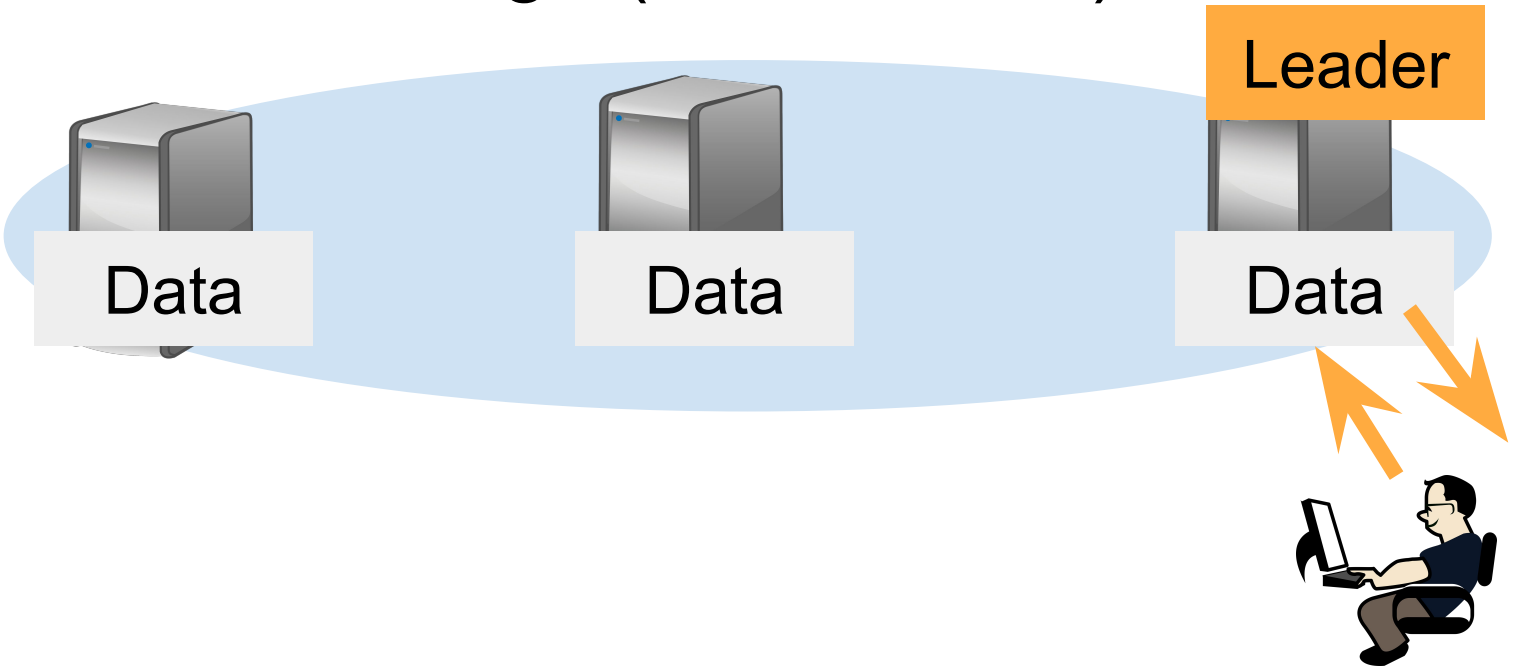
Microsoft:

- Azure storage

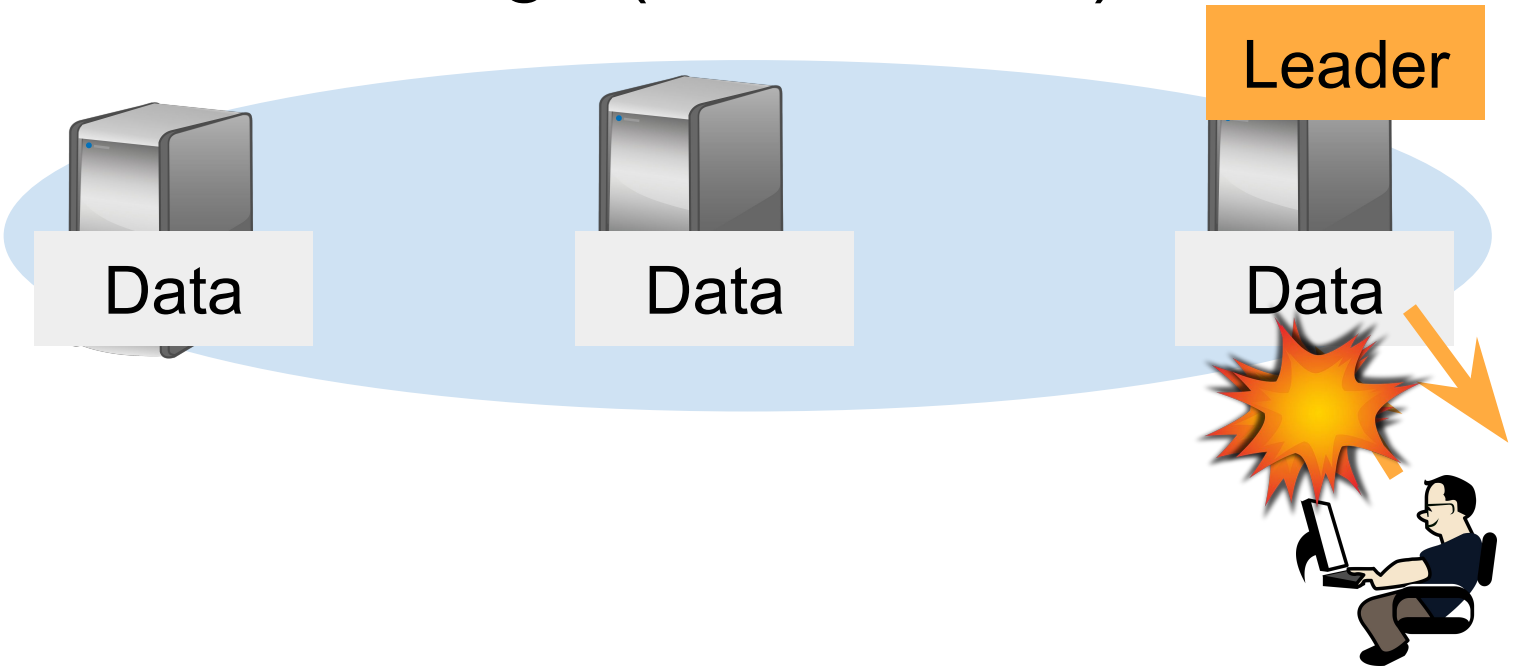
Open-source system:

- etcd, CockroachDB, ...etc

Leader-based Design (Paxos, Raft)



Leader-based Design (Paxos, Raft)



How to improve performance?

Existing Consensus-based Solutions

New consensus algorithms, with different models and assumptions

Consensus algorithms optimized for modern hardware

- SDN, RDMA, etc.

Better engineered consensus implementations

New algorithmic components, compatible with existing consensus algorithms

Many modern replicated key-value storages

support three types of operations:

(i) read from a single key,

(ii) write to a single key, and

(iii) perform a one-shot transaction

Blind Writes?

Modern systems support blind writes:

- Cassandra, Spanner, HBase, Redis, etcd, etc.

Versions object storage can be implemented using blind writes

- Microsoft Giza, Amazon S3, etc.

Data structures with associative and commutative properties

- Histogram, Hyperloglog, etc. [Cao et al. SIGMOD20]

One-Shot Transactions?

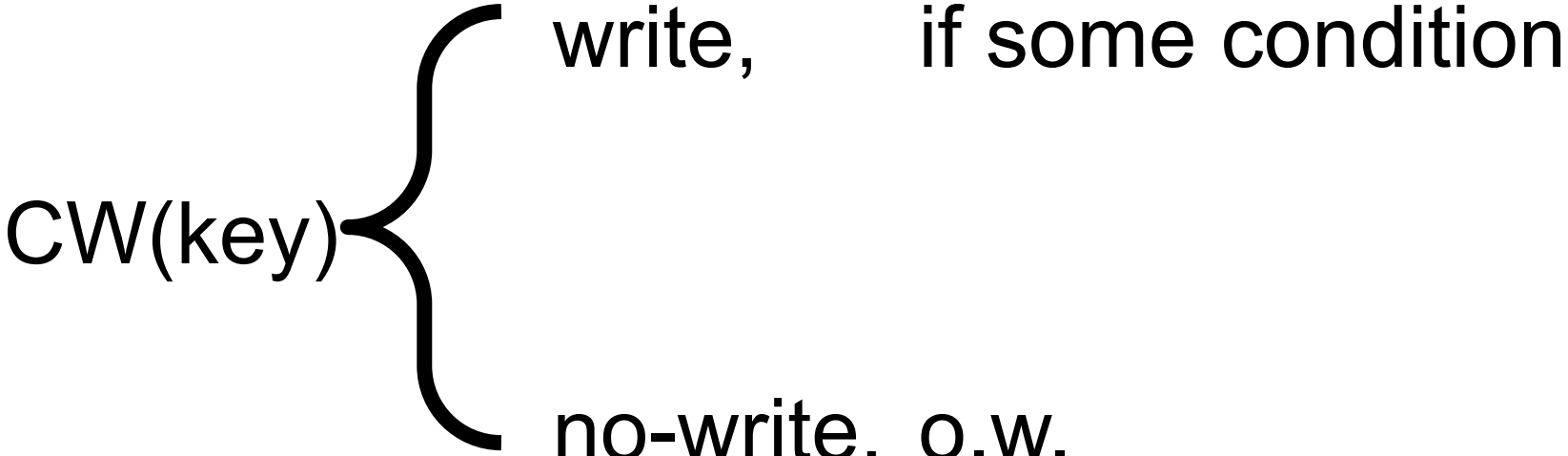
access multiple key-value pairs

provide stronger synchronization (e.g., read-modify-write, fetch-and-add, compare-and-swap)

or both

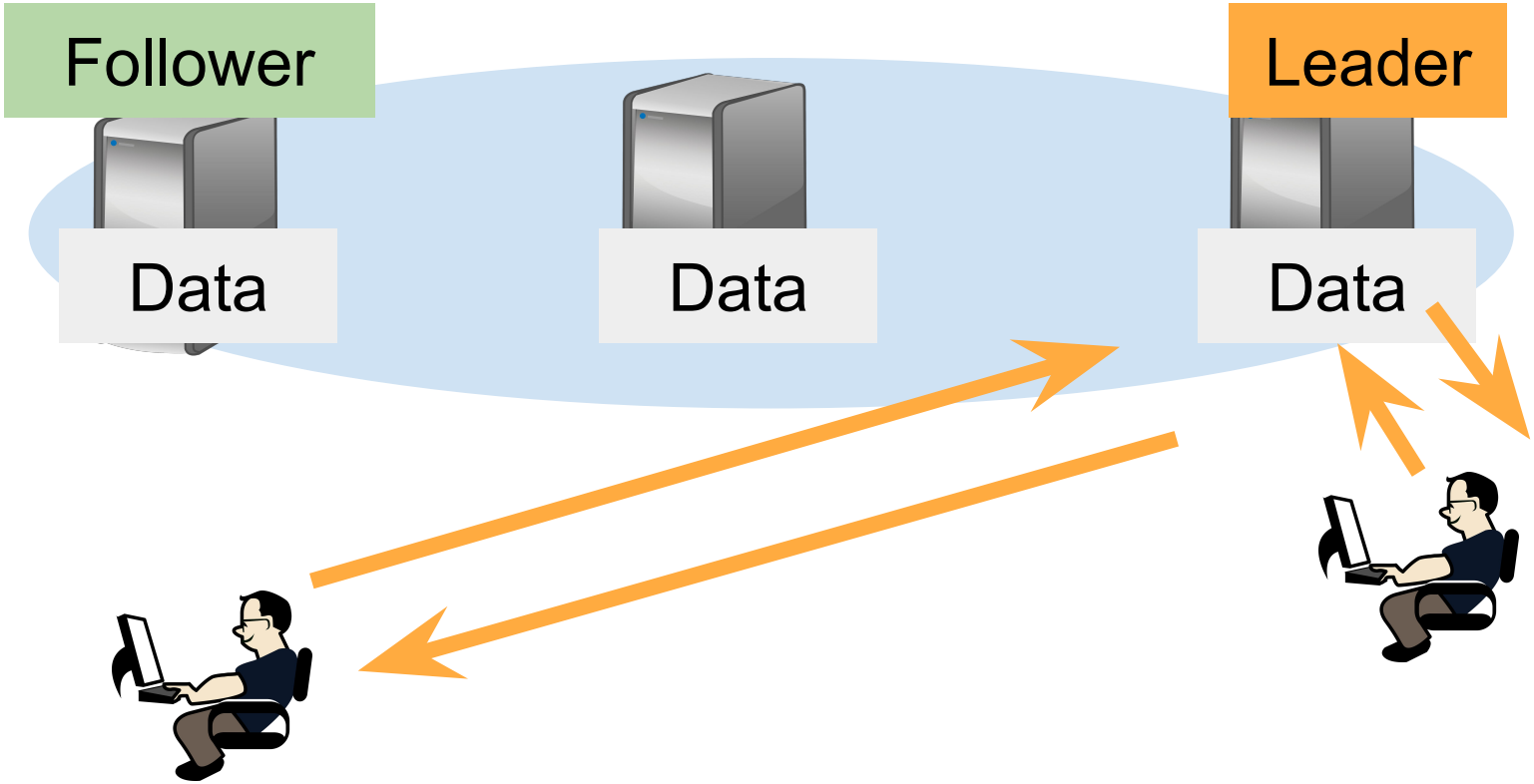
- represented as “stored procedures”
  [Kallman et al. VLDB08], [Mu et al. OSDI14]

Conditional Write

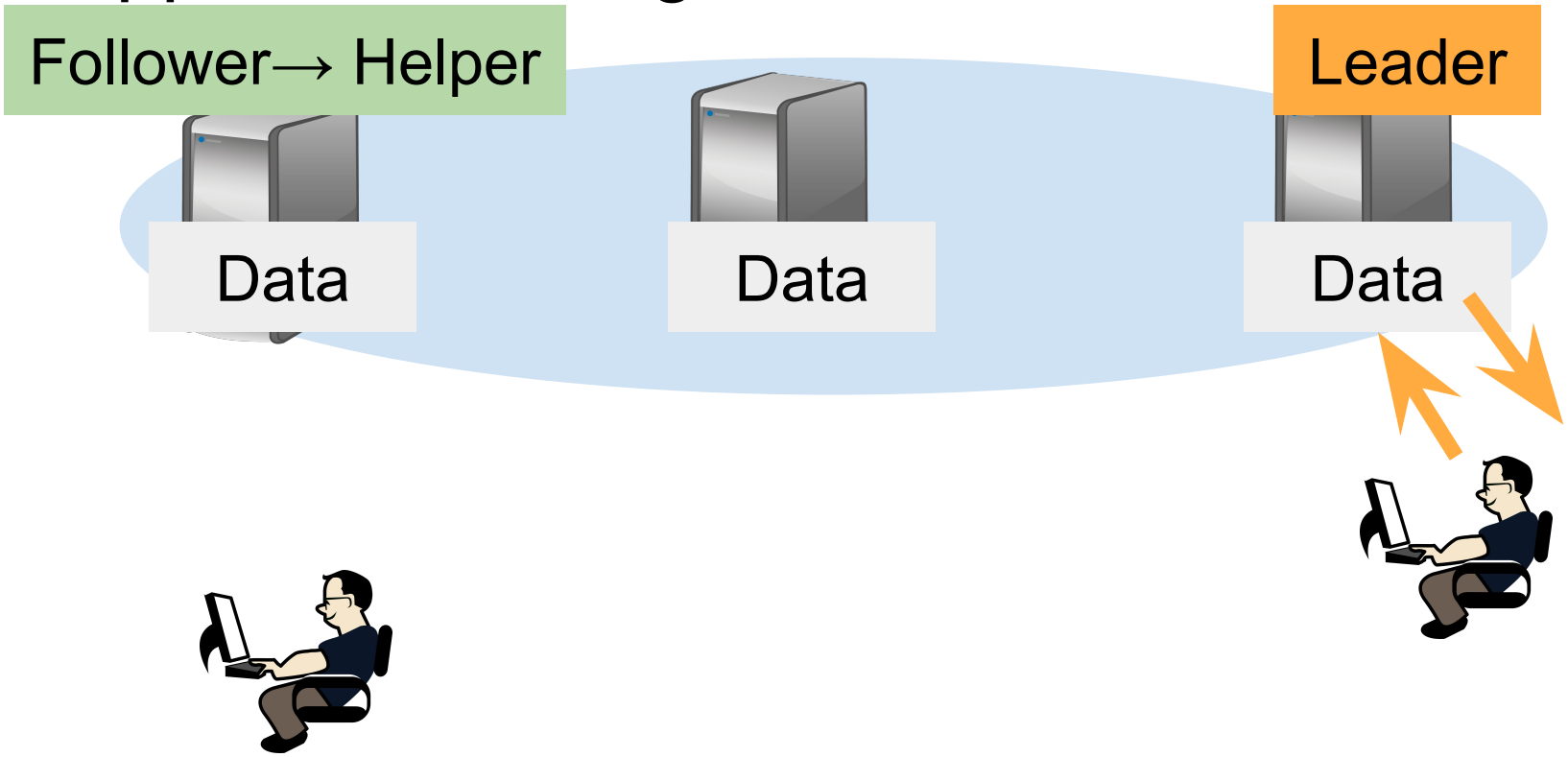


How to improve Multi-Paxos, by focusing on reads, writes, one-shot transactions?

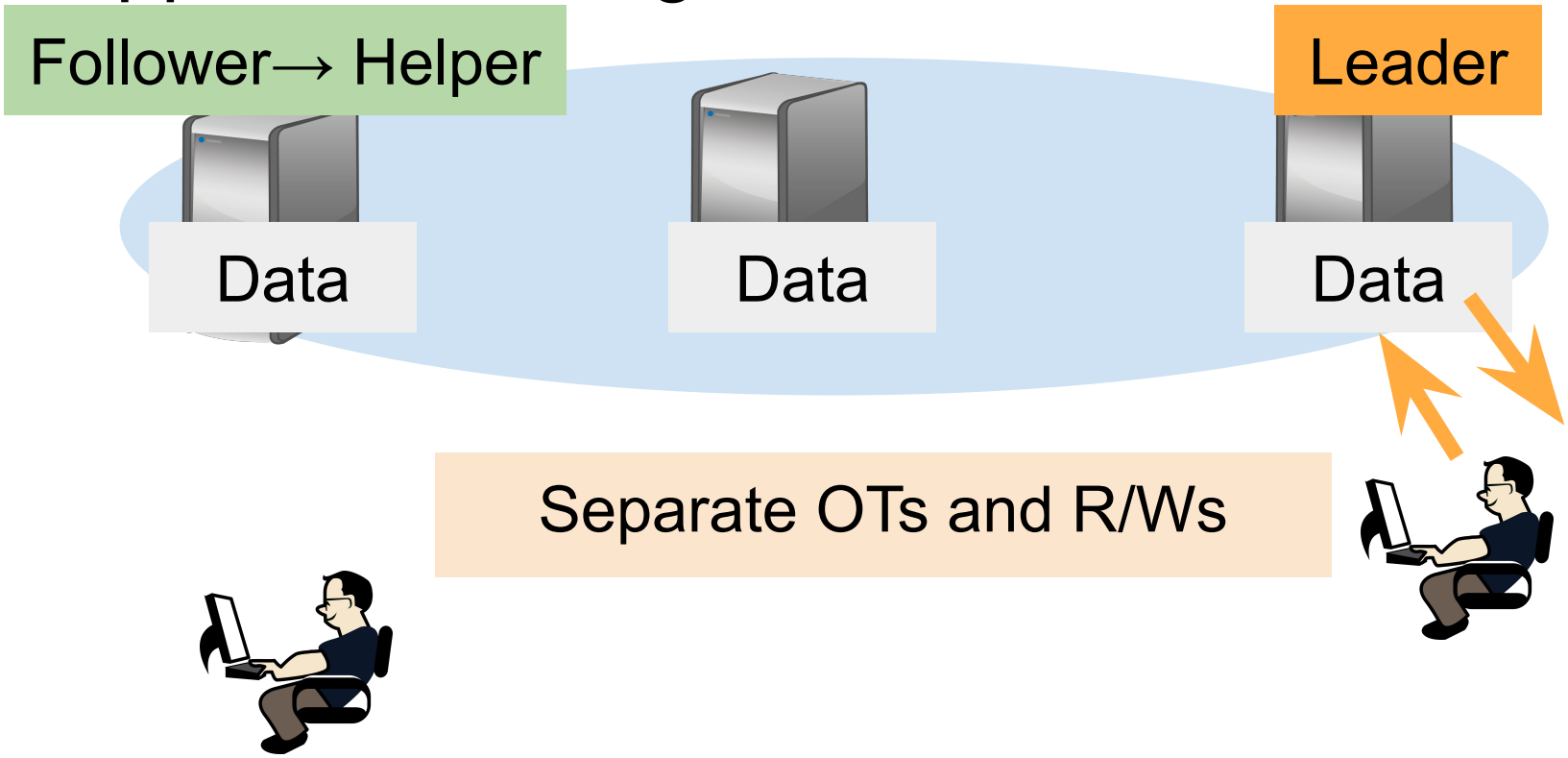
Multi-Paxos



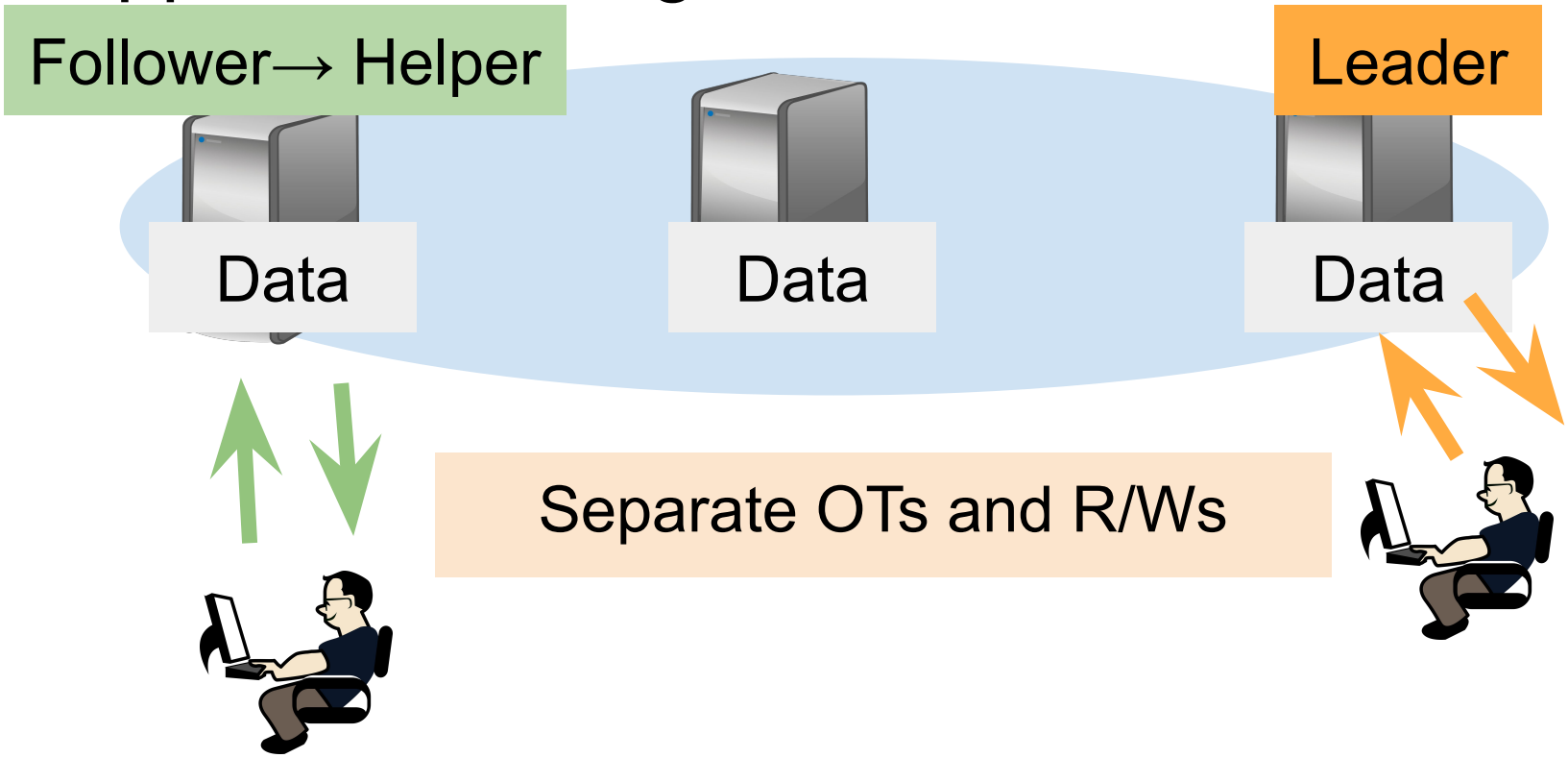
Pineapple: off-loading reads/writes



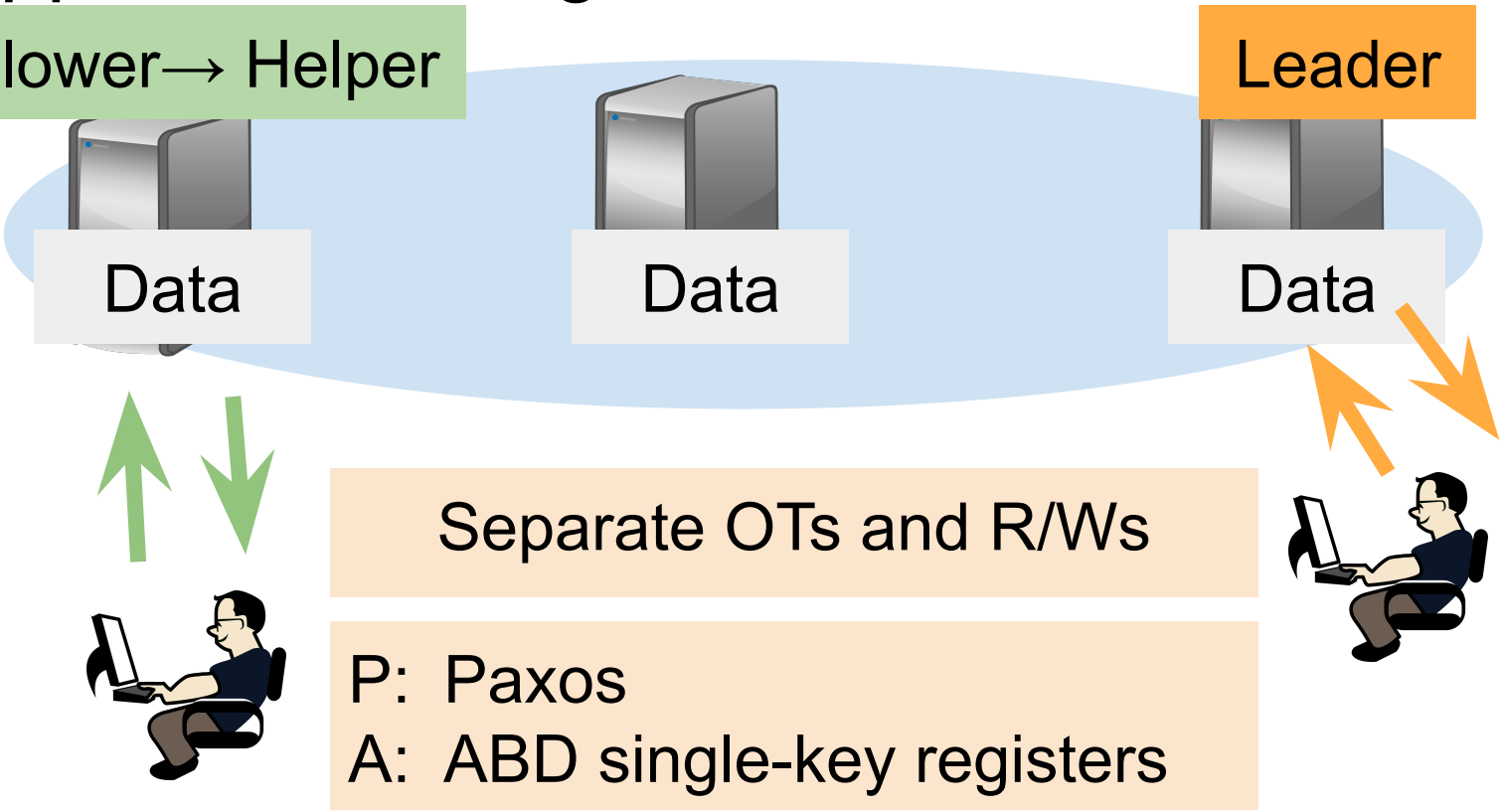
Pineapple: off-loading reads/writes



Pineapple: off-loading reads/writes



Pineapple: off-loading reads/writes



Pineapple: Highlights

Offloading single-key reads and writes to follower nodes

- lower tail-latency and higher throughput

1-RTT single-key reads

- N = 3: always
- N > 3: when no conflict

Non-blocking execution

- Any operation can be executed, after communicating with leader or quorum

Pineapple: Highlights

Offloading single-key reads and writes to follower nodes

- lower tail-latency and higher throughput

1-RTT single-key reads

- N = 3: always
- N > 3: when no conflict

Non-blocking execution

- Any operation can be executed, after communicating with leader or quorum

  (Unlike EPaxos [SODP13], Gryff [NSDI20], PQR [HotStorage19])

PSTAMP: Pineapple Timestamp

<tag, slot>

tag: (logical ts, writer-ID)

slot: (slot index)*

Pineapple in 3 Lines

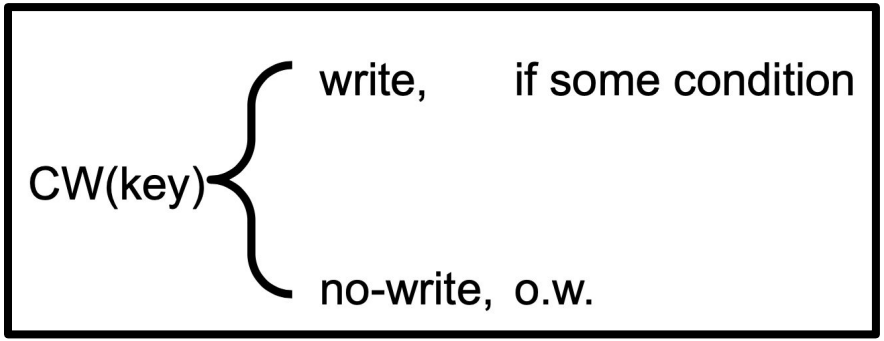
PSTAMP: <tag, slot>

ABD uses tag to read and write [Attiya et al. JACM95]

Multi-Paxos uses slot + one more RTT

(than normal Multi-Paxos)

Pineapple in 3 Lines



PSTAMP: <tag, slot>

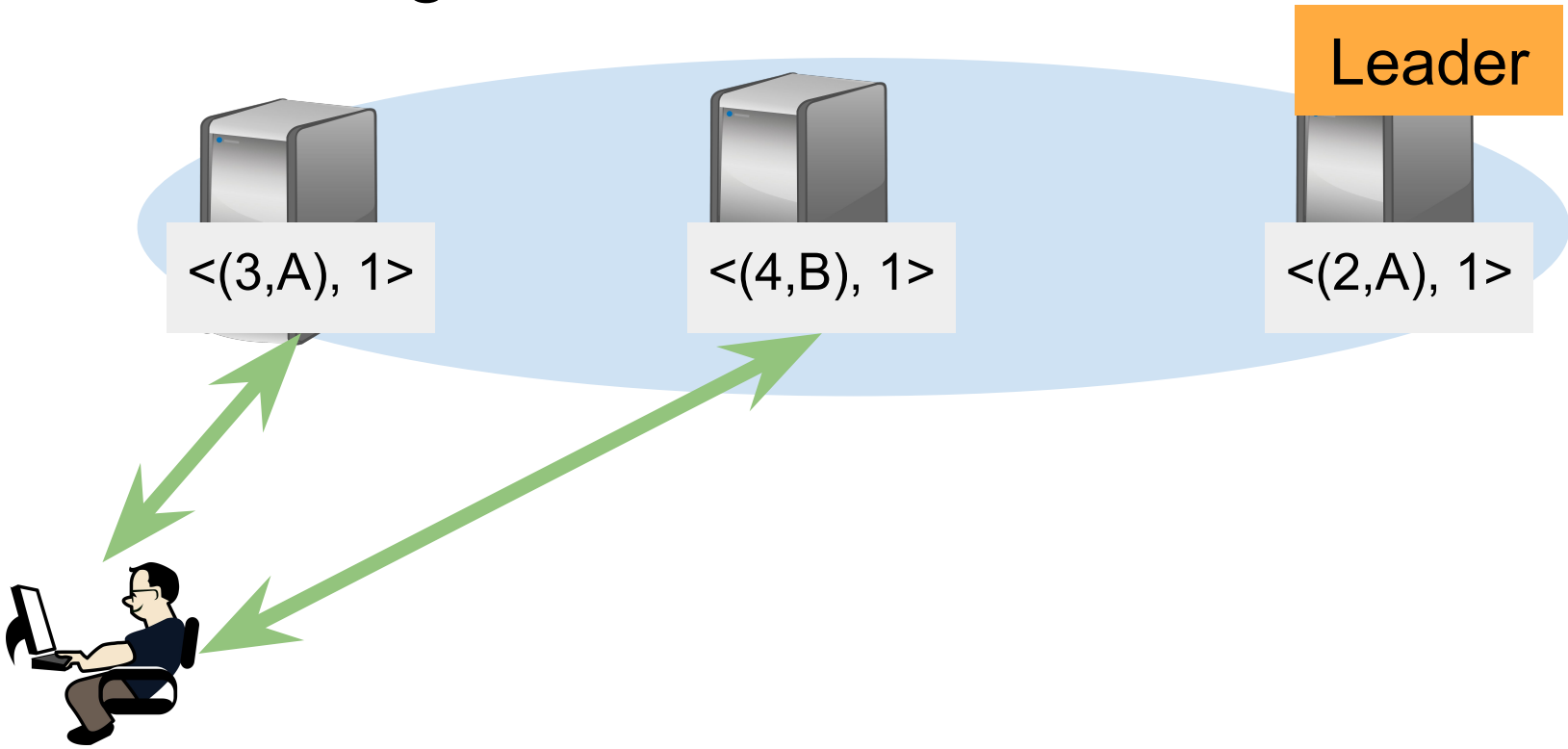
ABD uses tag to read and write [Attiya et al. JACM95]

Multi-Paxos uses slot + one more RTT

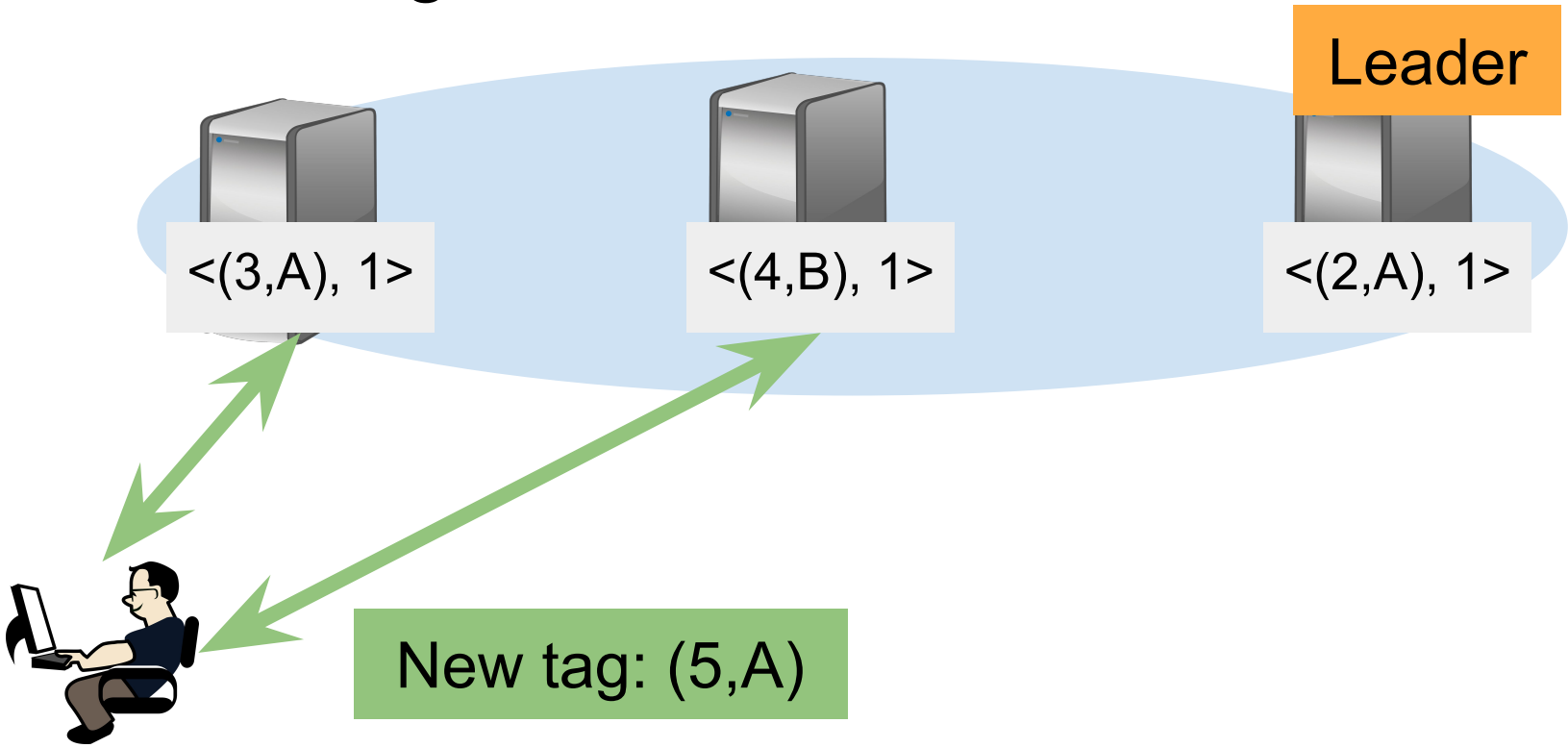
(than normal Multi-Paxos)

- fetch most recent “condition” (for one-shot transactions)

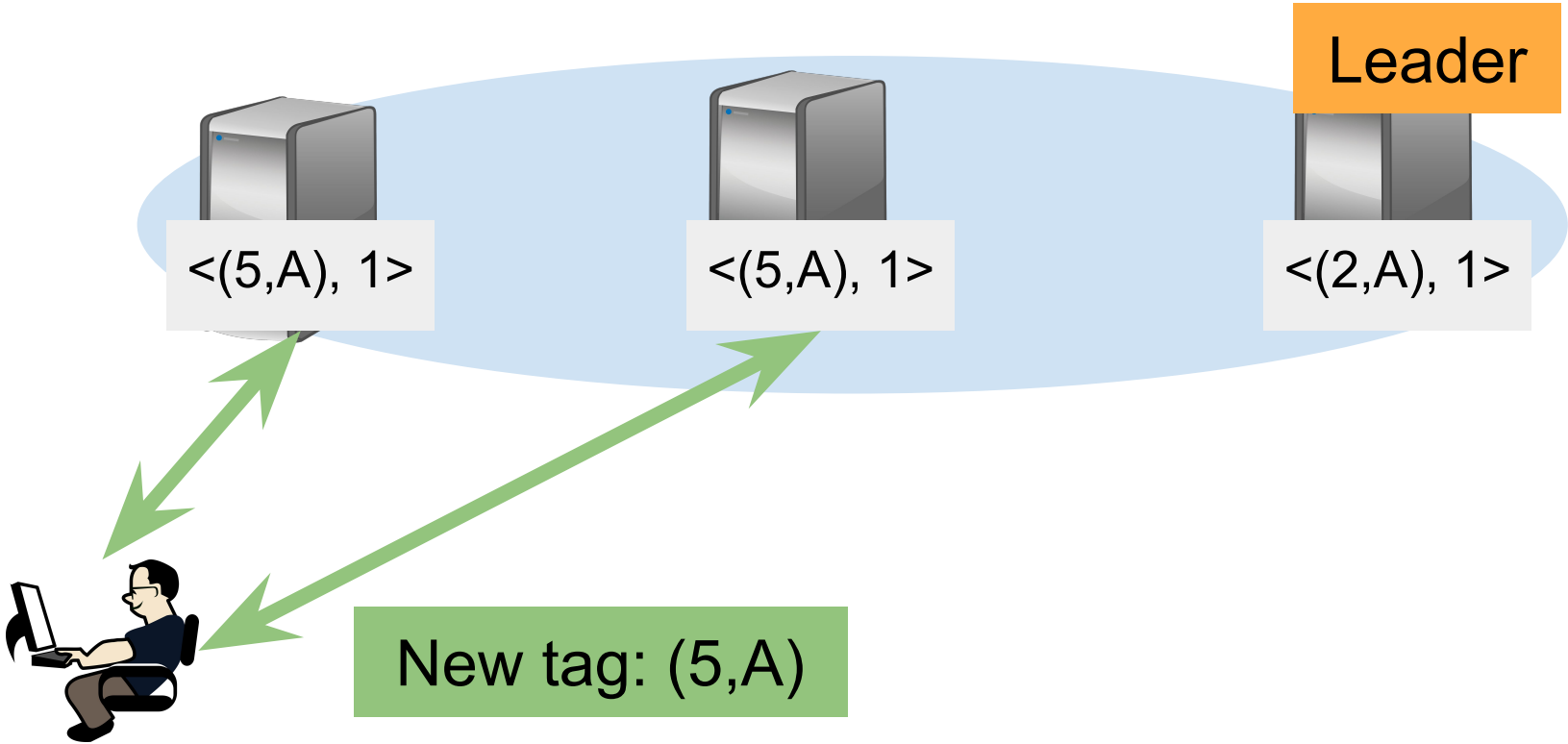
Write - GetTag



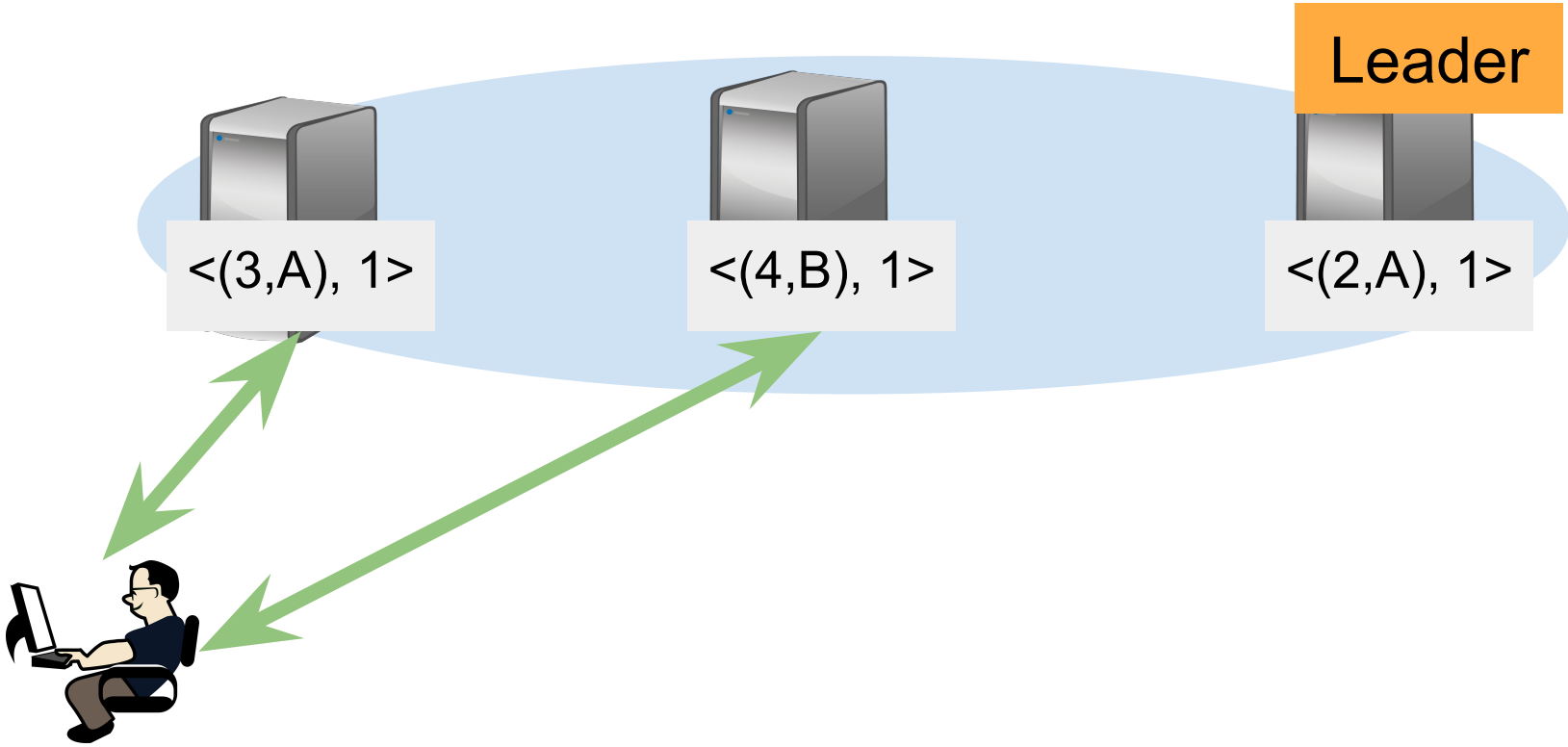
Write - GetTag



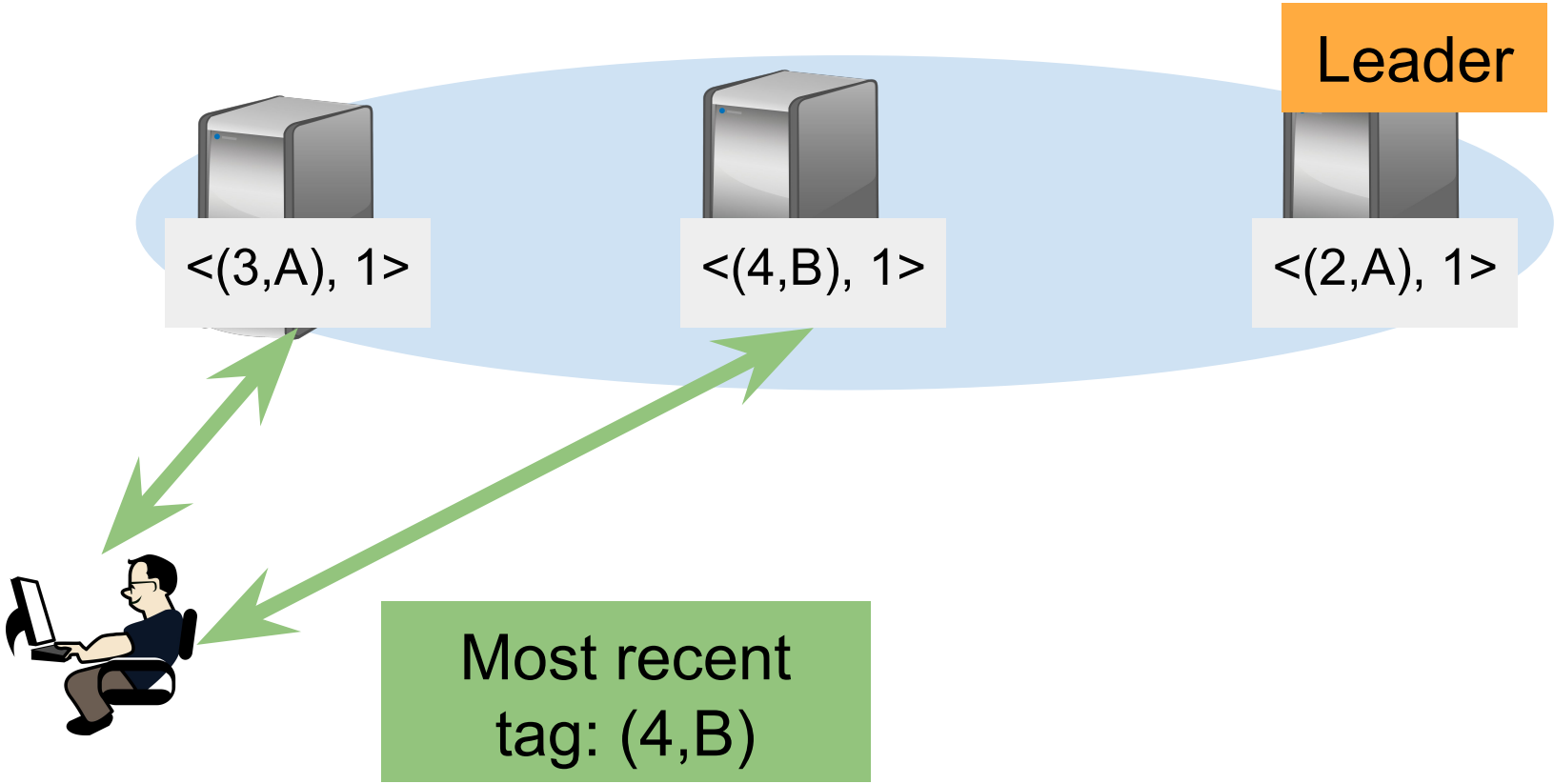
Write - Put



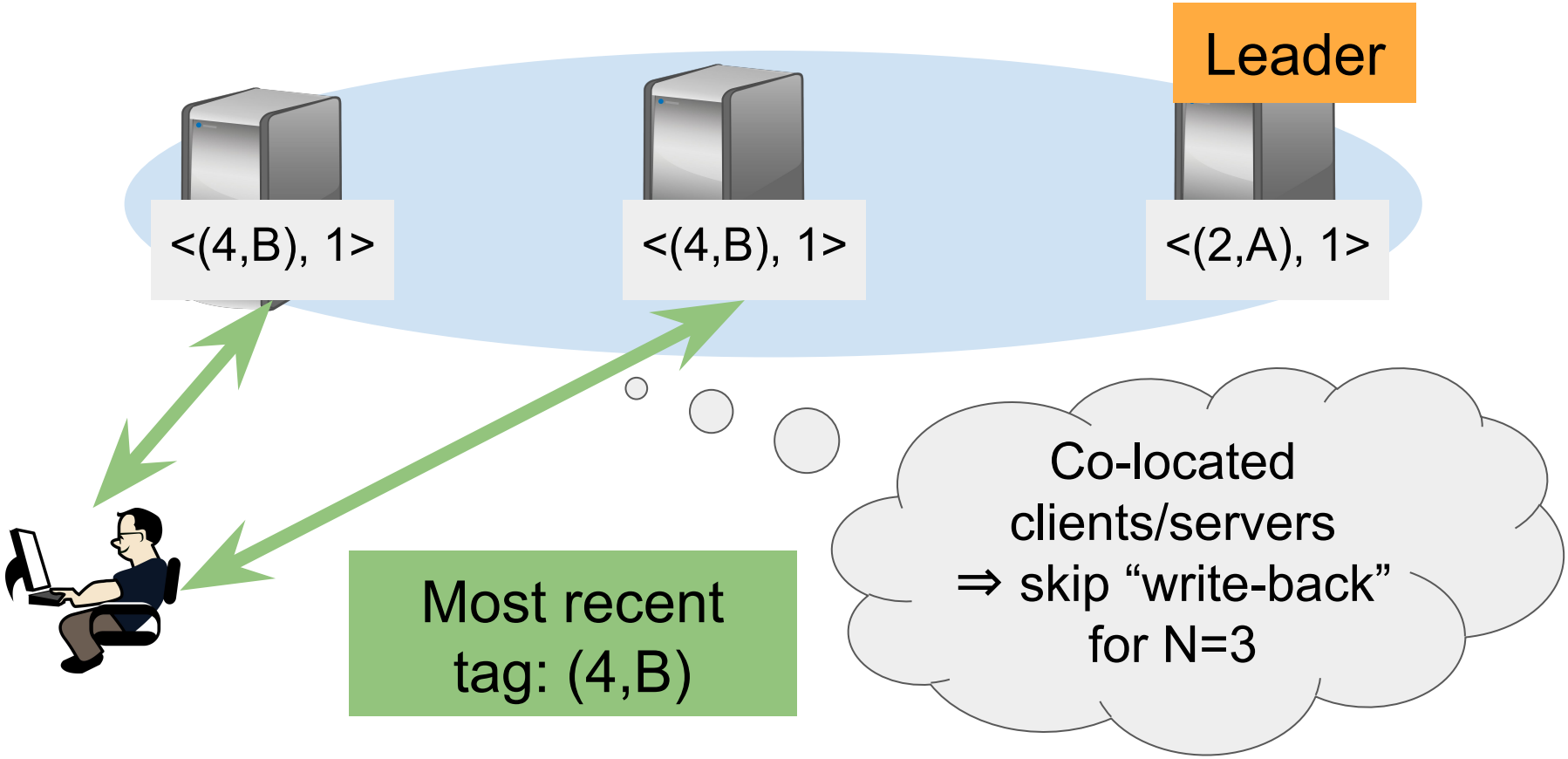
Read - GetValue



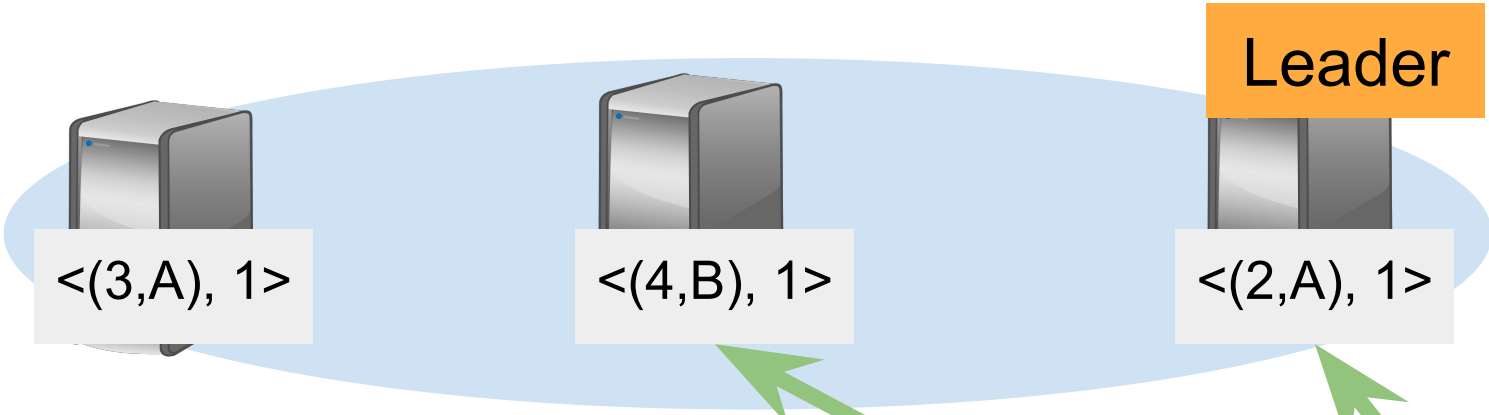
Read - GetValue



Read - Put


Leader
<(4,B), 1>
<(4,B), 1>
<(2,A), 1>
Most recent tag: (4,B)
Co-located clients/servers ⇒ skip “write-back” for N=3

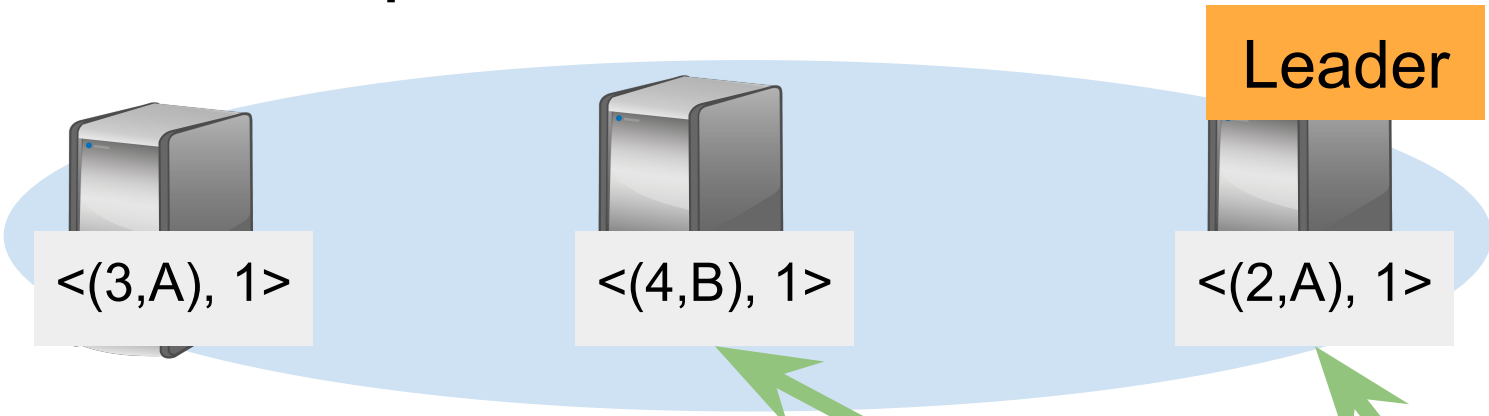
OT - GetValue



CW(key) { write, if some condition
no-write, o.w.

Most recent tag: (4,B)

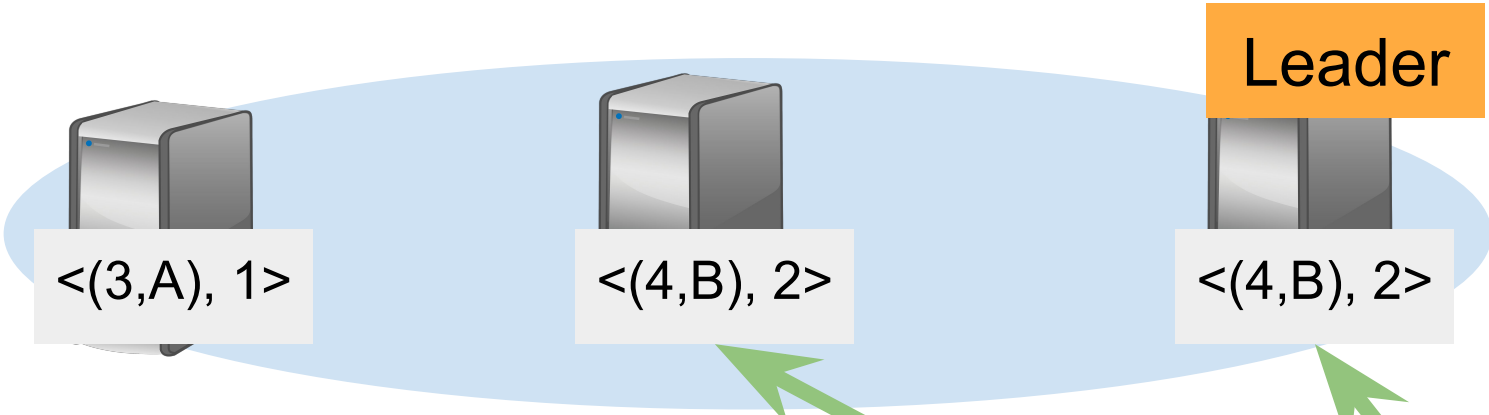
OT - Local Computation



CW(key) { write, if some condition
no-write, o.w.

New PSTAMP:
<(4,B), 2>

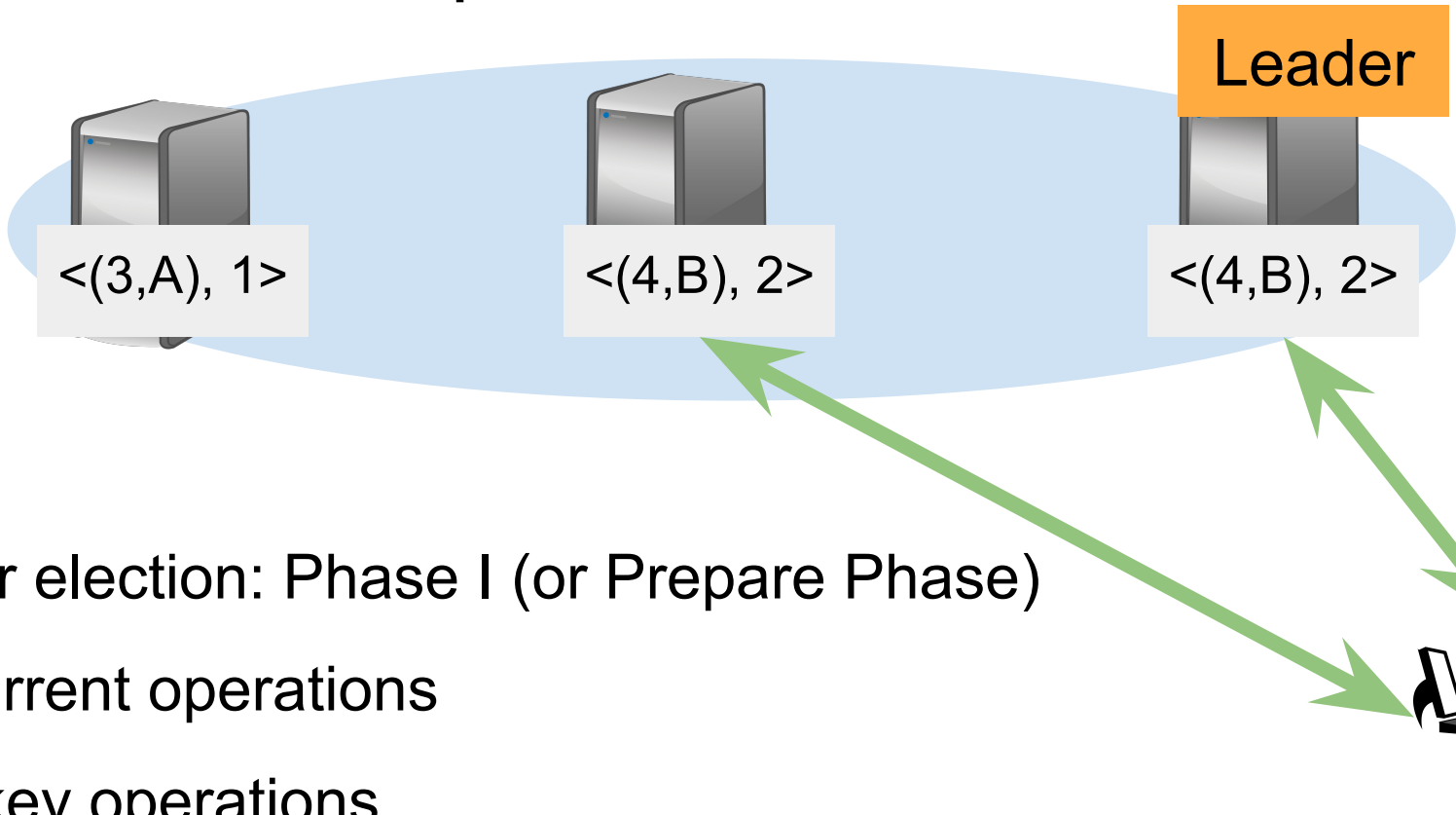
OT - Put



CW(key) { write, if some condition
no-write, o.w.

New PSTAMP:
<(4,B), 2>

Details: Check Paper



Leader election: Phase I (or Prepare Phase)

Concurrent operations

Multi-key operations

Evaluation

Maximum attainable throughput

Read/Write/RMW latency

Pineapple in production system (etcd)

State-of-the-art consensus protocols:

- MP, MP(L), Paxos Quorum Read [HotStorage19]
- EPaxos [SOSP13], Gryff [NSDI20] – EPaxos + ABD

Testbed: CloudLab

c6525-25g – for both servers and clients

- AMD 7302P at 3.00GHz, 16 cores, 128 GM RAM

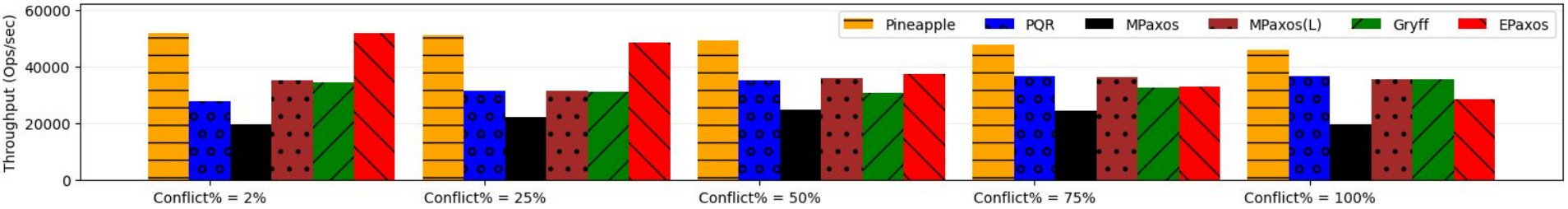
LAN:

- ~0.15ms RTT, 10 Gbps

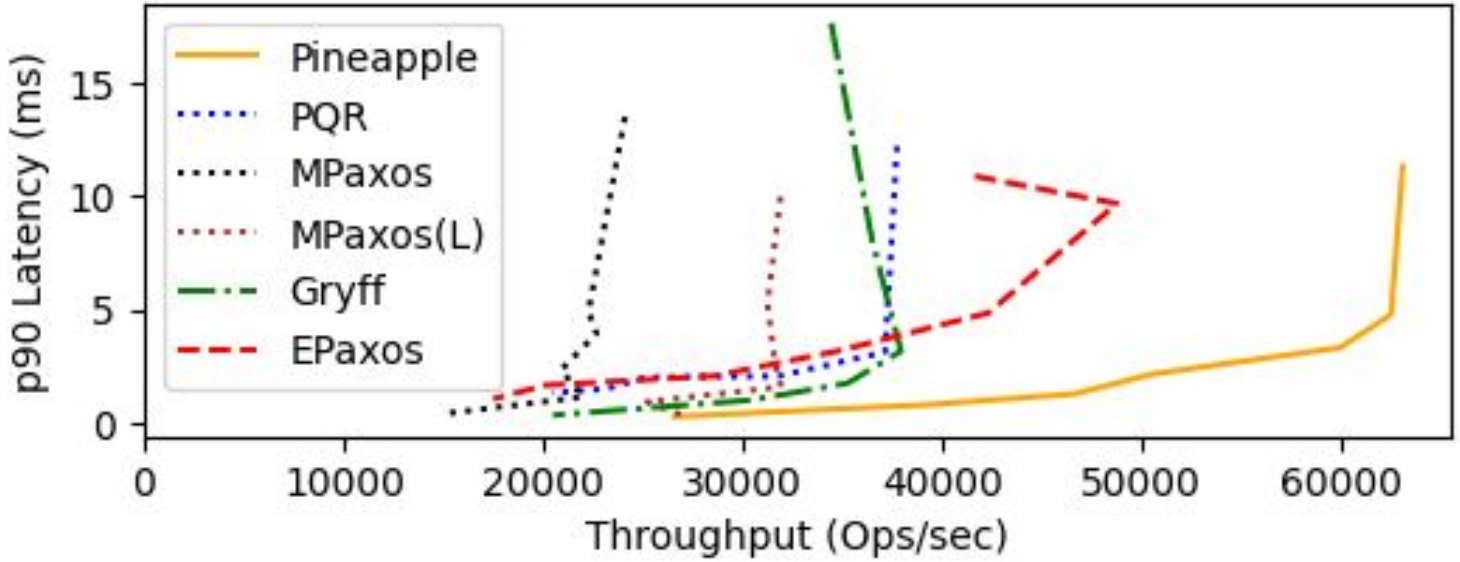
WAN:

- simulated AWS latency profile from [NSDI20]

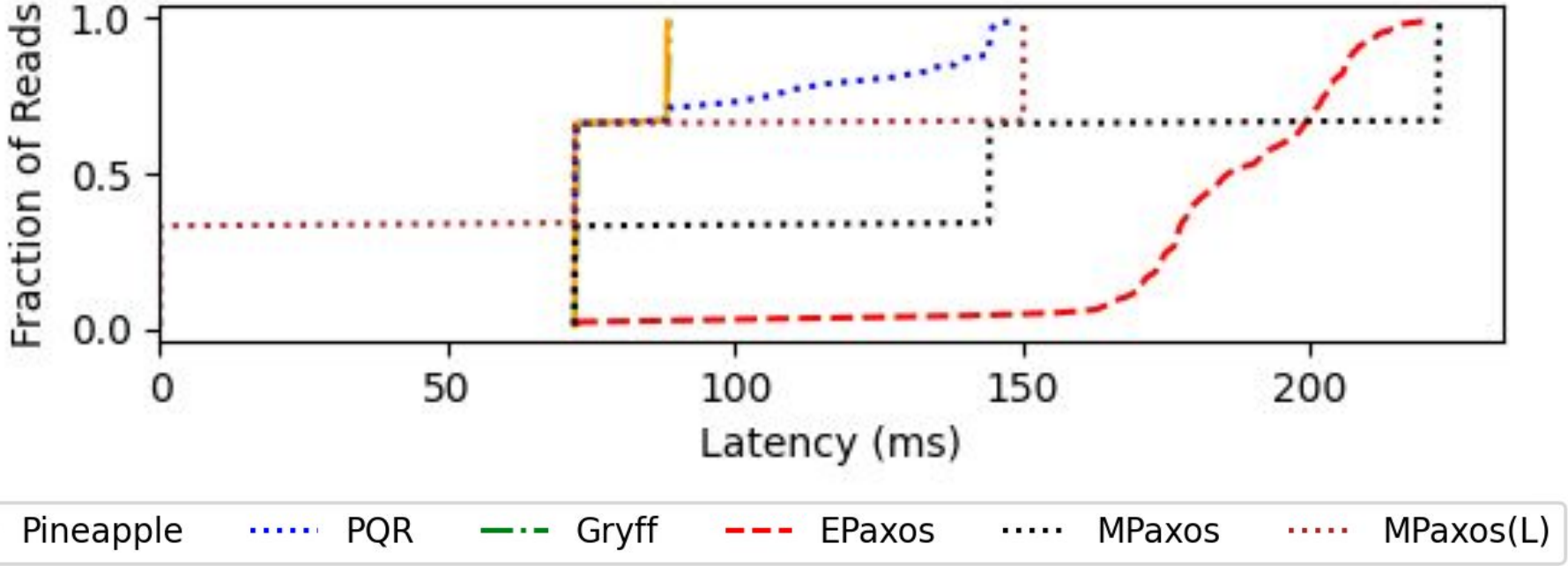
LAN: N=5, 20% RMW, 40% reads/writes



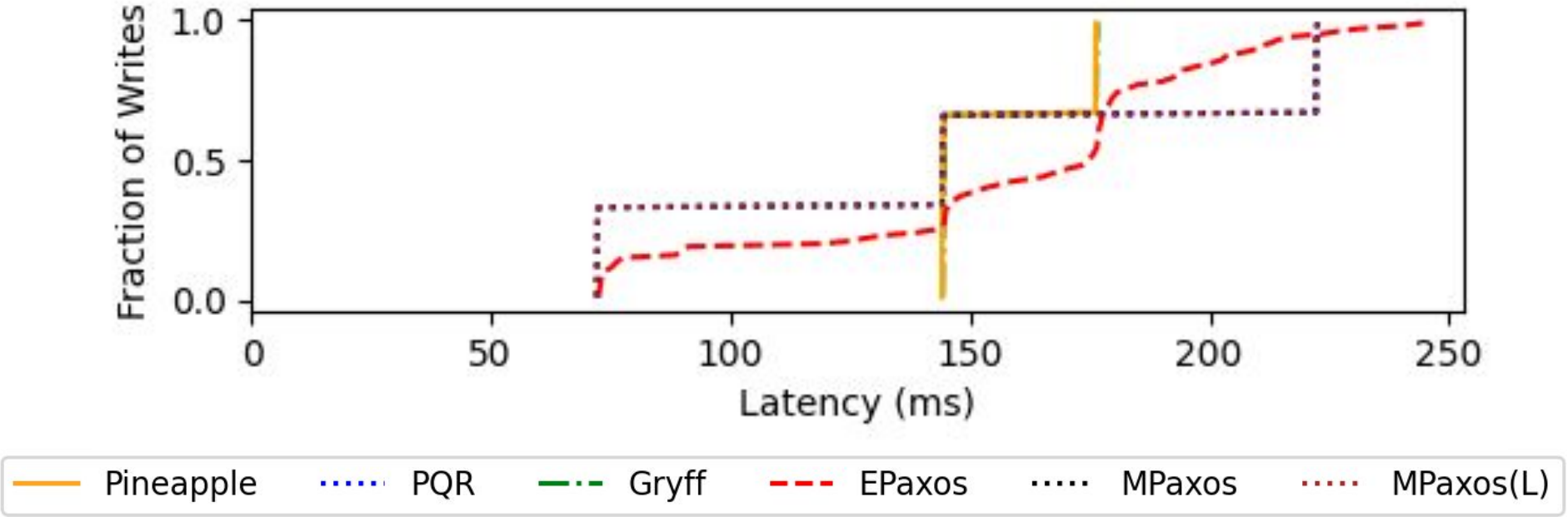
LAN: N=5, 49.5% writes/reads, 1% RMW, 25% conflict rate (~YCSB-A)



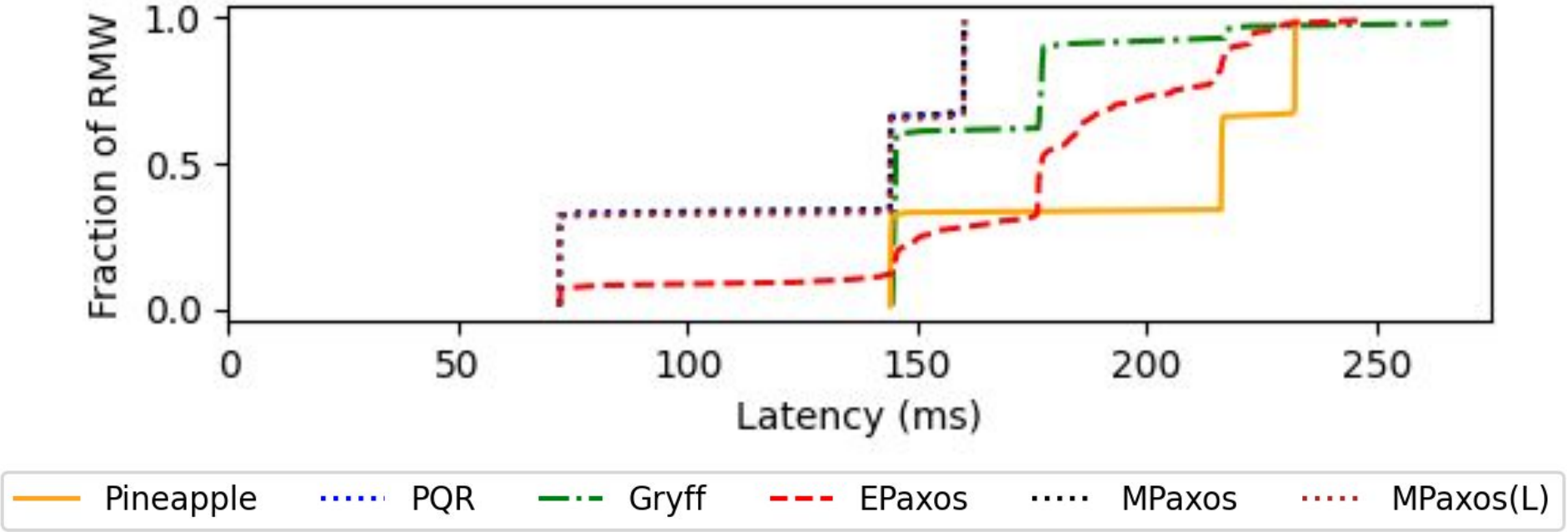
WAN: N=3, 94.5% reads, 4.5% writes, 1% RMWs, 25% conflict rate (~YCSB-B)



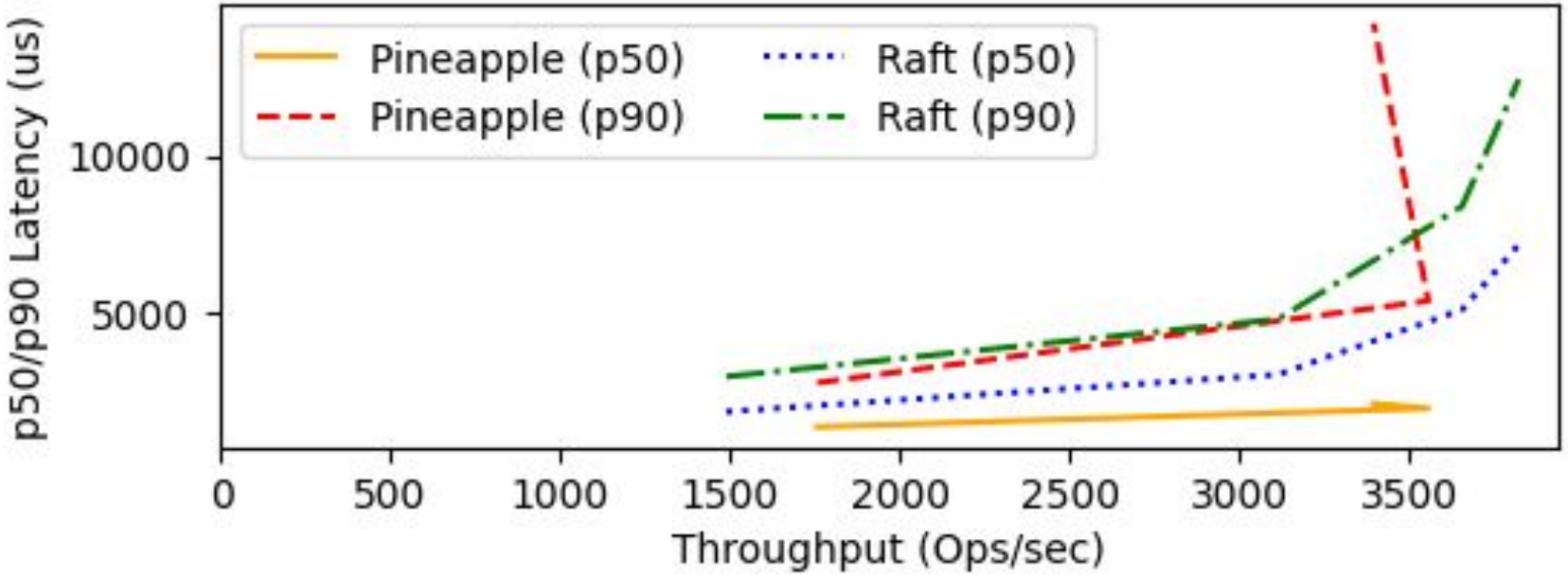
WAN: N=3, 94.5% reads, 4.5% writes, 1% RMWs, 25% conflict rate (~YCSB-B)



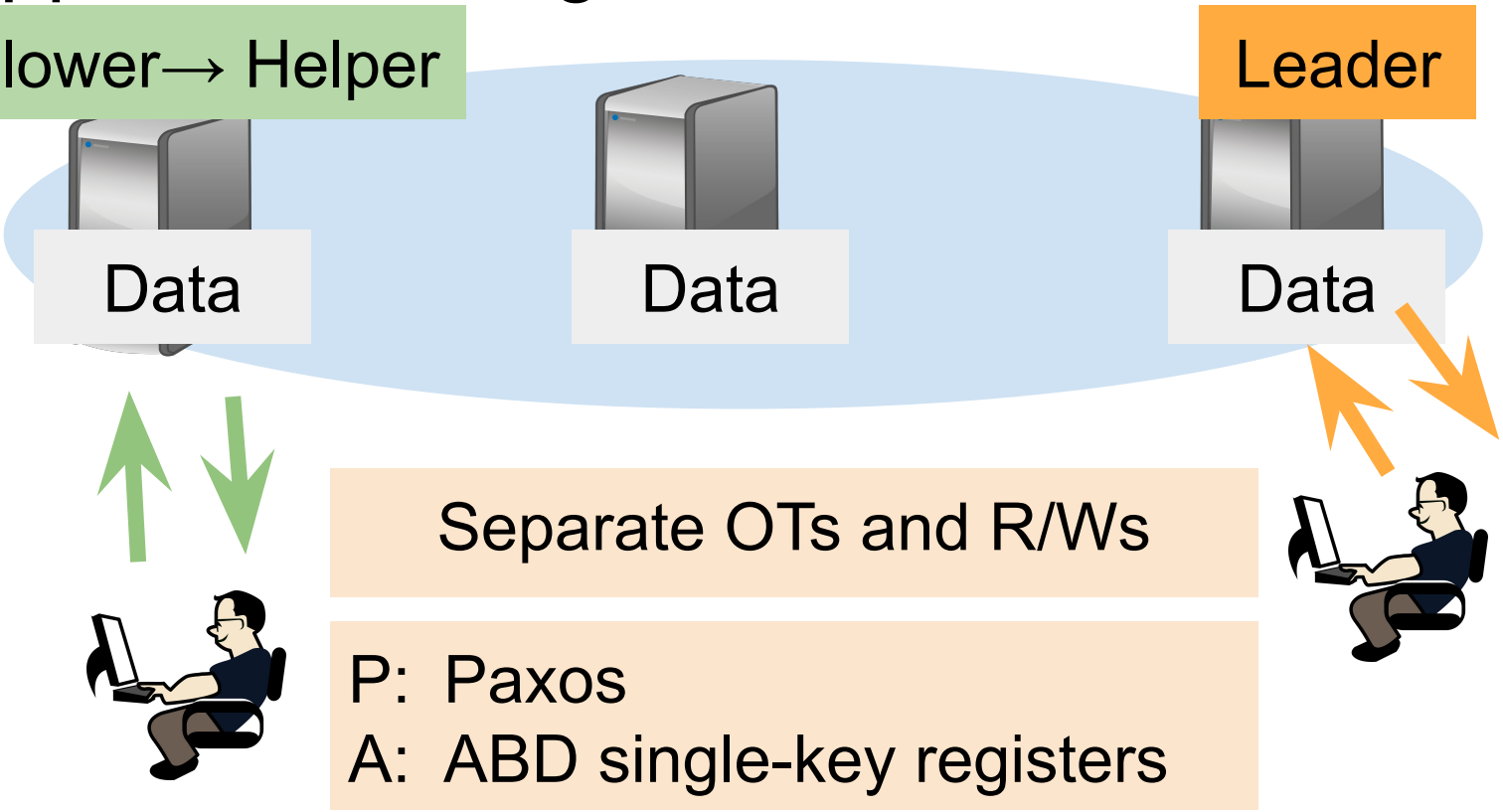
WAN: N=3, 94.5% reads, 4.5% writes, 1% RMWs, 25% conflict rate (~YCSB-B)



LAN: N=5, 50% reads, 25% writes/RMWs, Zipfian workloads in etcd



Pineapple: off-loading reads/writes



Thanks!
Questions?

lewistseng@acm.org

Image Attribution

etcd: https://github.com/etcd-io/etcd/blob/d375b67a5010fde147bda335f36d5b9a5a7db4f4/logos/etcd-horizontal-color.png

CockroachDB: https://en.wikipedia.org/wiki/CockroachDB