Strong and Efficient Consistency with Consistency-Aware Durability

Aishwarya Ganesan, Ramnatthan Alagappan,

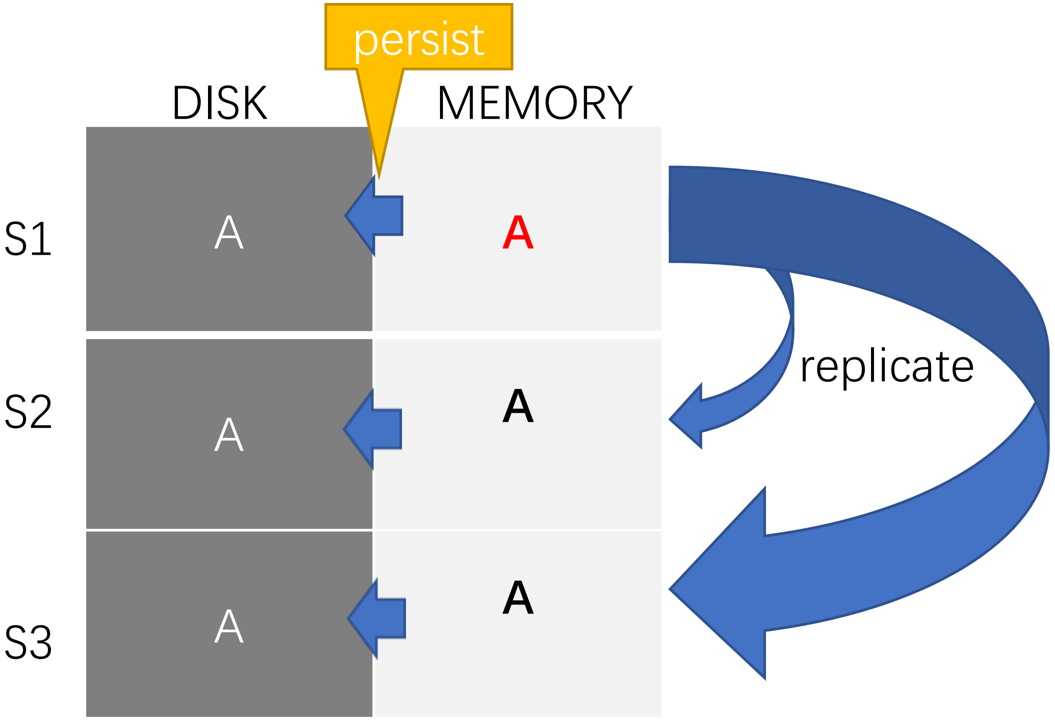
Andrea C. Arpaci-Dusseau, and Remzi H. Arpaci-Dusseau

University of Wisconsin – Madison

- Background and motivation
- Consistency-aware durability and Cross-client Monotonic read
- ORCA design
- Evaluation

Background and motivation

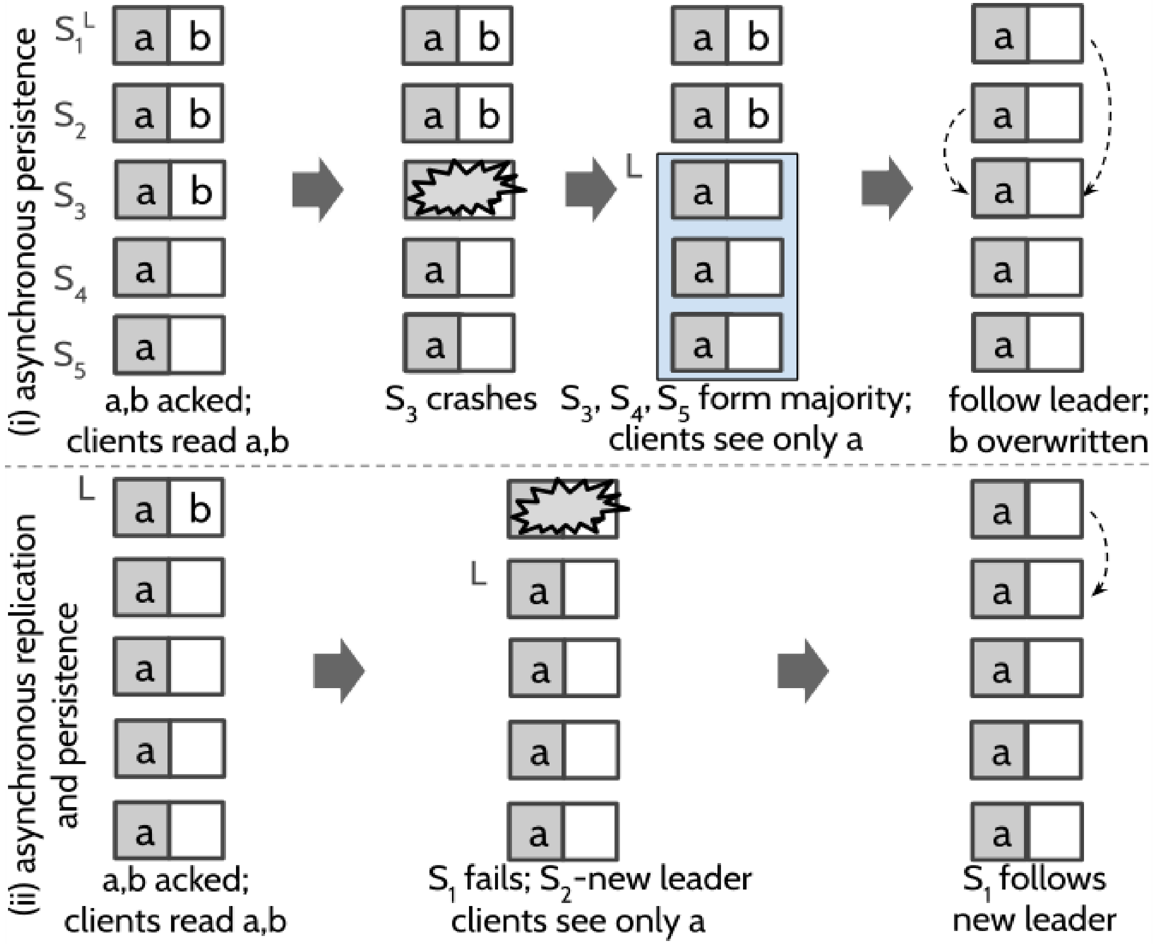
- Synchronous durability model (synchronously replicate and persist on a majority)



Replication	Persistence	Throughput (ops/s)	Avg. Latency (μs)
async	async	24215	330
sync	async	9889 (2.4× ↓)	809
sync	sync	2345 (10.3× ↓)	3412

Advantage: strong consistency
Disadvantage: low performance

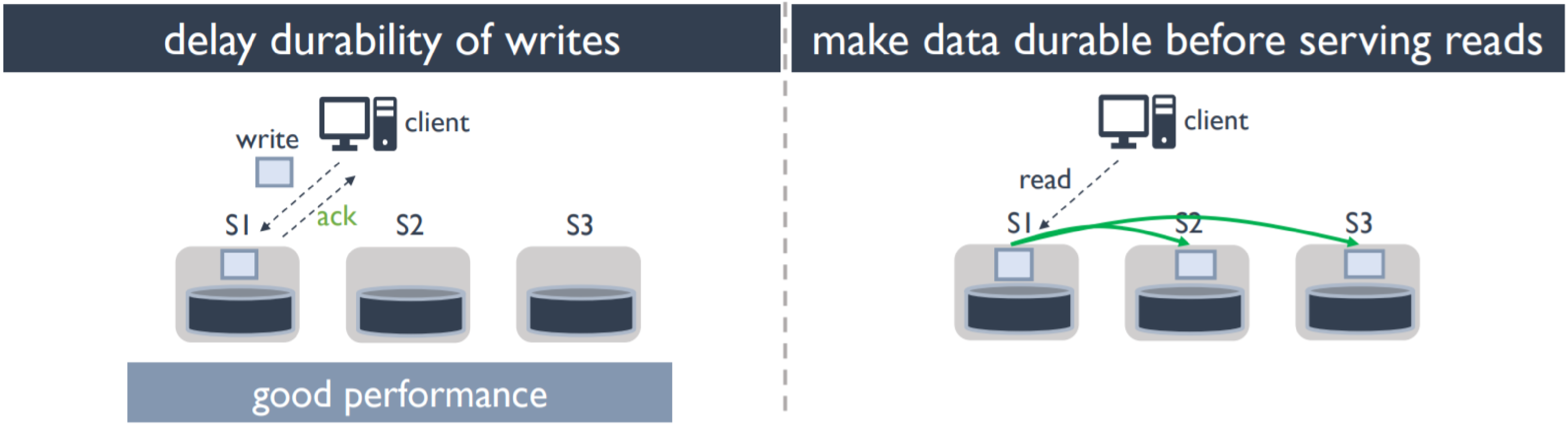
Asynchronous durability model (1.asynchronously persist
2.asynchronously replicate and persist)



Advantage: high performance
Disadvantage: weak consistency

Consistency-aware durability and Cross-client Monotonic read

- Consistency-aware durability(CAD)
- Key idea: CAD shifts the point of durability to reads from writes



Consistency-aware durability and Cross-client Monotonic read

- Cross-client Monotonic read
- Key idea: a read from a client guaranteed to return at least the latest state returned to a previous read from any client.

ORCA design

(basing on leader-based majority system)

Leader-based systems (e.g., MongoDB, ZooKeeper)

leader – a dedicated node

others are followers

writes flow through leader, establishes a single order

Majority

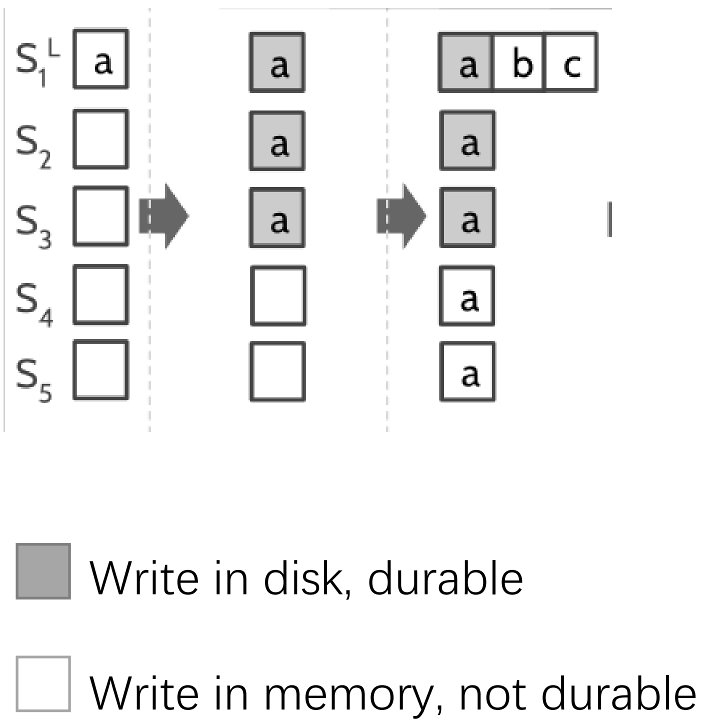
data is safe when persisted on majority nodes (e.g., 3 out of 5 servers)

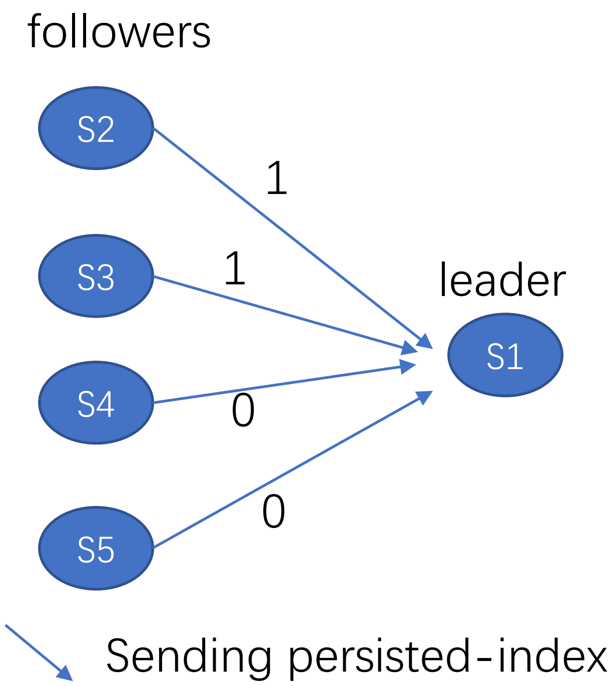
- **Durability-check mechanism**
- **Lease-based active set technique**
- **Two-step lease-breaking mechanism**

Durability-check mechanism

(efficiently separates requests that read non-durable items from those that access durable ones)

- Durable-index: index of the latest durable item in the system(owned by leader)
- Persisted-index: index of latest persistency in every member node (owned by leader and followers in order to update durable-index)
- Update-index of item i: index of the last update that modified i
- Durability check: i is durable if update-index of i ≤ durable-index of system



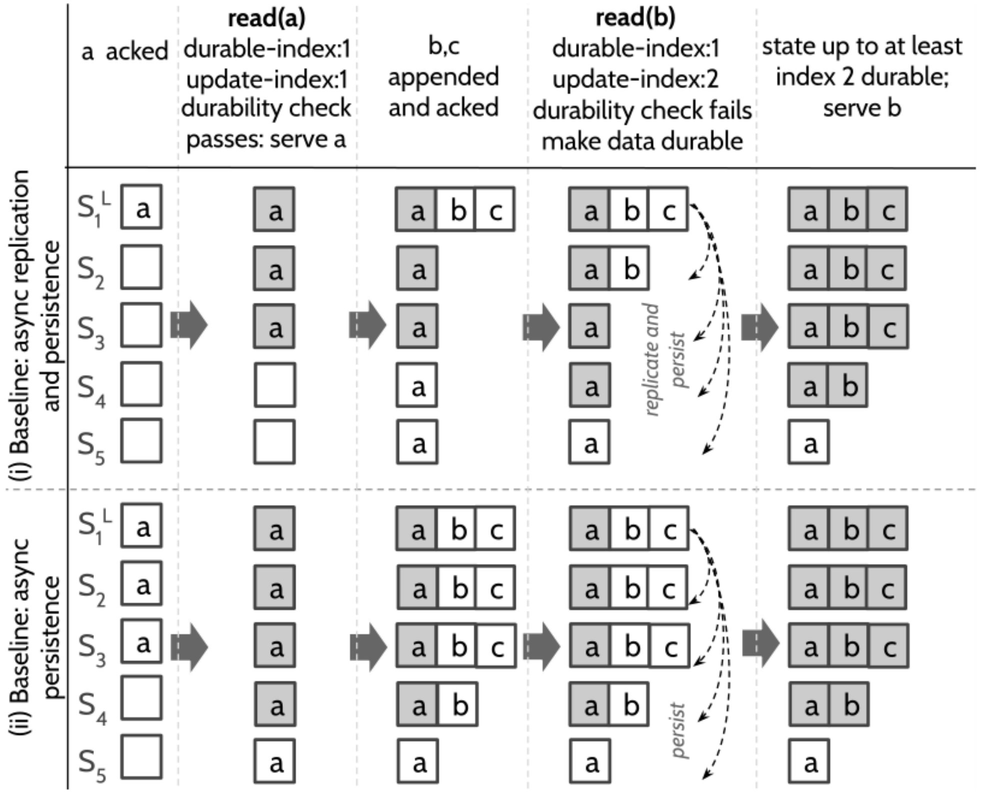


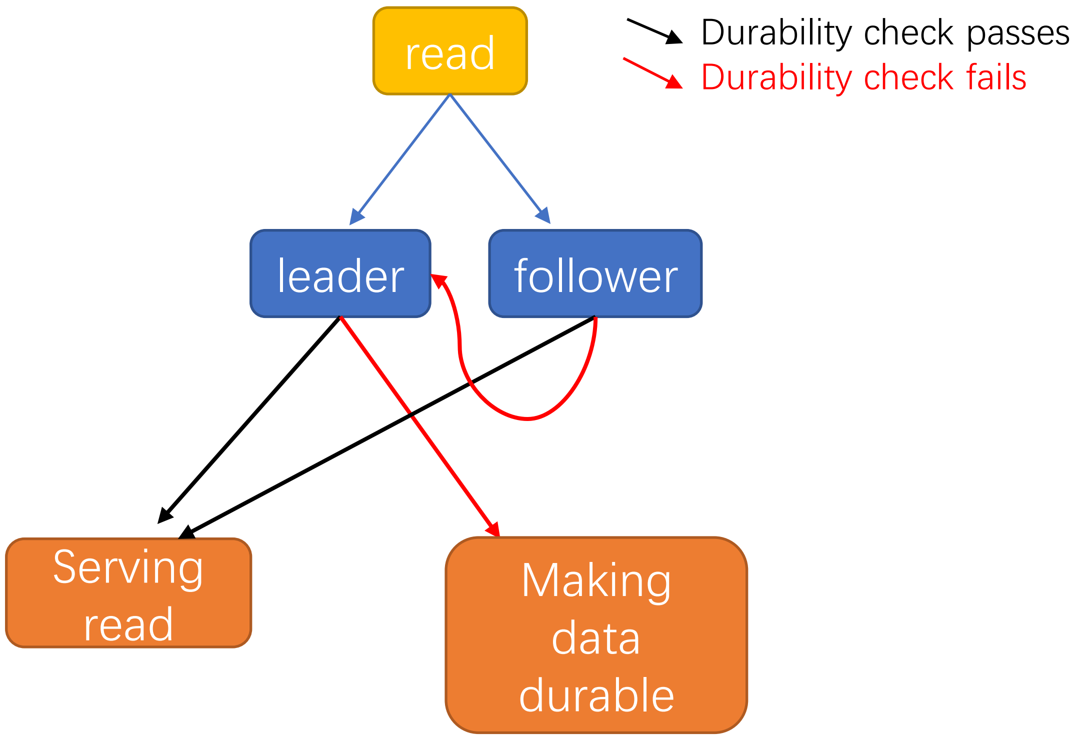
If the majority has the same persisted-index, then leader makes sure that durable-index=this persisted-index

Durability-check mechanism

(efficiently separates requests that read non-durable items from those that access durable ones)

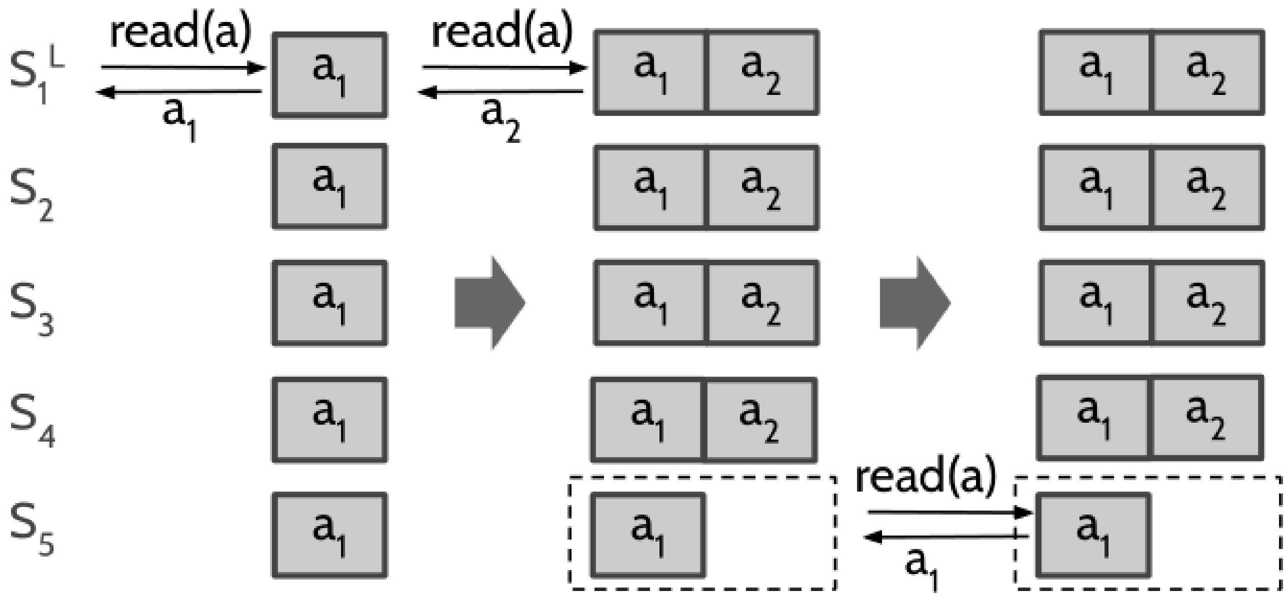
- Durable-index: index of the latest durable item in the system(owned by leader)
- Persisted-index: index of latest persistency in every member node (owned by leader and followers in order to update durable-index)
- Update-index of item i: index of the last update that modified i
- Durability check: i is durable if update-index of i≤durable-index of system





Lease-based active set technique
(ensures monotonic reads while allowing reads at many nodes)

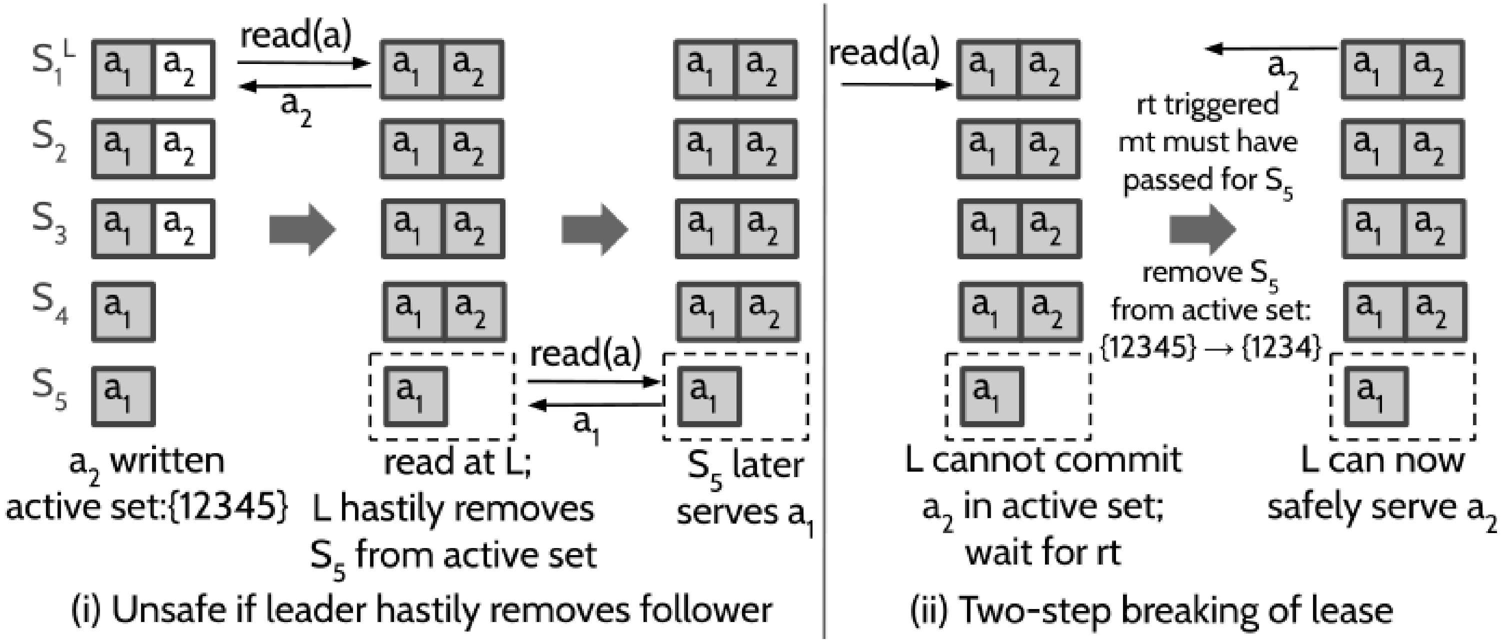
Requests: **R1**: When the leader intends to make a data item durable (before serving a read), it ensures that the data is persisted and applied by *all* the members in the active set.
R2: Only nodes in the active set are allowed to serve reads.



Problem: S_5 could provide old data if marking itself out after removed out of active set by leader.

Two-step lease-breaking mechanism
(helps correctly manage active-set membership)

Requests: once mt(mark-out timeout) passes, the follower marks itself out; once rt(mark-out) timeout passes, the leader removes the follower from the active set.

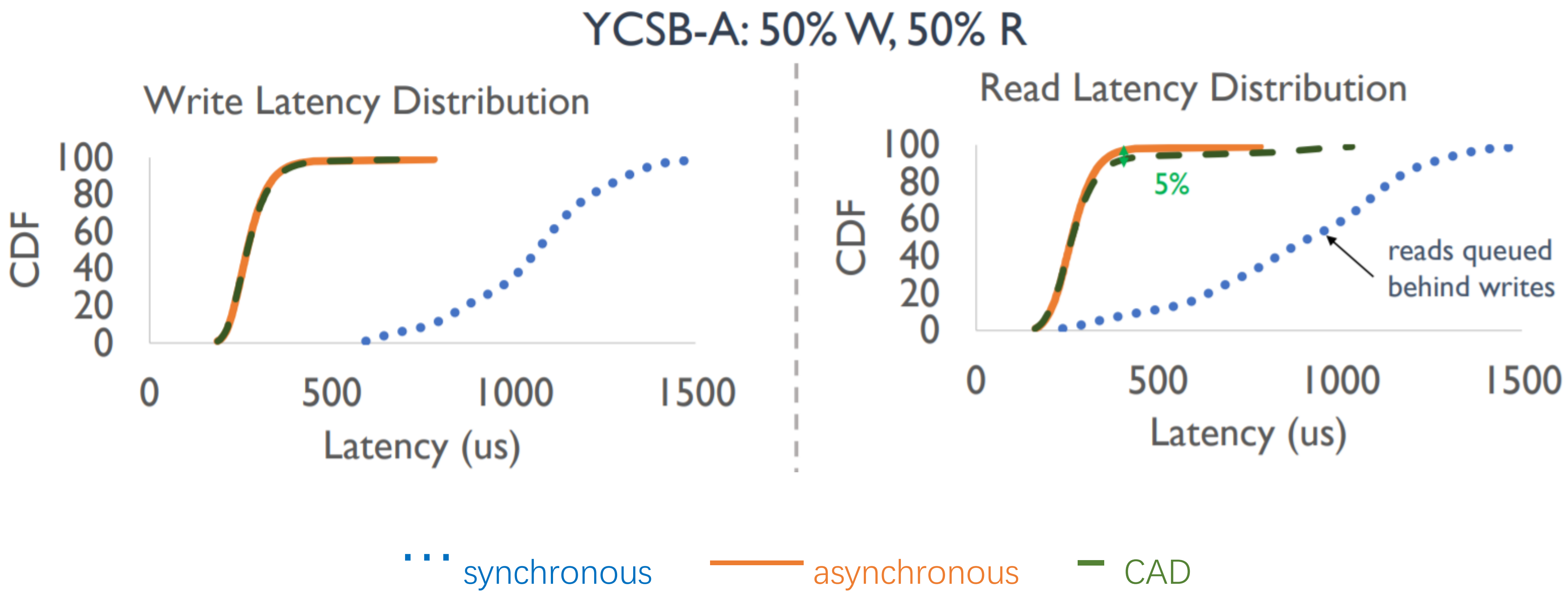


rt≥5*mt makes sure that marking out is before removing out

Evaluation

- Implemented in ZooKeeper
- Evaluate different durability models
- compare CAD against synchronous durability model and synchronous one
- Evaluate overall system performance
- ORCA against strongly and weakly consistent ZooKeeper

CAD Durability Layer Performance

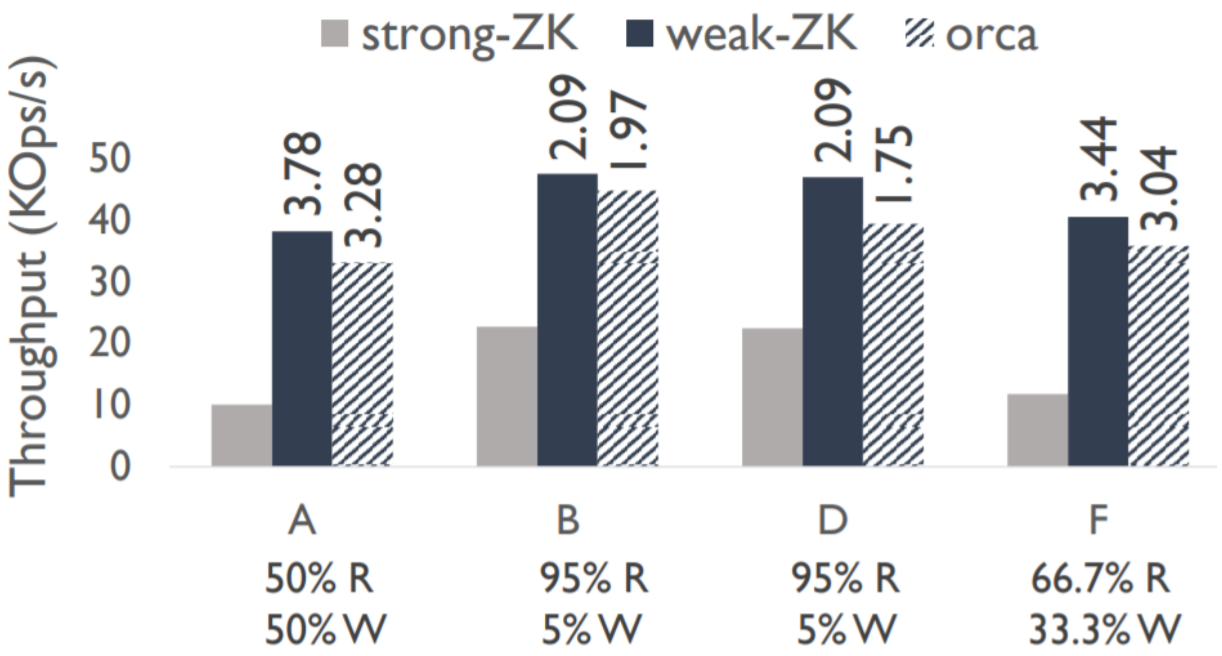


CAD matches the performance of synchronous durability but writes faster than asynchronous one.

Only 5% slower than read latency of asynchronous due to no-durable read.

ORCA system performance

- Strong-ZK——uses synchronous durability, reads only at leader
- Weak-ZK——uses asynchronous durability, reads at many notes
- ORCA——uses CAD, read at many nodes



Strong-ZK performs poorly due to immediate durability and leader-restricted reads

Weak-ZK performs well due to eventual durability and scalable reads

ORCA adds little overheads compared to weak-ZK

reads that access non-durable data