30th IEEE International Symposium on High-Performance Computer Architecture (HPCA 2024)

“MINOS*: Distributed Consistency and Persistency Protocol Implementation & Offloading to SmartNICs”

Antonis Psistakis, Fabien Chaix❖, and Josep Torrellas
University of Illinois Urbana-Champaign, USA
❖FORTH, Greece
{psistaki, torrella}@illinois.edu, chaix@ics.forth.gr

*MINOS: King of Crete island (greek mythology)

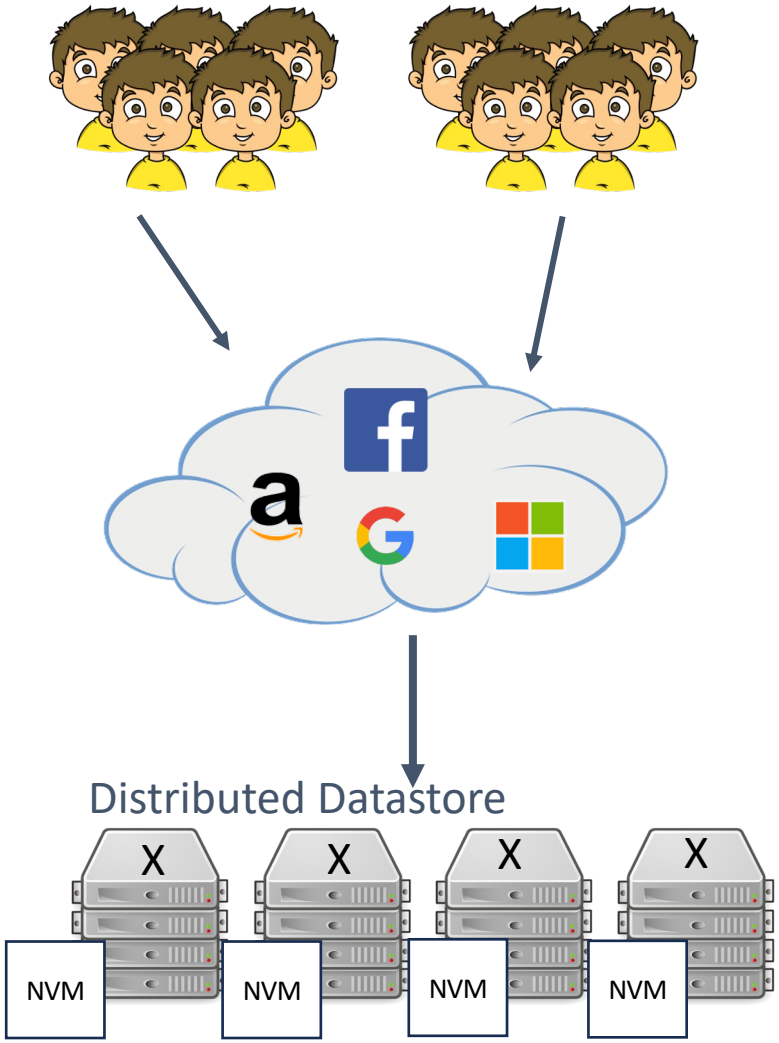
Edinburgh, Scotland
March 5th, 2024

Introduction

- Datacenters focus
  - Performance (latency and tput)
  - Availability
  - Reliability

To achieve the above:

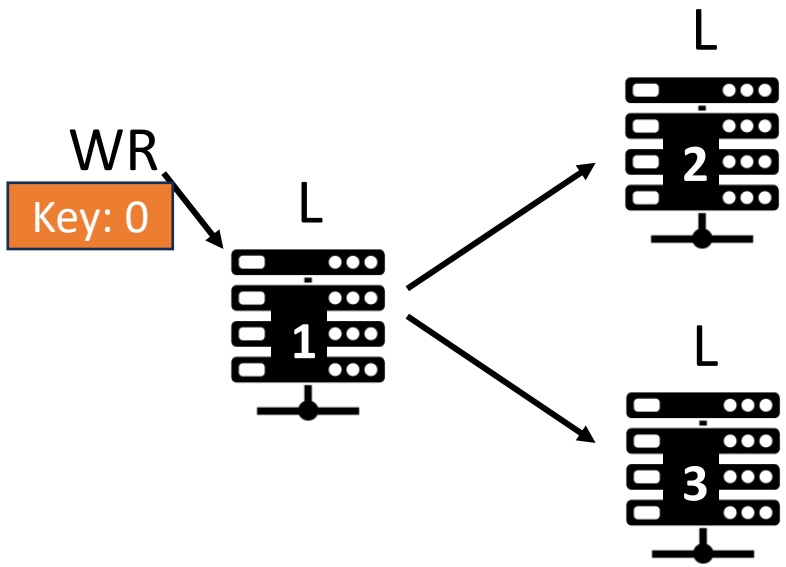
- Replicate data across Distributed Datastores
- Persistency

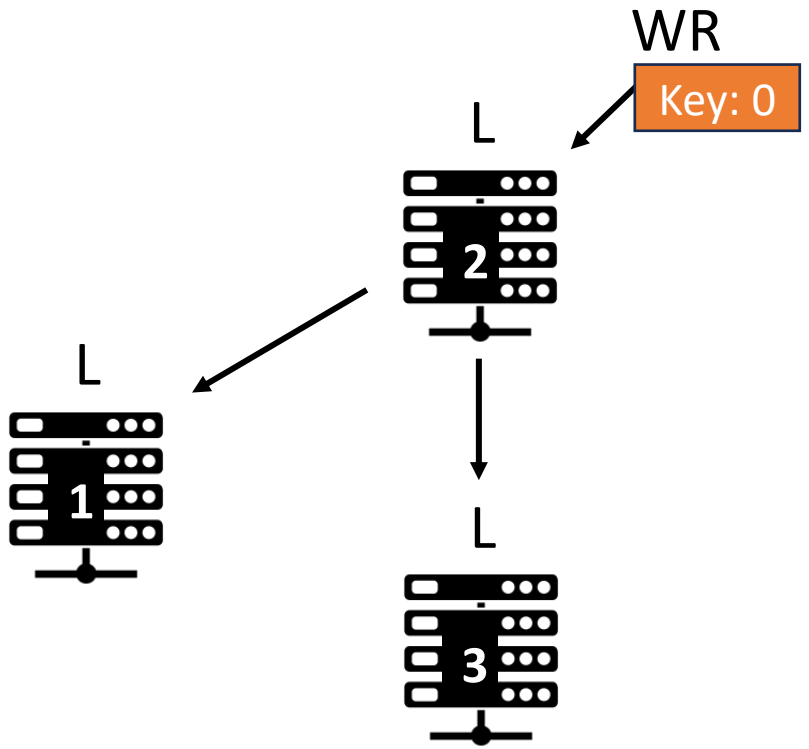


Introduction

Leaderless distributed systems: All participating nodes can process any client request

- 👍 Performance
- 👎 Programming complexity, Recovery





Background: Definitions

- **Consistency Model**: When updates become visible (replicated) to all nodes
  - **Linearizable** (Lin): A client write must update all the replica nodes in the system before it completes.

- **Persistency Model**: When updates are persisted to non-volatile memory (NVM)

Persistency Model*	When is an update persisted in a node's NVM?
Synchronous (Synch)	When the local volatile memory is updated
Read-Enforced (REnf)	Before the value updated is read
Scope	At the end of the scope
Eventual (Event)	Sometime in the future

i-acoma group

* Kokolis et al., "Distributed Data Persistency", MICRO 2021

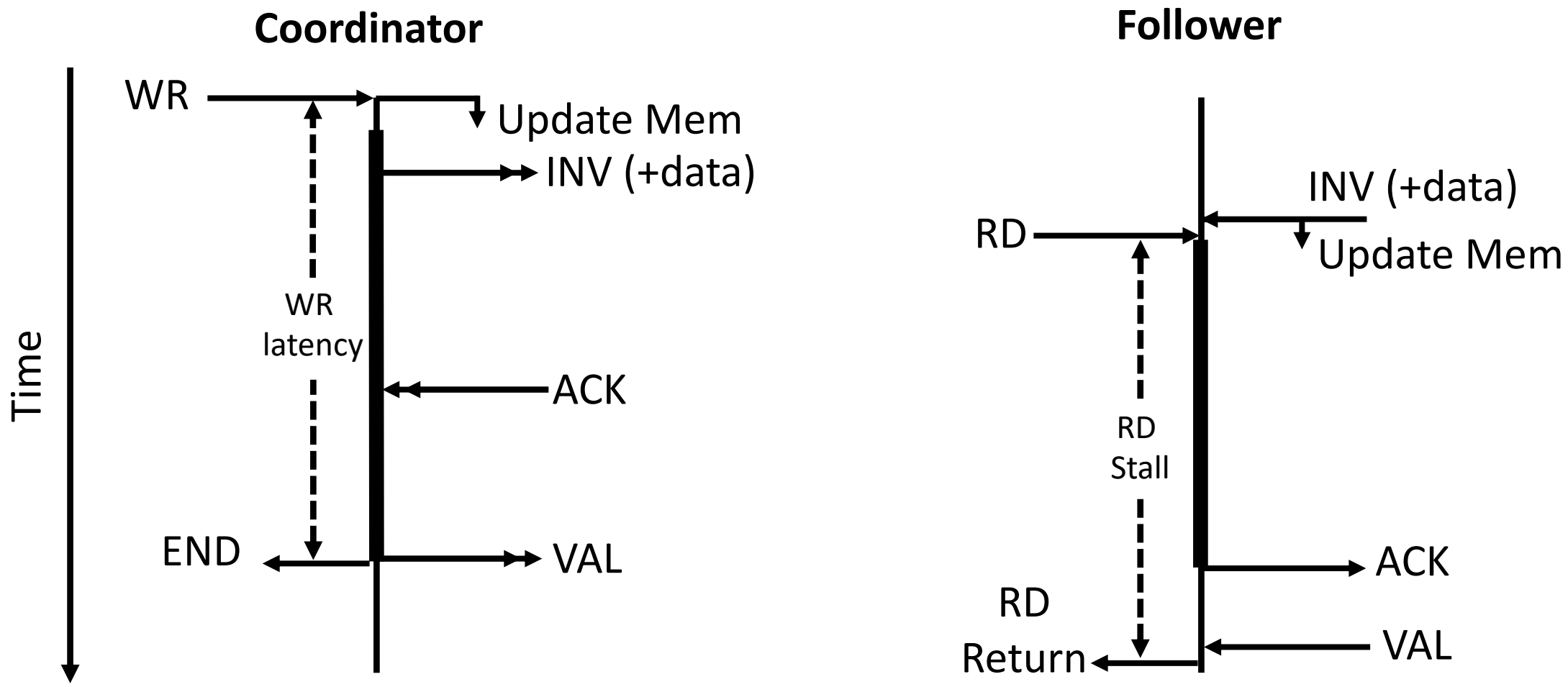
Background: Definitions

- **Type of Nodes**
  - **Coordinator**: The node that receives the client request
  - **Follower**: The nodes with replicas. Need to participate in the update
- **Type of Messages***

Message
Invalidation (**INV**)
Acknowledgement (**ACK**)
Validation (**VAL**)

* Katsarakis et al., “Hermes: A Fast, Fault Tolerant and Linearizable Replication Protocol”, ASPLOS 2020

Background: DDP <Lin, Synch> Example



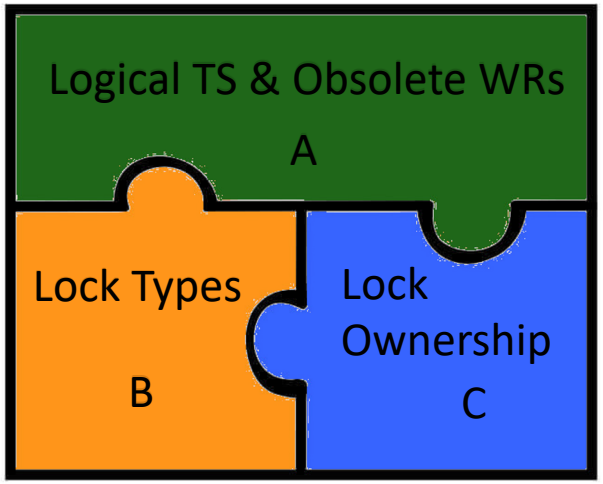
Contributions

1. Novel algorithms for real-system implementation of a *Leaderless* system, supporting various *consistency* and *persistency* models (MINOS-B)

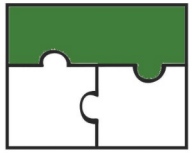
2. New architecture that *offloads* the models into SmartNICs (MINOS-O)

Contribution: MINOS-Baseline (MINOS-B)

- Set of novel leaderless algorithms that efficiently implement consistency and persistency (DDP) models
- MINOS-B relies on three elements to support concurrent and conflicting writes:
  A. Logical Timestamps & Obsolete Writes
  B. Lock Types
  C. Lock Ownership



MINOS-B: Logical Timestamps & Obsolete Writes



Logical Timestamps

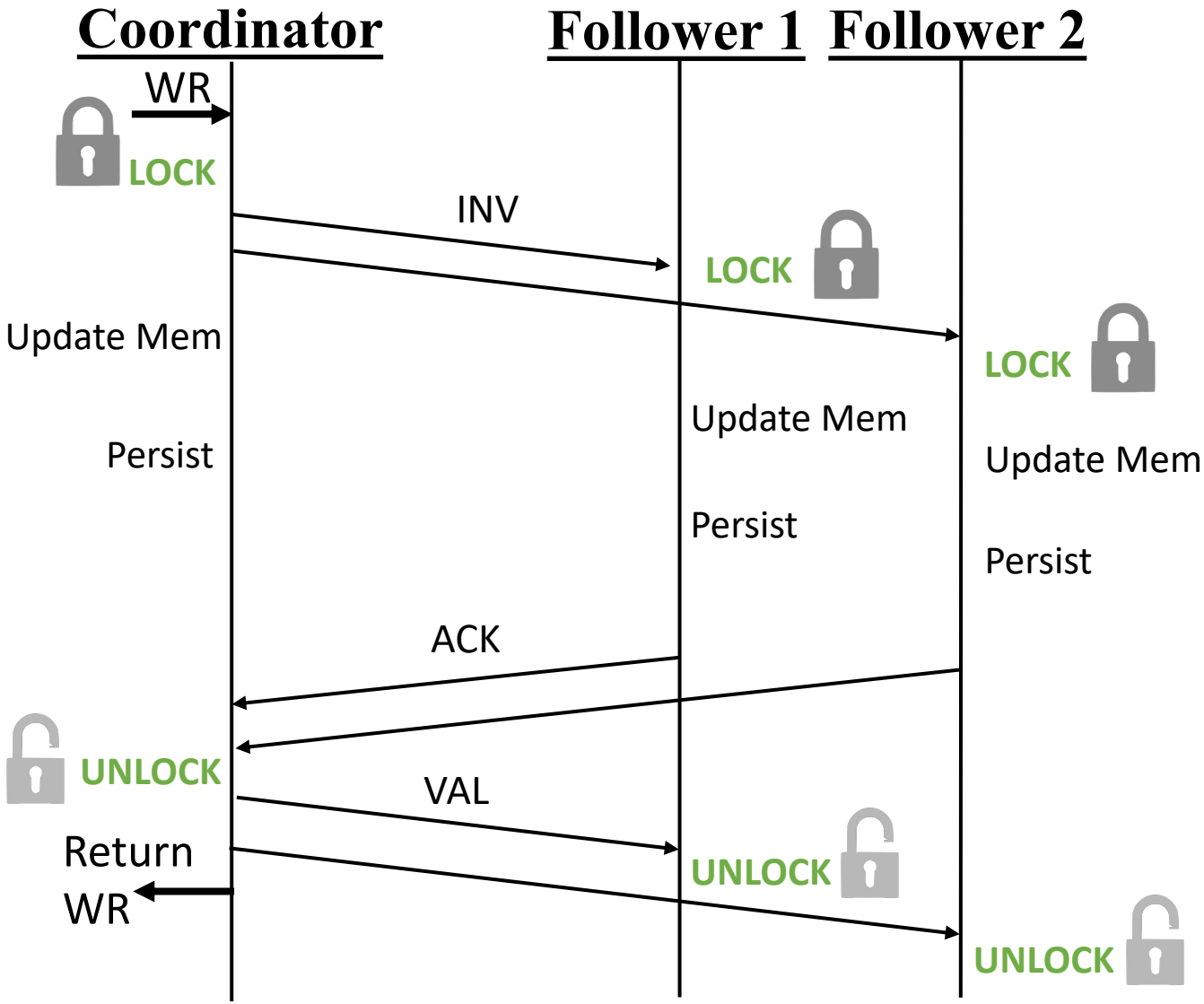
- Used to maintain order of requests
- Each write operation carries its own timestamp

Obsolete Writes

- Write that reaches a node and the record is already updated by a later update
- Returns immediately to the sender without updating record

MINOS-B: Lock Types

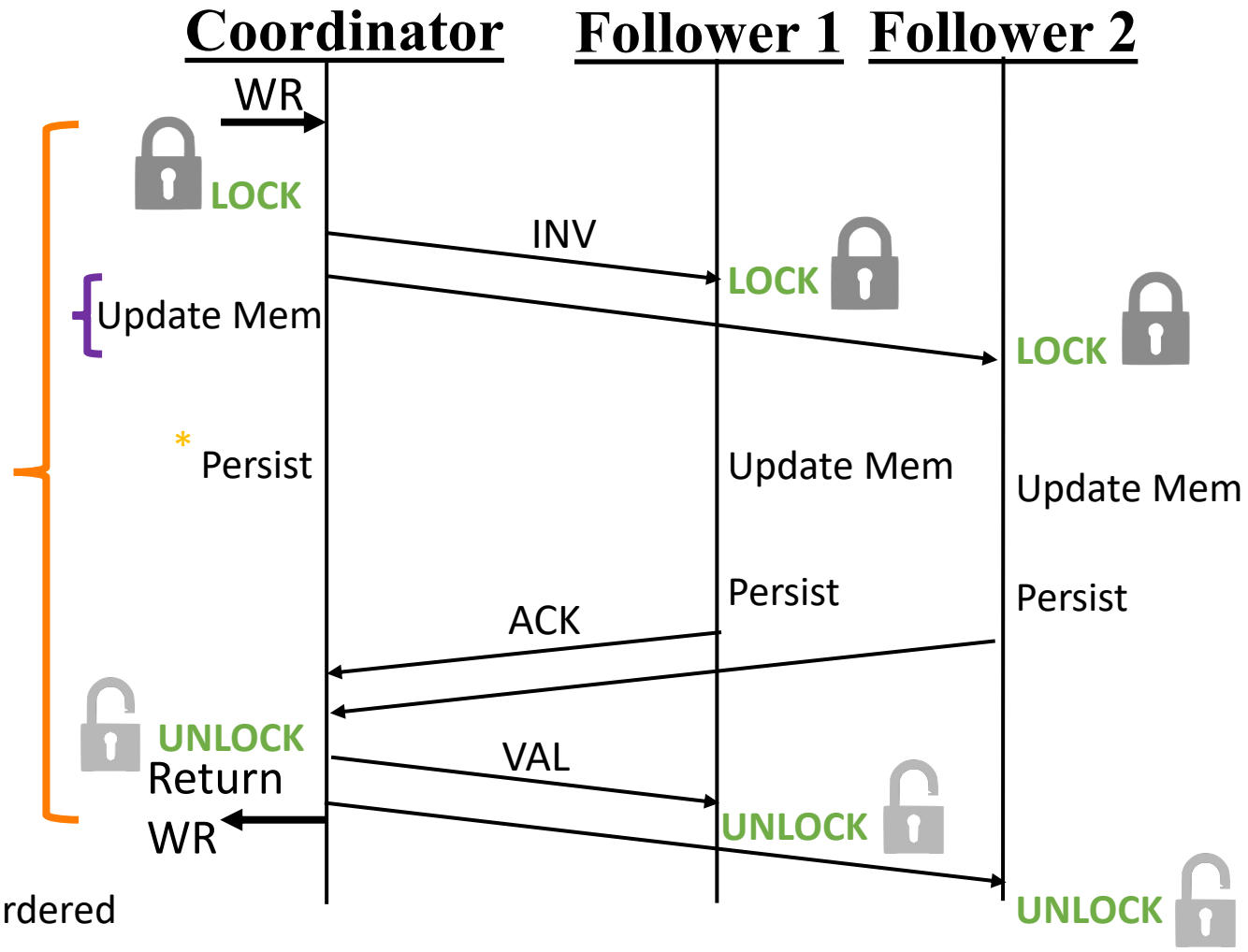
Naïve approach: Use plain Locks



MINOS-B: Lock Types

What <u>really</u> needs lock protection?

1. Prevent reads on records currently being written
   - Scope: Client Write
2. Prevent concurrent writes on the same record
   - Scope: Local Write



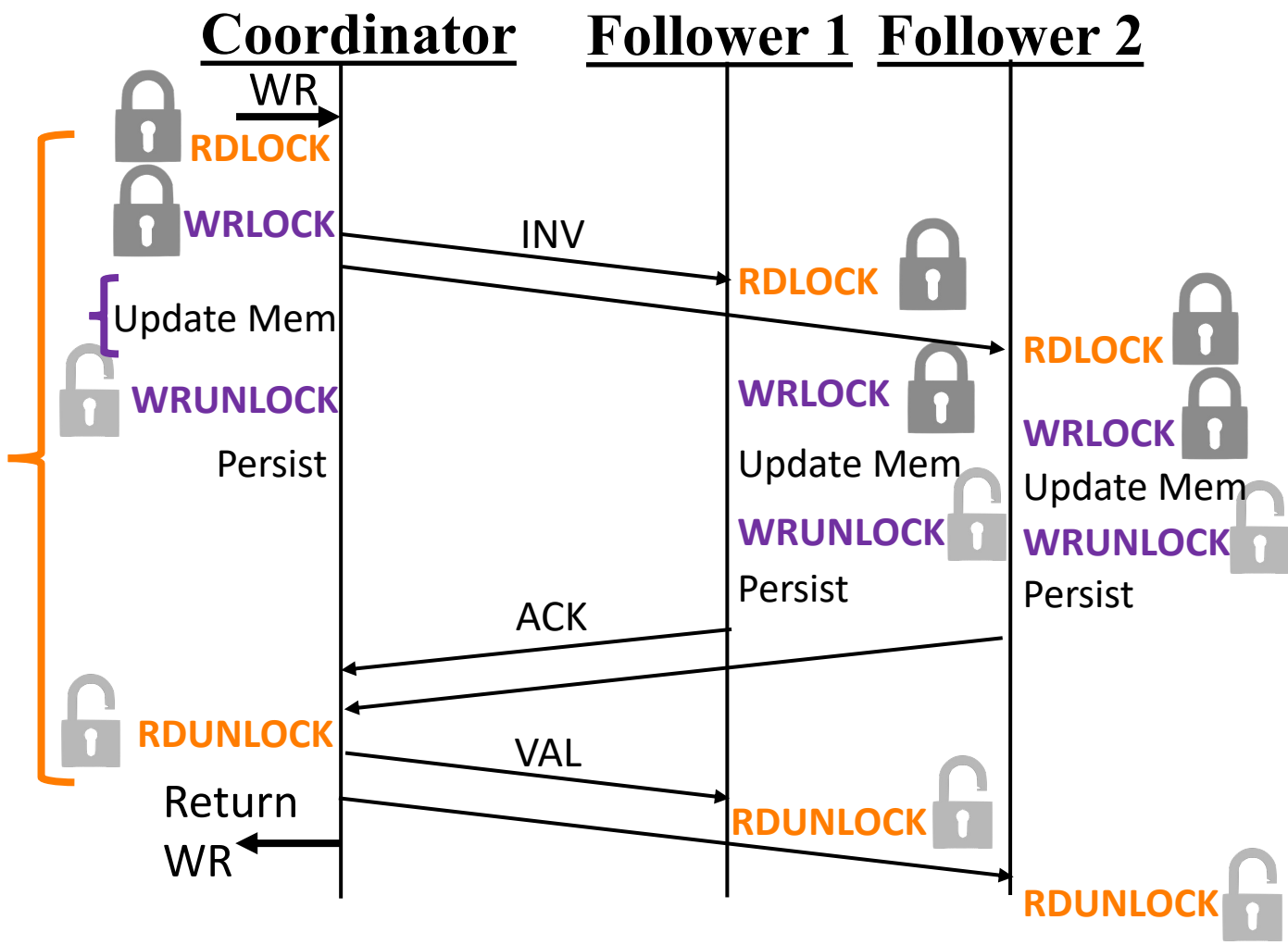
* Note: Persist is done atomically in a REDO log, okay to be re-ordered

MINOS-B: Lock Types

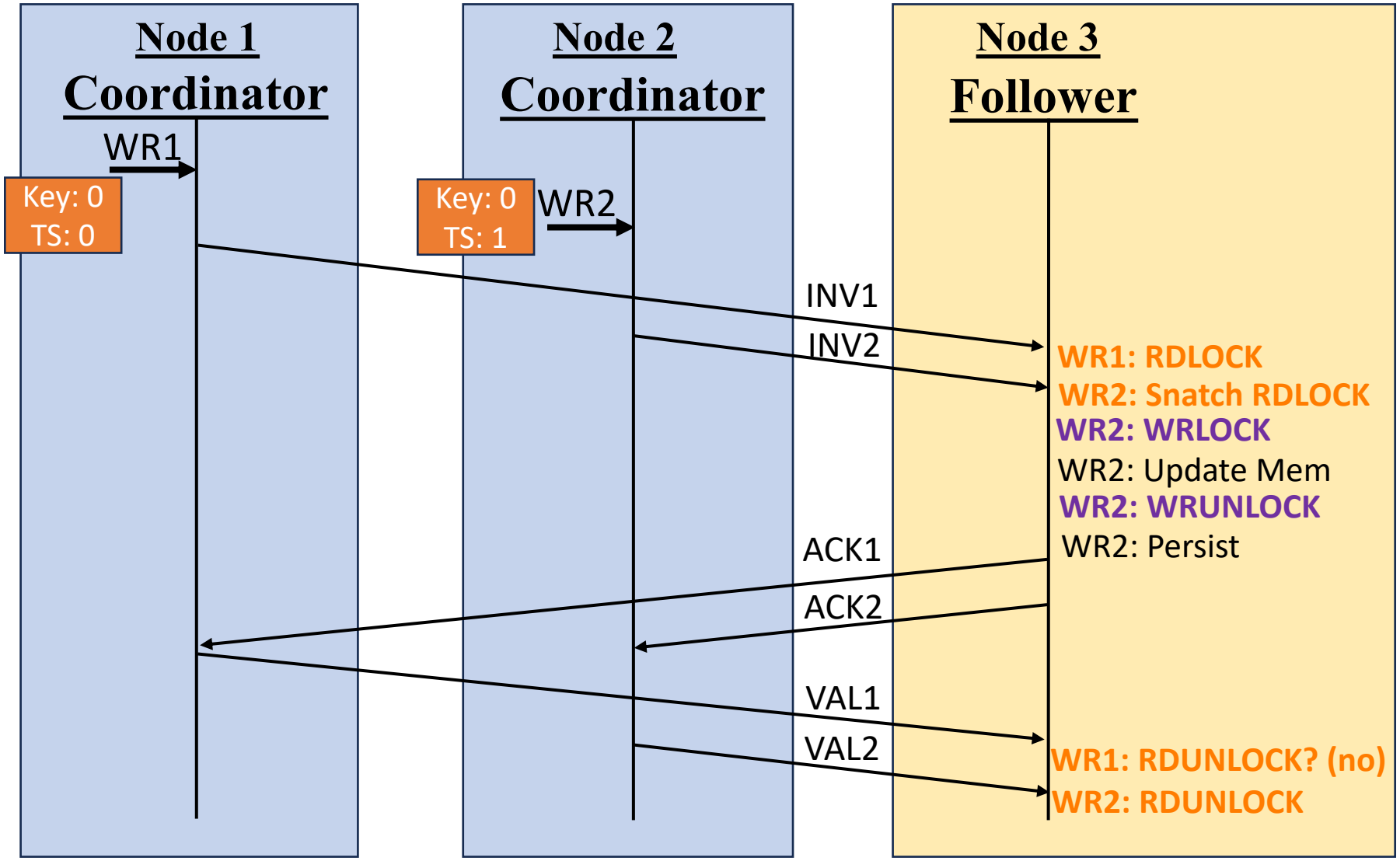
Our Approach

1. Prevent reads on records currently being written
   - Scope: Client Write
2. Prevent concurrent writes on the same record
   - Scope: Local Write

Optimization: Snatching Read-Lock ownership



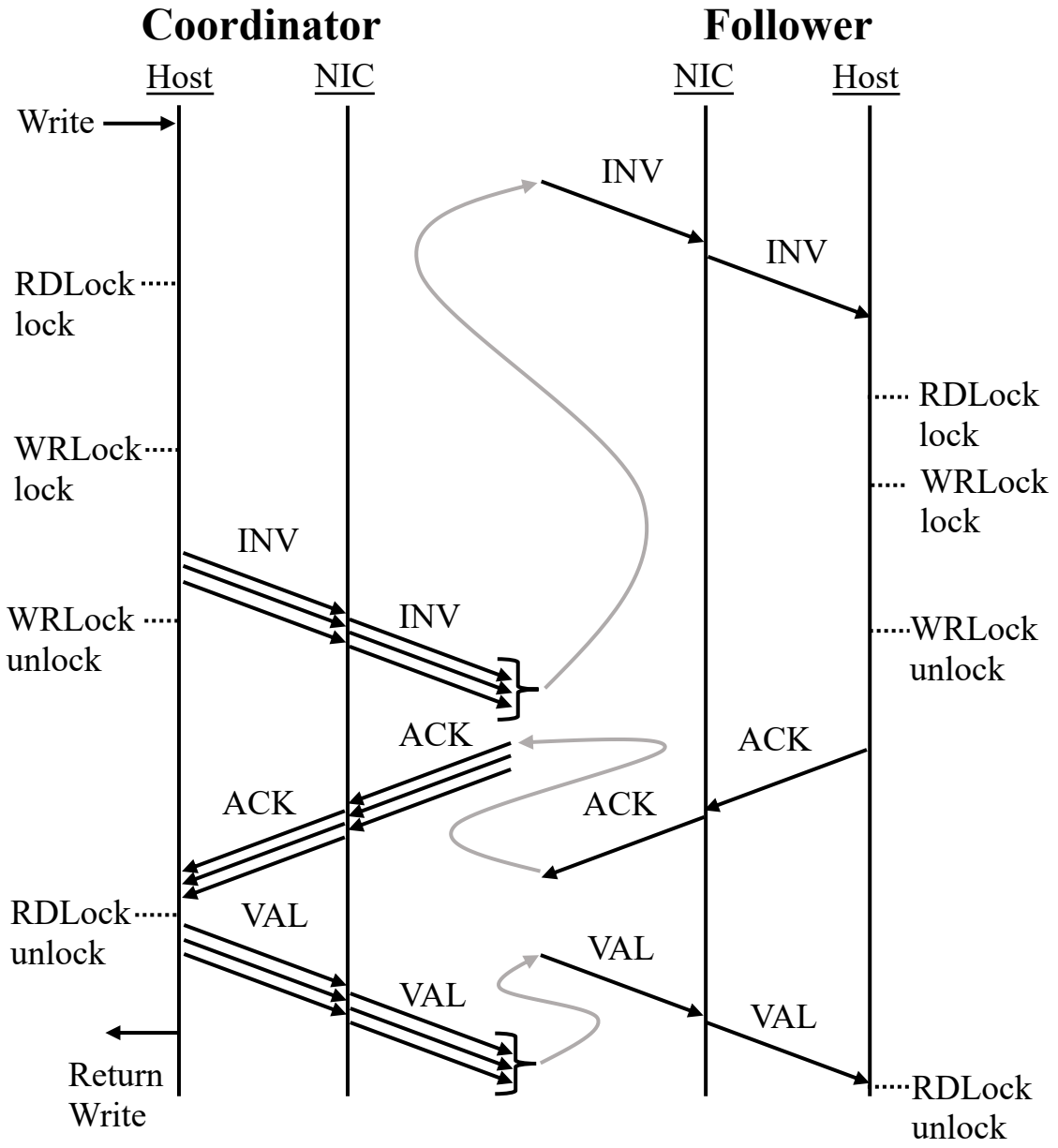
MINOS-B: Read-Lock Ownership Example



NODE 3	KEY 0
RDLOCK	<null>
WRLOCK	0
Volatile State	<WR2>

MINOS-B: Full <Lin, Synch> Algorithm

1. Process new WR for key *k*
2. If Obsolete(TS_{WR}):
3. | Exit
4. Snatch RDLock(k)
   // RDLock is set.
   // This thread may be the RDLock Owner.
5. WRLock(k)
   // This thread can perform a local-write.
6. If !Obsolete(TS_{WR}):
7. | Send INVs — a
8. | Update Mem
9. WRUnlock(k)
10. If INVs sent: — b
11. | Persist to NVM — c
12. | Spin for all ACKs — d
13. If RDLock_Owner(k) == Me:
14. | RDUnlock(k)
15. If INVs sent: Send VALs — e



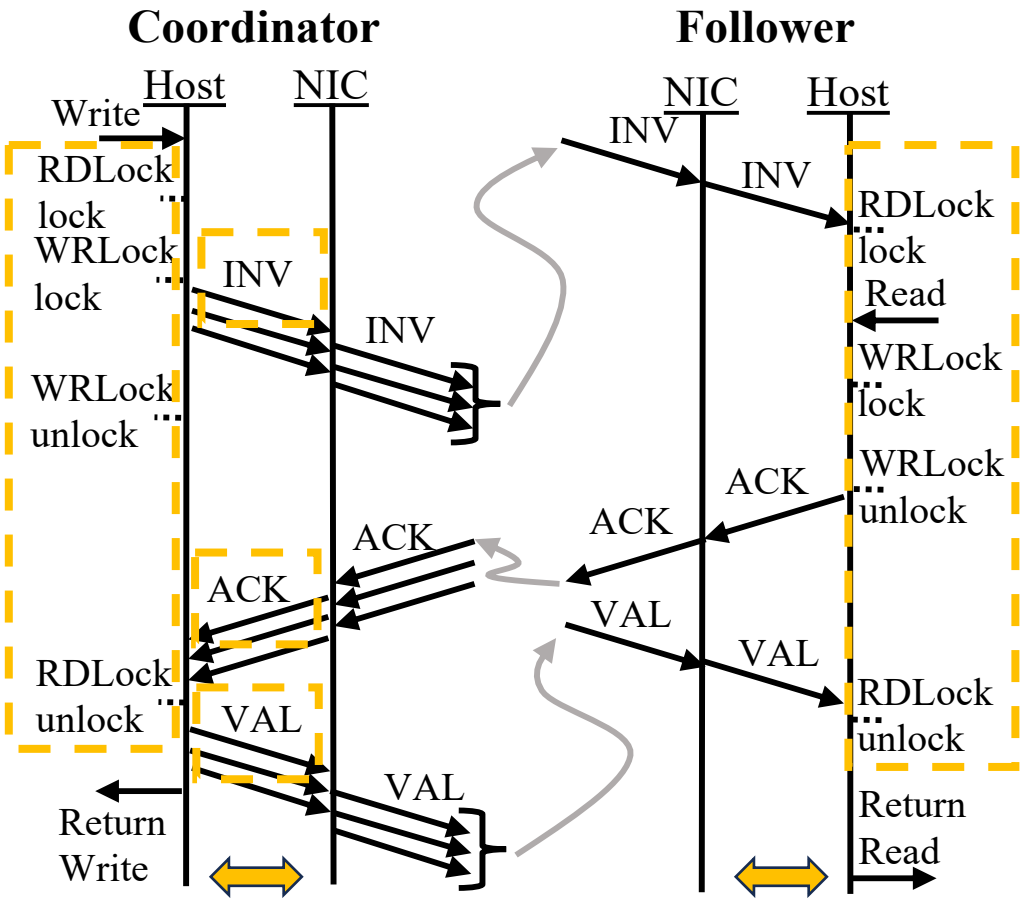
16. Process new INV for key *k* — f
17. If Obsolete(TS_{WR}):
18. | Send ACK — g
19. | Exit
20. Snatch RDLock(k)
21. WRLock(k)
22. If !Obsolete(TS_{WR}):
23. | Update Mem — h
24. WRUnlock(k)
25. If updated Mem: Persist to NVM
26. Send ACK — i

27. Process new VAL for key *k* — j
28. If RDLock_Owner(k) == Me:
29. | RDUnlock(k)

30. < Not applicable > — k

Sources of Overhead

- Same msg sent to multiple receivers one-by-one
- Heavy involvement of the Follower CPUs
- Expensive PCIe crossings
- Software overhead of concurrency control



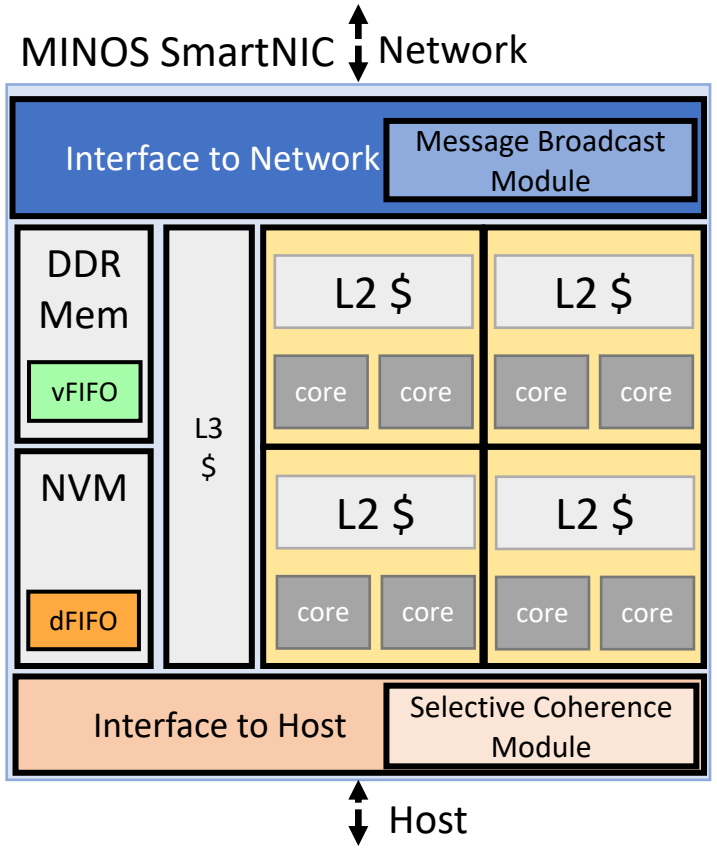
MINOS-B <Lin, Synch> Algorithm

Contribution: MINOS-Offload (MINOS-O)

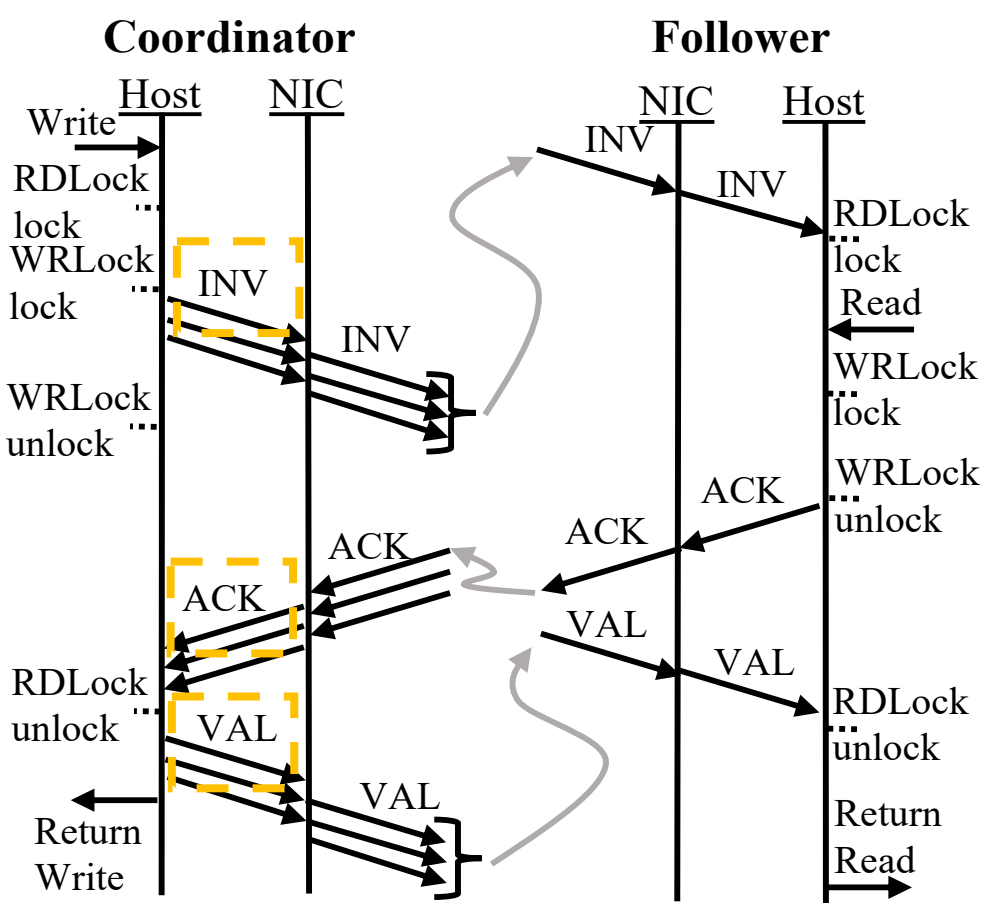
Idea: Offload consistency and persistency operations to SmartNICs

SmartNIC enhancements:

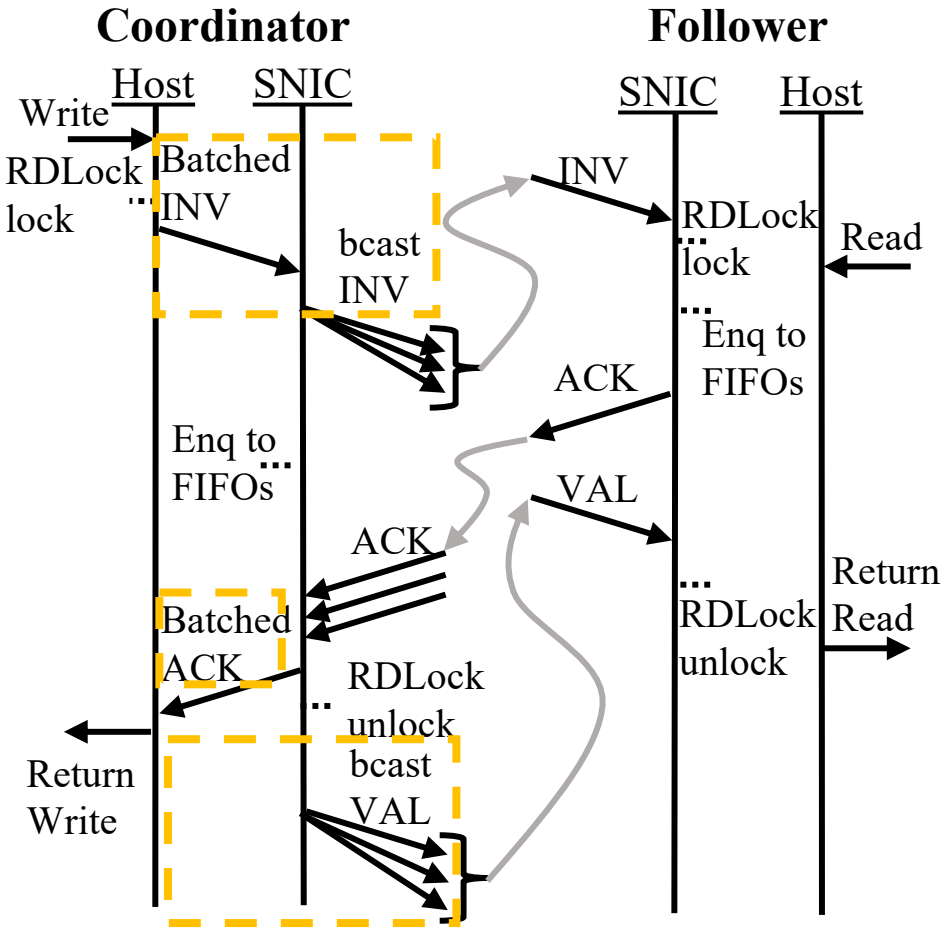
1. Equipped with both *volatile*- and *non-volatile* memories
2. Metadata is *cache-coherent* with host CPU
3. Batching of messages between Host ⇔ SmartNIC
4. Broadcast support



MINOS-O Features



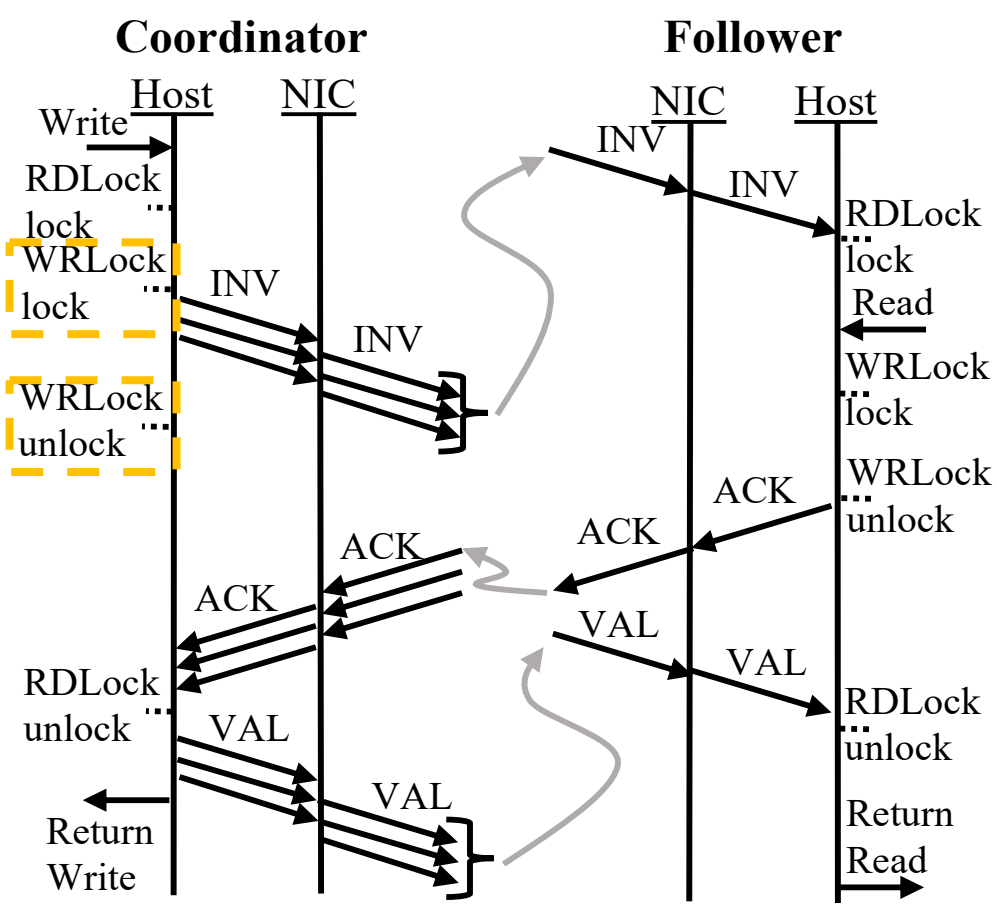
MINOS-B <Lin, Synch> Algorithm



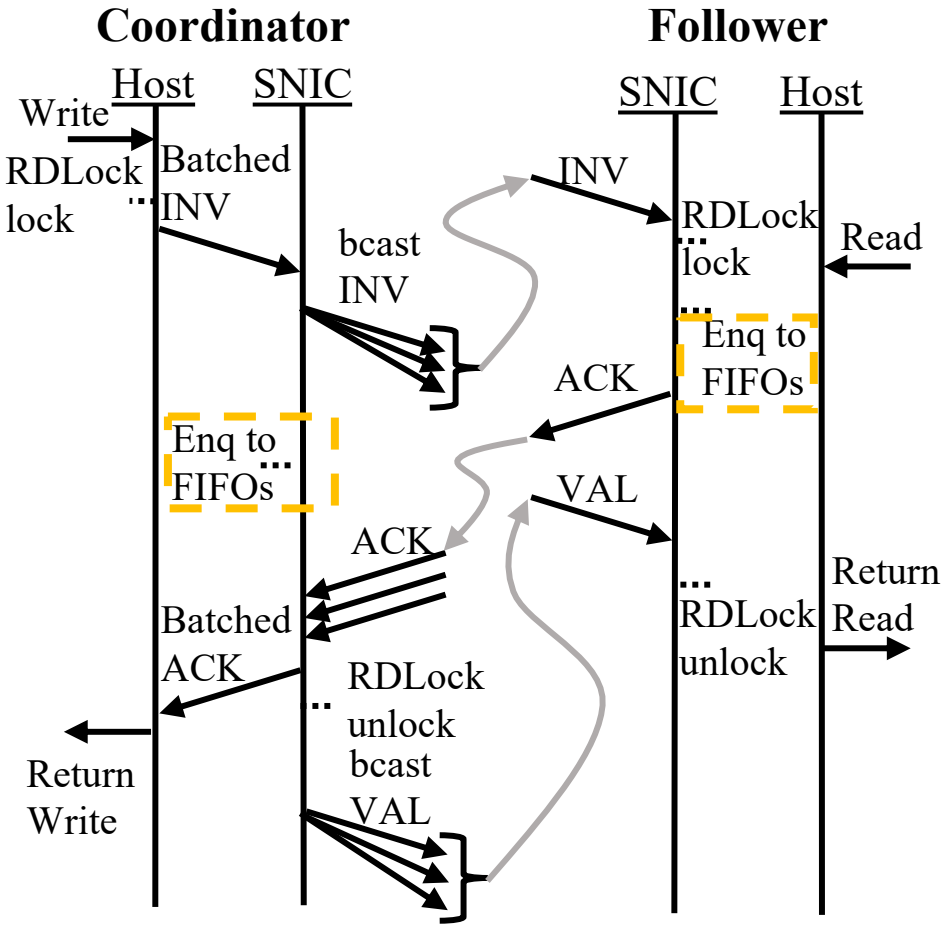
MINOS-O <Lin, Synch> Algorithm

1. Batched Messages + Broadcast

MINOS-O Features



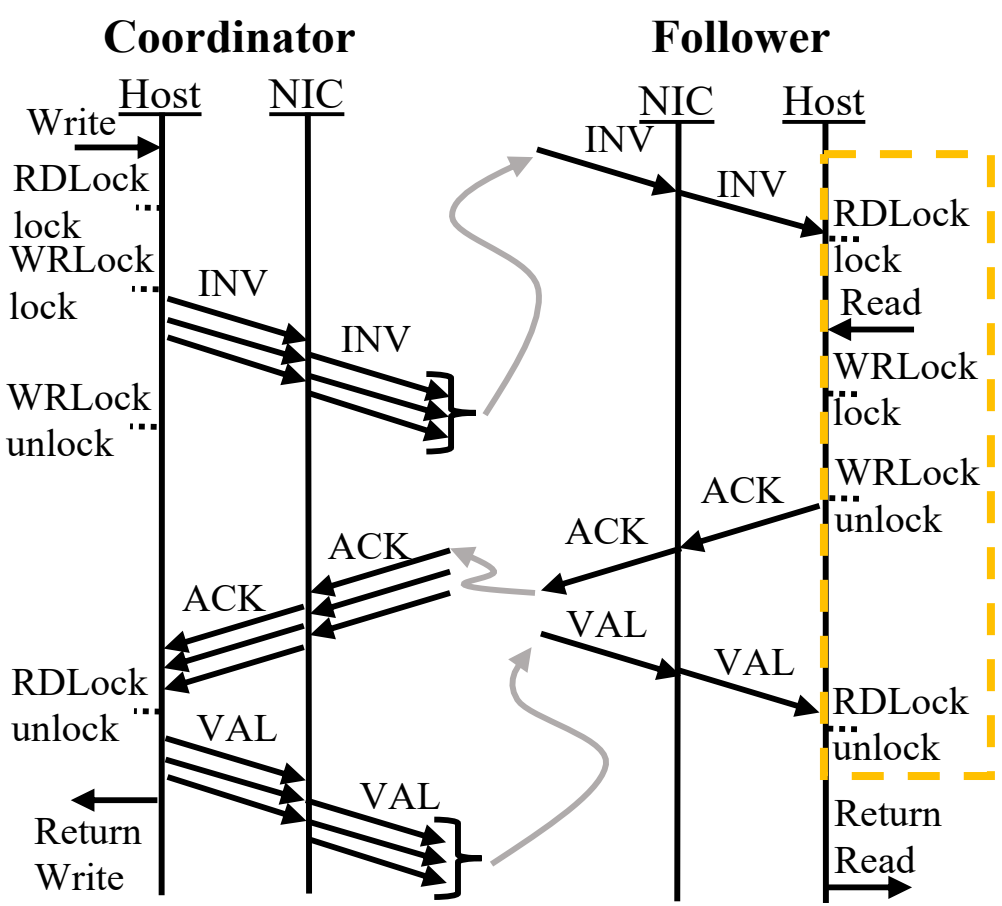
MINOS-B <Lin, Synch> Algorithm



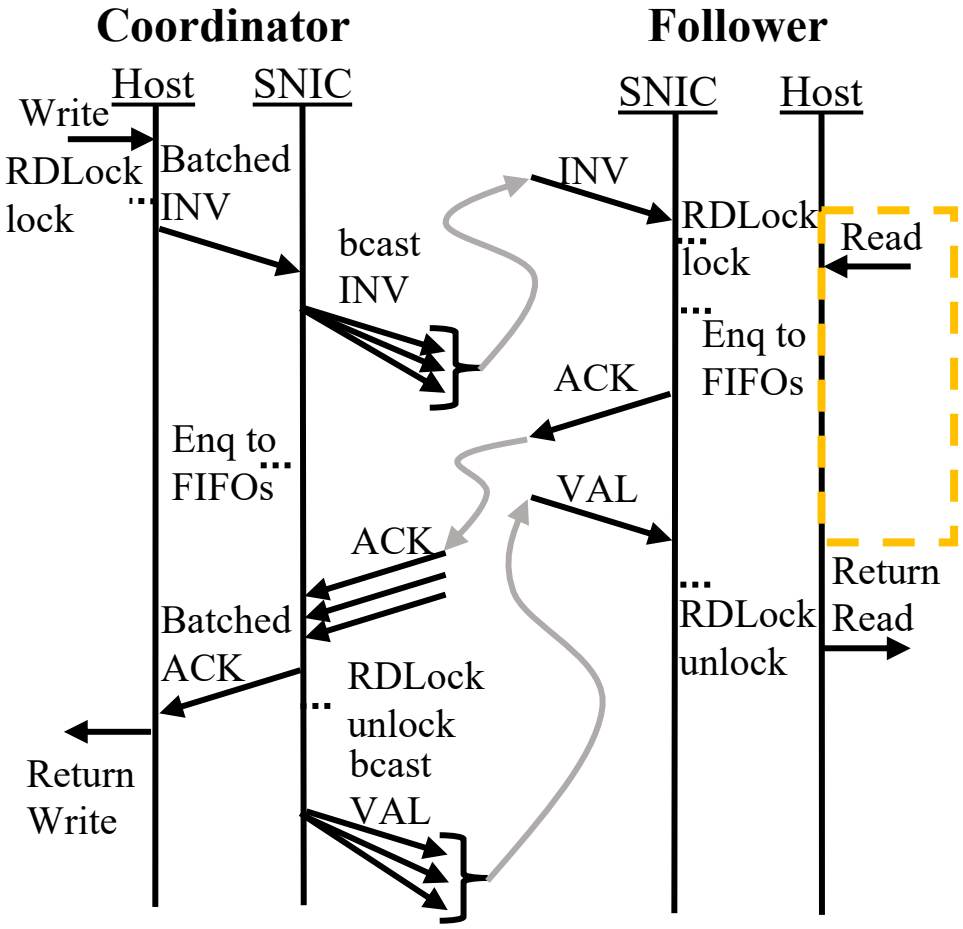
MINOS-O <Lin, Synch> Algorithm

2. Offload models to SmartNIC

MINOS-O Features



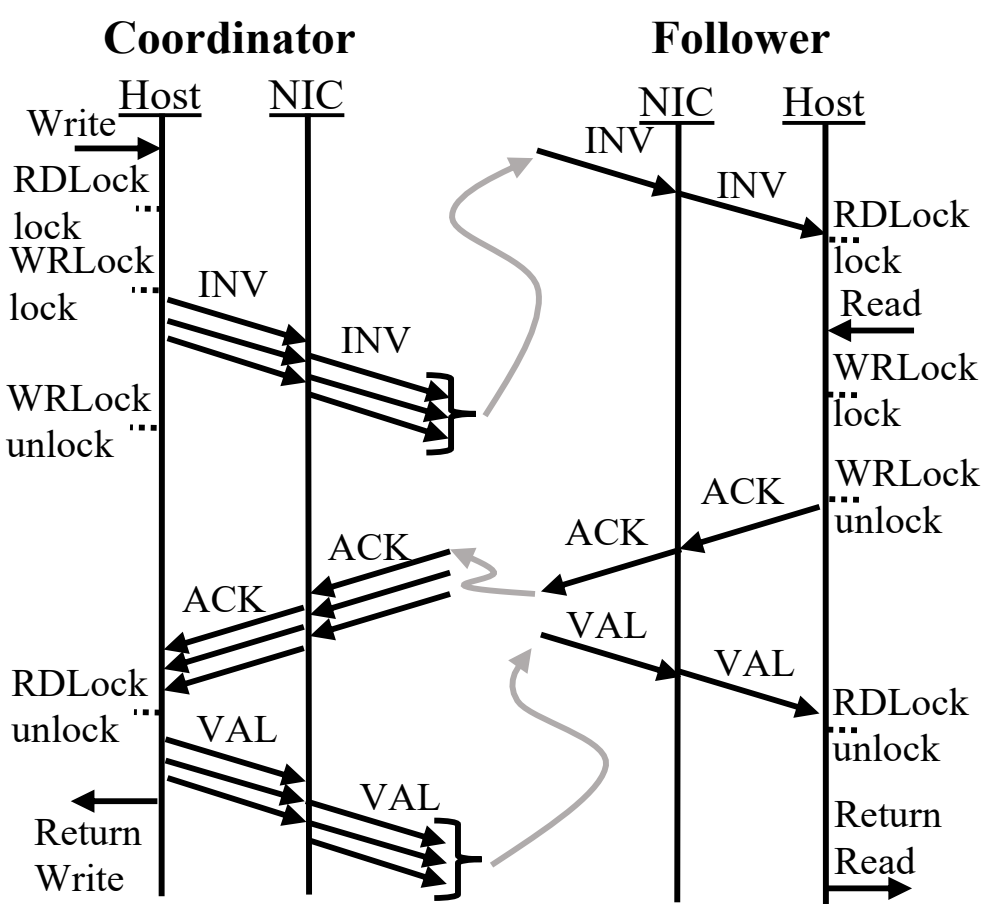
MINOS-B <Lin, Synch> Algorithm



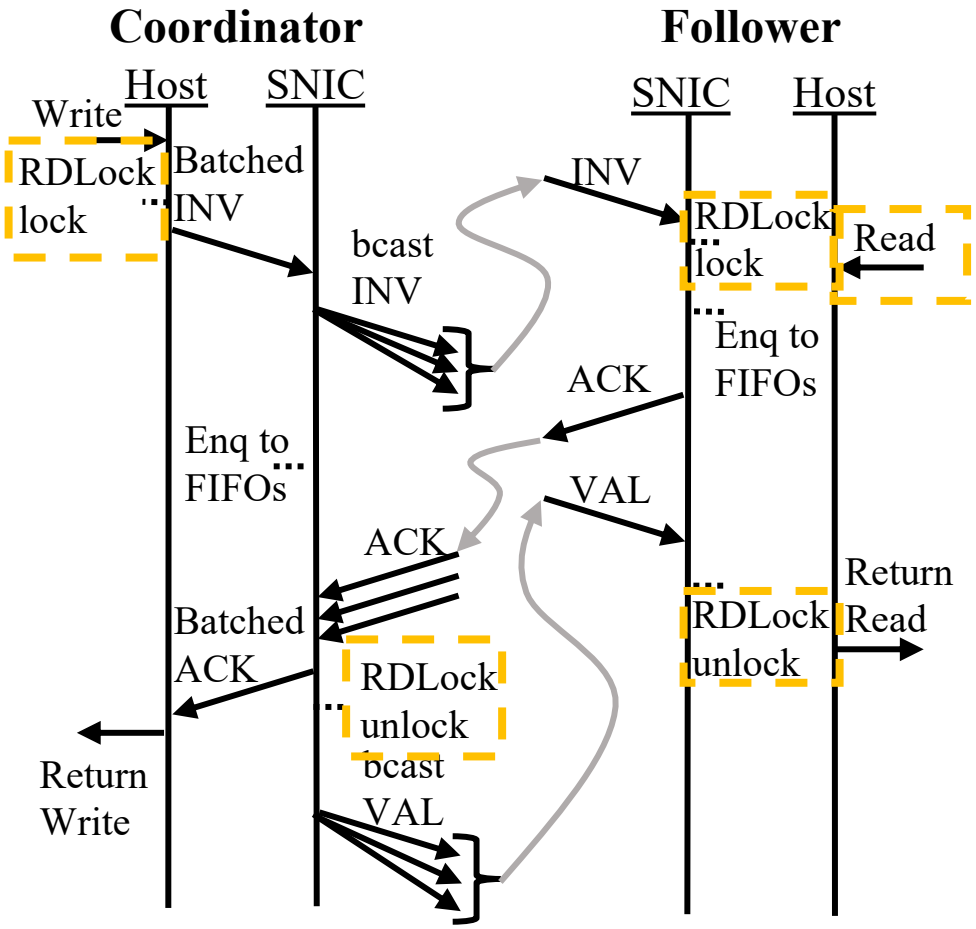
MINOS-O <Lin, Synch> Algorithm

3. No Follower Host CPU Involvement

MINOS-O Features



MINOS-B <Lin, Synch> Algorithm



MINOS-O <Lin, Synch> Algorithm

4. Metadata Coherence

Methodology

Simulated System

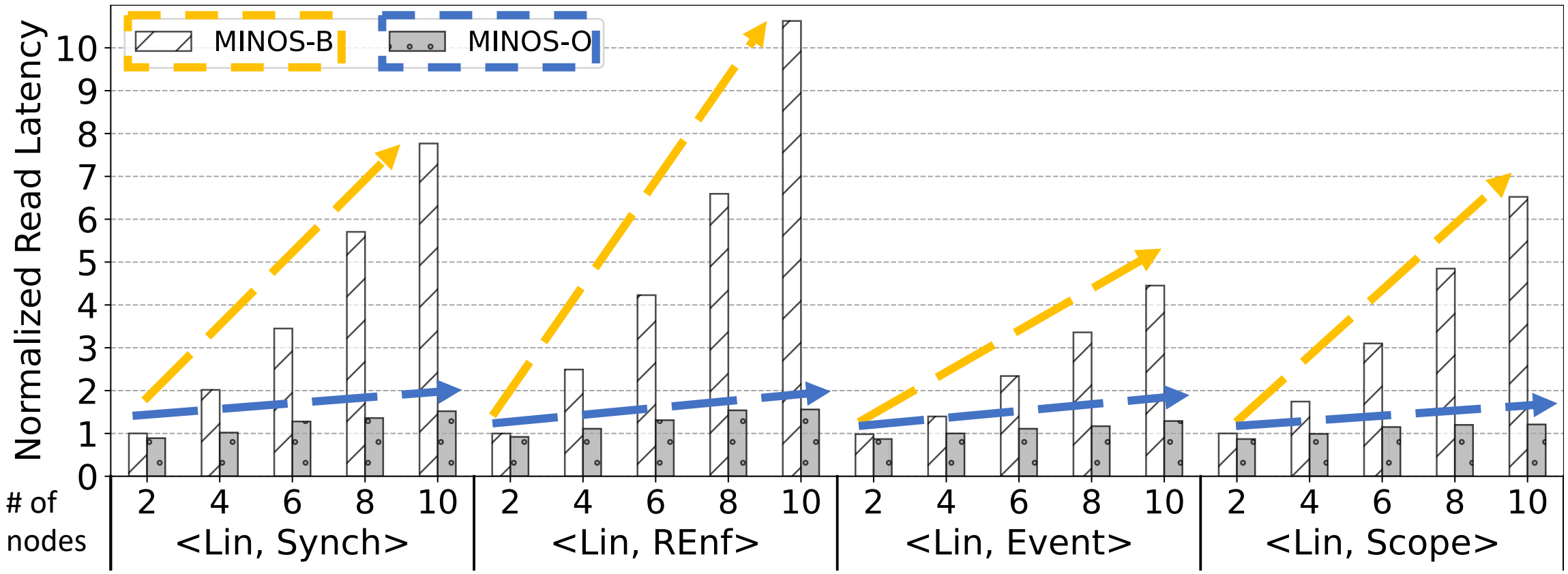
- 2-16 nodes, 5 cores each (2.1 GHz)
- SmartNIC with 8 cores at 2GHz

Benchmarks

- Microbenchmark: YCSB
- Macrobenchmark: DeathStar Benchmark*

* Gan et al., “An Open-Source Benchmark Suite for Microservices and Their Hardware-Software Implications for Cloud & Edge Systems”, ASPLOS19

Evaluation: Read Scalability



MINOS-O reduces read latency by 3.1x on average

Conclusion

- **MINOS-Baseline (MINOS-B):** set of new algorithms for efficient implementation of leaderless consistency and persistency models
- **MINOS-Offload (MINOS-O)**: offloads MINOS-B algorithms to SmartNICs
  - 2.7x average latency reduction over MINOS-B
  - 2.4x average throughput increase over MINOS-B
  - End-to-end microservice average latency: 35% reduction over MINOS-B

30th IEEE International Symposium on High-Performance Computer Architecture (HPCA 2024)

"MINOS: Distributed Consistency and Persistency Protocol Implementation & Offloading to SmartNICs"

Antonis Psistakis, Fabien Chaix❖, and Josep Torrellas
University of Illinois Urbana-Champaign, USA
❖FORTH, Greece
{psistaki, torrella}@illinois.edu, chaix@ics.forth.gr

Thank you!

Scan to read our paper

Edinburgh, Scotland
March 5th, 2024

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

i-acoma group