Dorado: Clustered Hardware Cache Coherence for 1,000+ Cores

Jovan Stojkovic, Abraham Farrell, Gerasimos Gerogiannis,
Zhangxiaowen Gong, Christopher J. Hughes, Josep Torrellas

University of Illinois Urbana-Champaign + Intel

ISCA, June 2026

i-acoma group

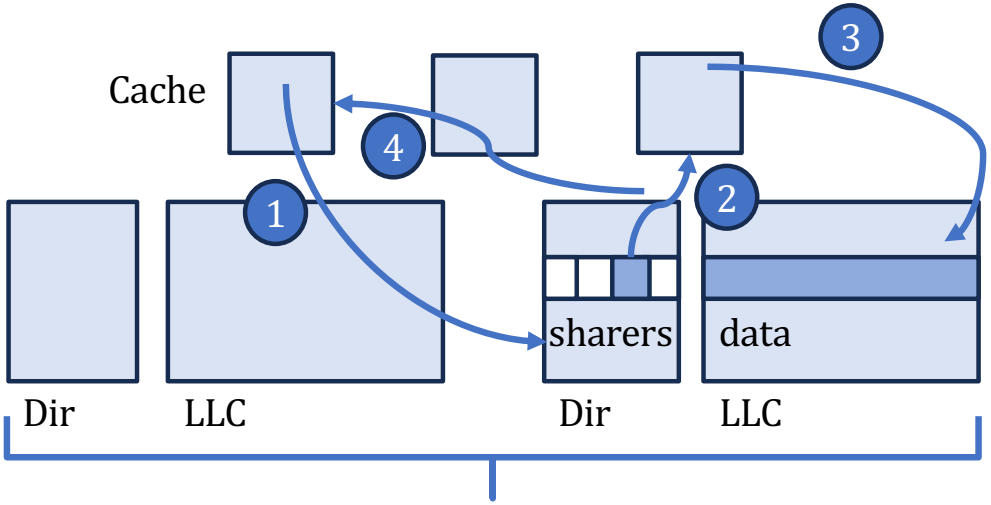
ILLINOIS

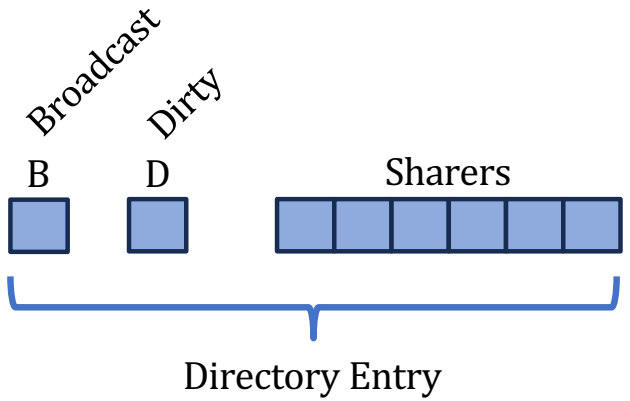
Introduction

- Datacenter processors continue to grow in core count
  - 192 in AMD EPYC 9965, 288 in Intel Clearwater Forest, 512 in Ampere Aurora
- These processors support directory-based hardware cache coherence
  - Applications: graph analytics, in-memory key-value stores, scientific computations, ML inference…
- We may see processors scale to 1000+ cores

Directory-Based Coherence Protocols

- Directory is distributed in slices, and each memory line has a home directory slice
- Main sources of overhead for large-scale directory protocols:
  - Long latency and traffic induced by coherence transactions
  - Directory storage required to track data sharers





How to Address these Problems?

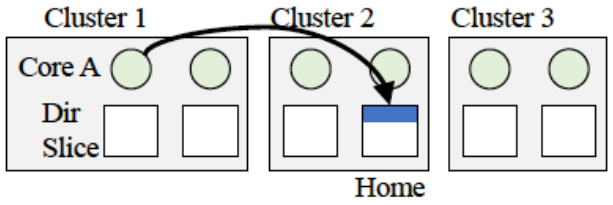
- Latency/traffic: enable clusters of cores, so that cores mostly operate on shared lines locally, without requiring remote accesses

- Storage: Most lines are not widely shared, so use few pointers per entry
  - But still need to support “many-sharer” lines

Contributions

Dorado: New directory-based protocol for 1,000+ cores that exploits clusters

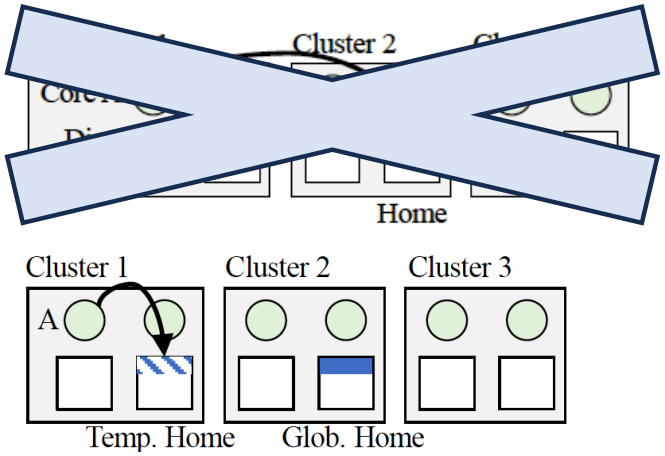
- Two-Level Homes (TLH): Each line has a Global Home and multiple Temporary homes to satisfy many transactions locally within a cluster



Contributions

Dorado: New directory-based protocol for 1,000+ cores that exploits clusters

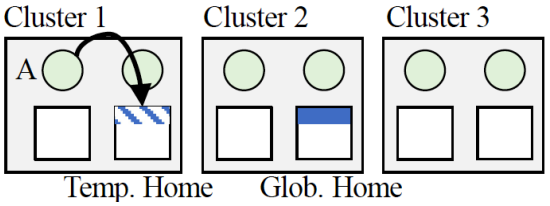
- Two-Level Homes (TLH): Each line has a Global Home and multiple Temporary homes to satisfy many transactions locally within a cluster

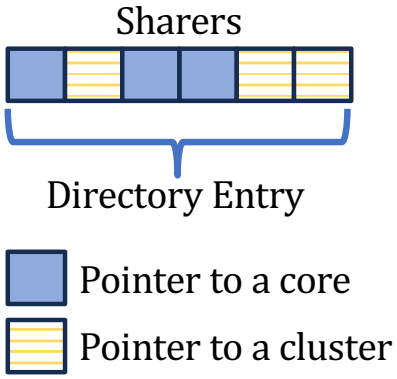


Contributions

Dorado: New directory-based protocol for 1,000+ cores that exploits clusters

- Two-Level Homes (TLH): Each line has a Global and multiple Temporary homes to satisfy many transactions locally within a cluster
- Dynamic Apportioning: The space of a directory entry is shared between pointers of different types, dynamically

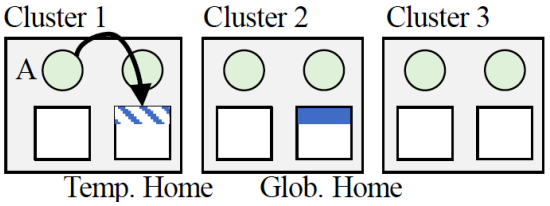


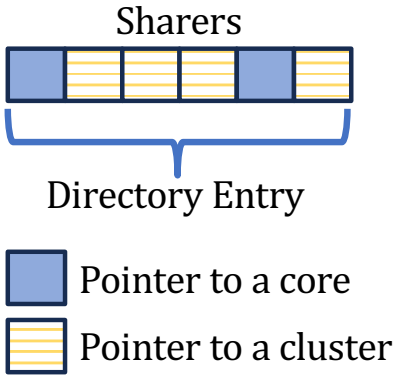


Contributions

Dorado: New directory-based protocol for 1,000+ cores that exploits clusters

- Two-Level Homes (TLH): Each line has a Global and multiple Temporary homes to satisfy many transactions locally within a cluster
- Dynamic Apportioning: The space of a directory entry is shared between pointers of different types, dynamically

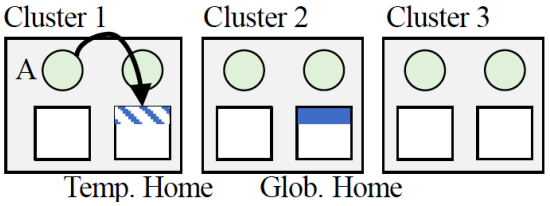


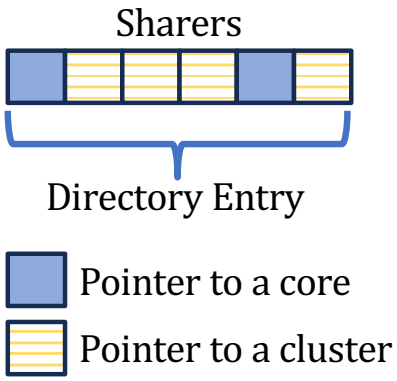


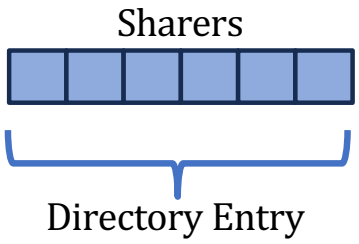
Contributions

Dorado: New directory-based protocol for 1,000+ cores that exploits clusters

- Two-Level Homes (TLH): Each line has a Global and multiple Temporary homes to satisfy many transactions locally within a cluster
- Dynamic Apportioning: The space of a directory entry is shared between pointers of different types, dynamically
- SetOverflow: The directory entry of a "many-sharer" line grows by dynamically grabbing/returning pointers within a group of directory entries



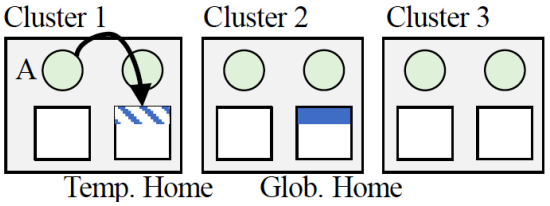


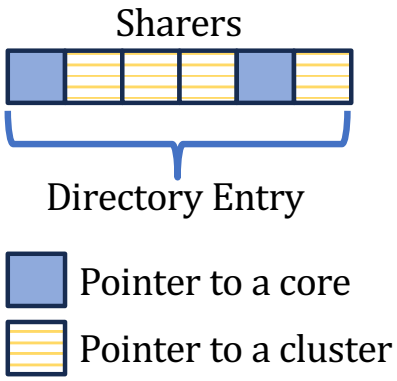


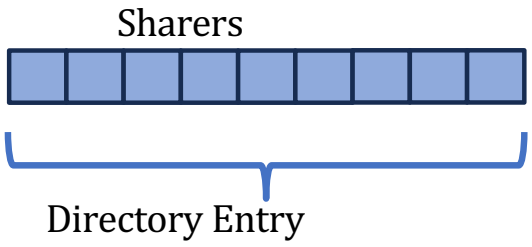
Contributions

Dorado: New directory-based protocol for 1,000+ cores that exploits clusters

- Two-Level Homes (TLH): Each line has a Global and multiple Temporary homes to satisfy many transactions locally within a cluster
- Dynamic Apportioning: The space of a directory entry is shared between pointers of different types, dynamically
- SetOverflow: The directory entry of a "many-sharer" line grows by dynamically grabbing/releasing pointers within a directory set



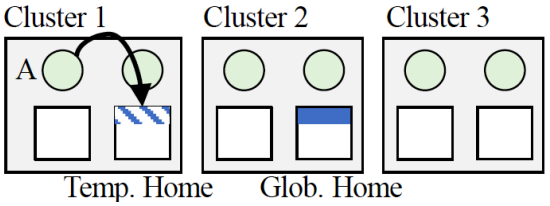


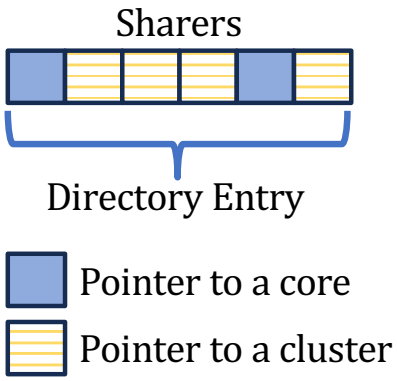


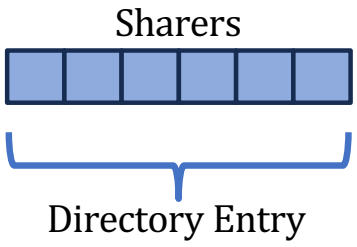
Contributions

Dorado: New directory-based protocol for 1,000+ cores that exploits clusters

- Two-Level Homes (TLH): Each line has a Global and multiple Temporary homes to satisfy many transactions locally within a cluster
- Dynamic Apportioning: The space of a directory entry is shared between pointers of different types, dynamically
- SetOverflow: The directory entry of a "many-sharer" line grows by dynamically grabbing/releasing additional shared-pointers within a directory set

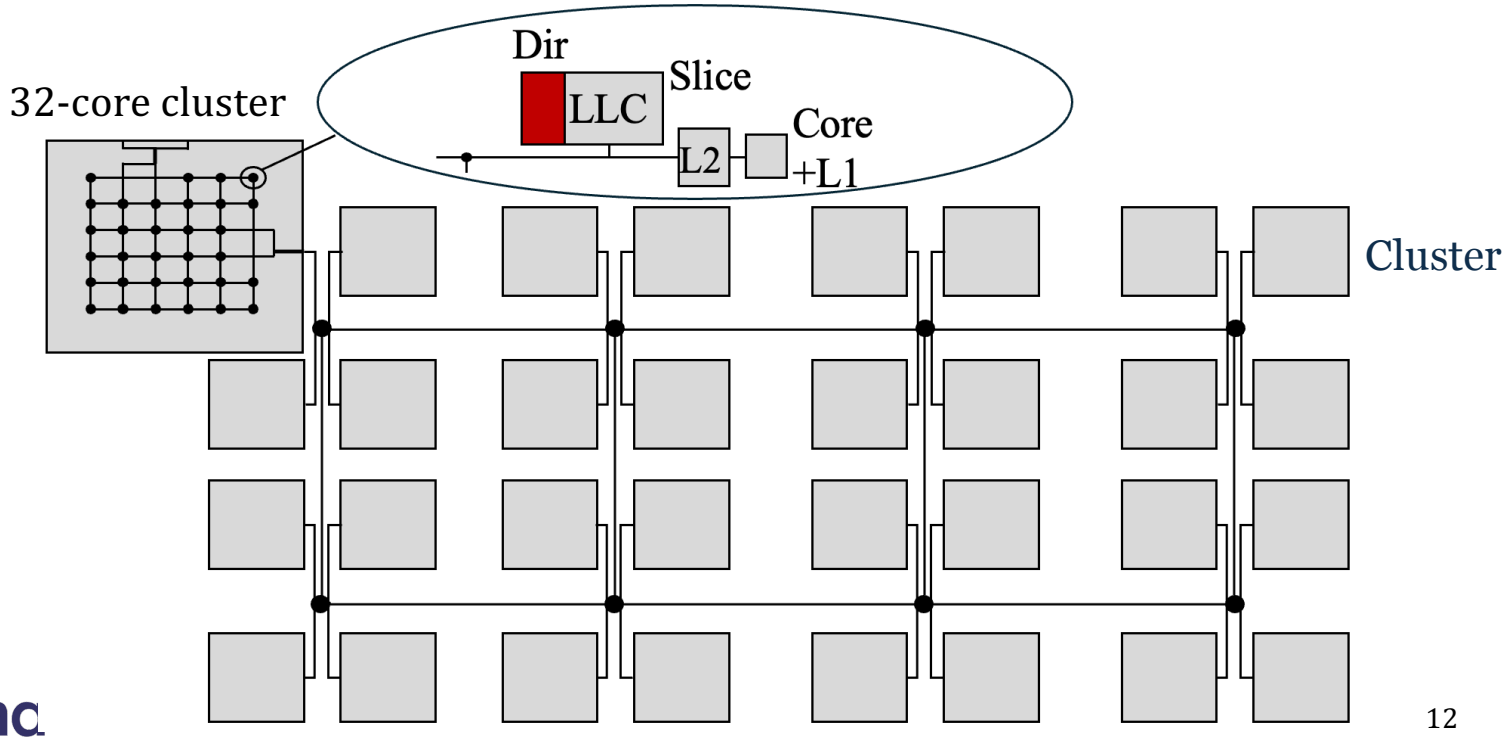






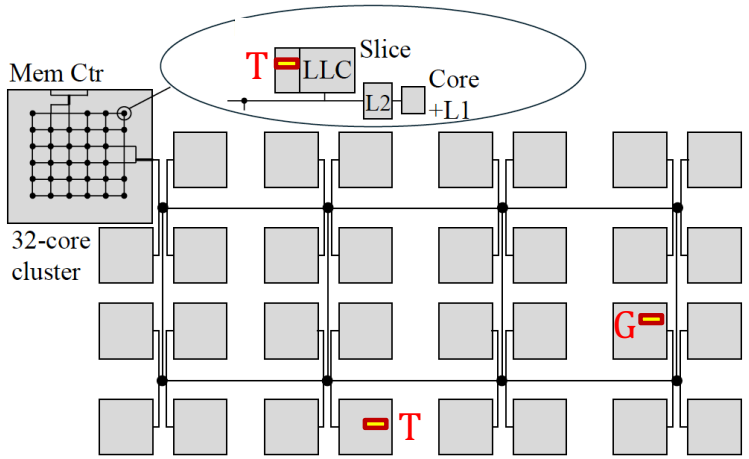
Example Machine Architecture

Server with 32 clusters of 32 cores each. Total: 1,024 cores and 1,024 directory slices



Two-Level Homes (TLH): Concept

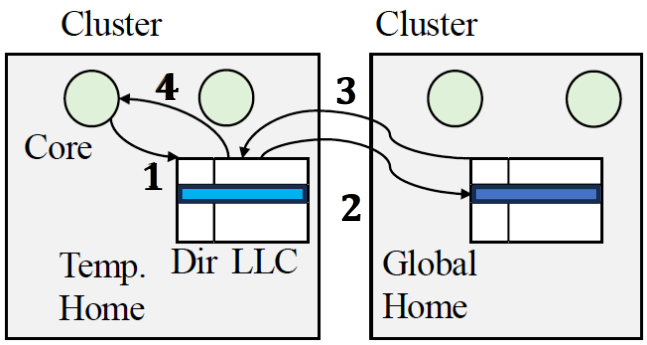
- The physical address of a line determines its Global Home cluster
- When a core accesses a line whose Global home is remote, Dorado creates a Temporary Home in the local cluster
- The entry remains in a local directory slice for as long as it's not evicted
- Directory entries of a line in the Global home and Temporary homes are consistent

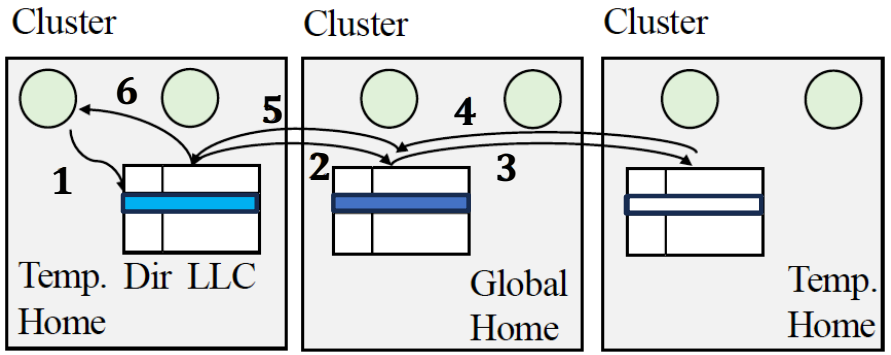


i-acoma group

Two-Level Homes (TLH): Operation

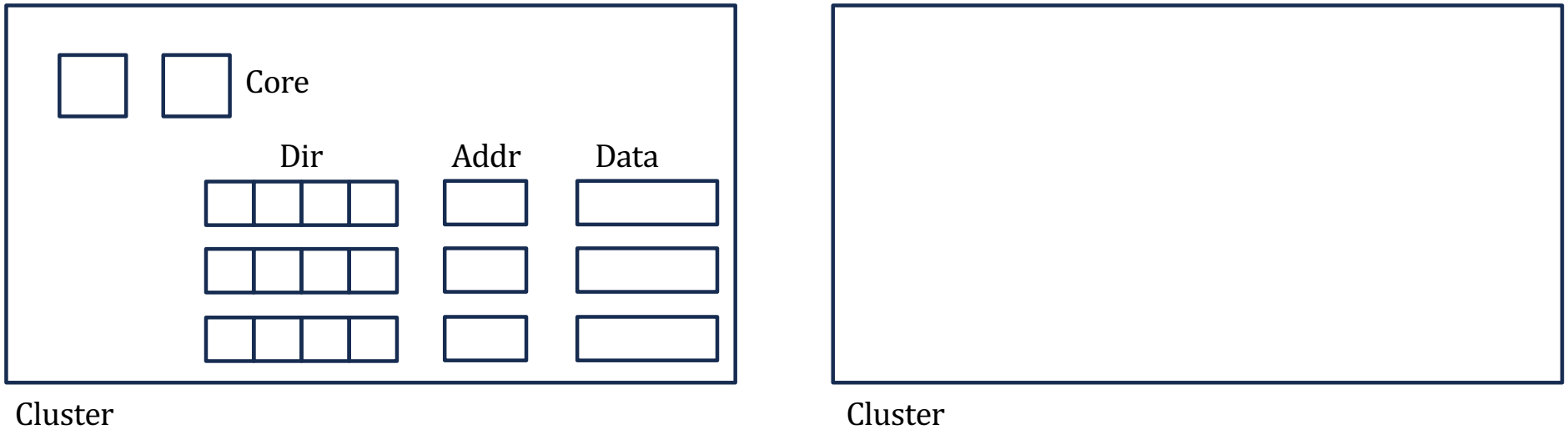
- Temporary homes reduce access latency and bandwidth
- The Global home is the serialization point of coherence transactions to the line → eliminates inconsistency





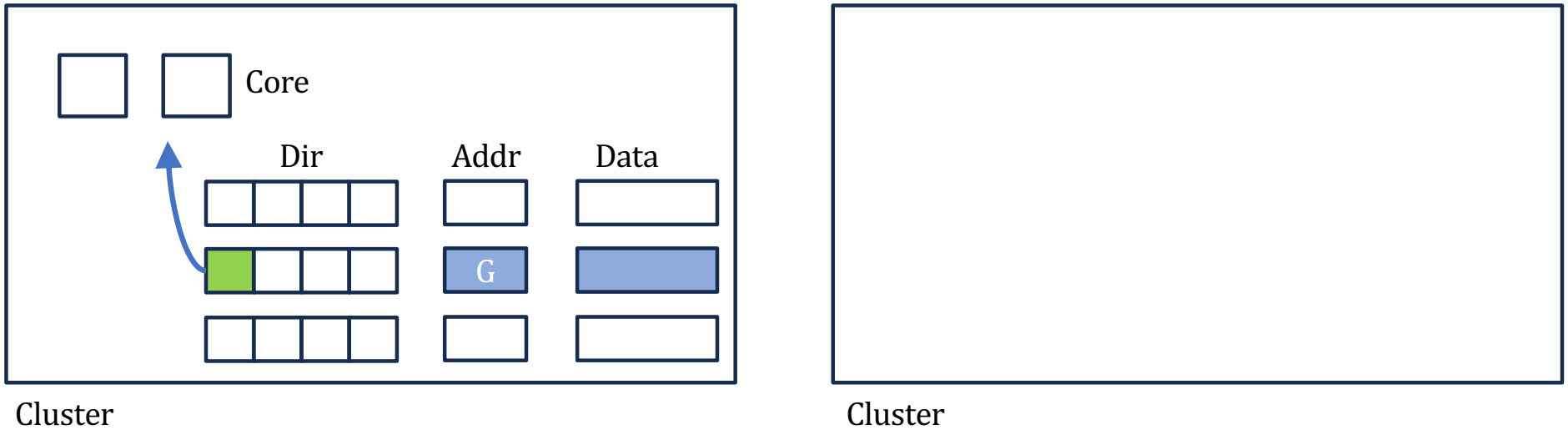
- A transaction T that may change the state of the Global home cannot lock any resource in a Temporary home until T has reached and locked the Global home → No deadlock

Dynamic Apportioning



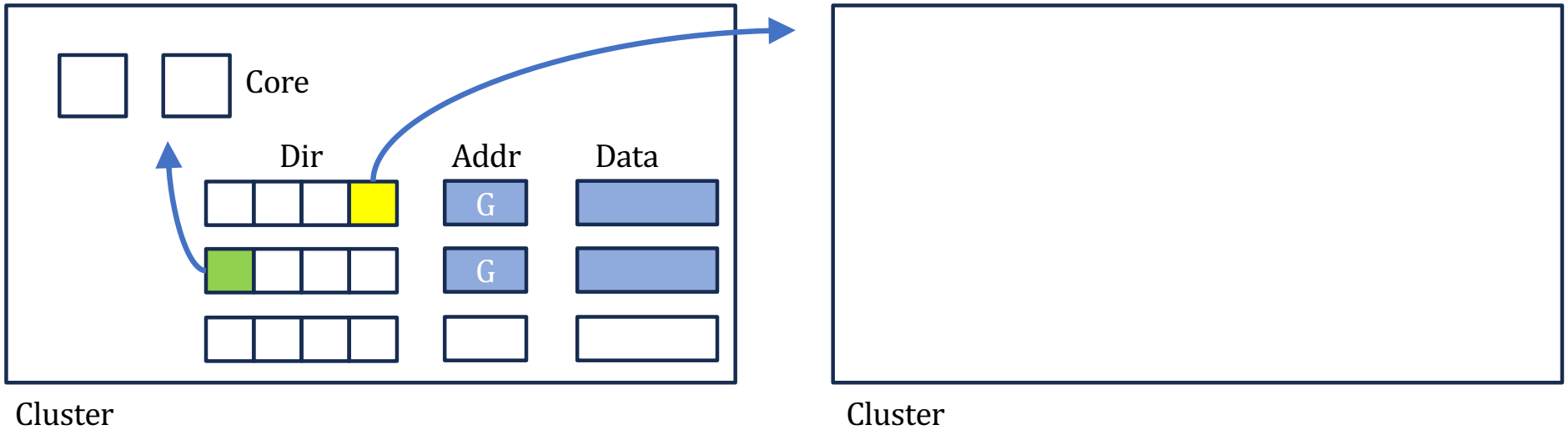
Pointer Type	Contents	Type of Home

Dynamic Apportioning



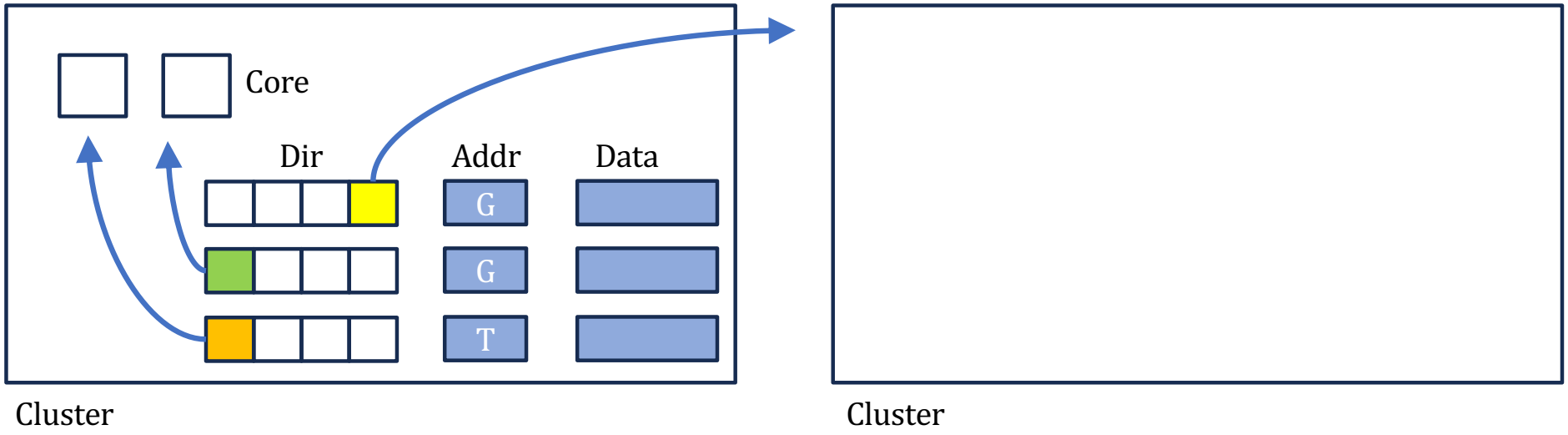
Pointer Type	Contents	Type of Home
1. LocalHome Data & Local Sharer (LLptr)	Core ID	Global

Dynamic Apportioning



Pointer Type	Contents	Type of Home
1. LocalHome Data & Local Sharer (LLptr)	Core ID	Global
2. LocalHome Data & Remote Sharer (LRptr)	Cluster ID	Global

Dynamic Apportioning

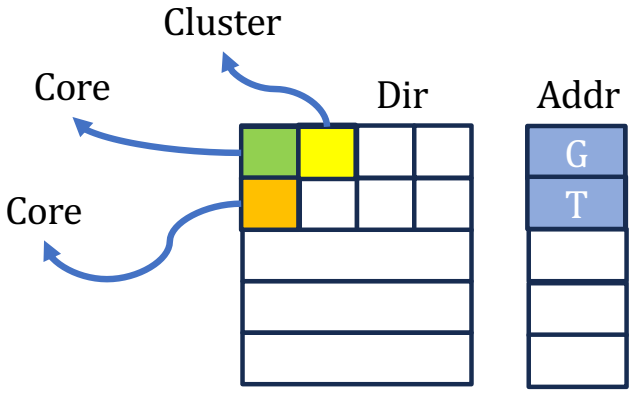


Pointer Type	Contents	Type of Home
1. LocalHome Data & Local Sharer (LLptr)	Core ID	Global
2. LocalHome Data & Remote Sharer (LRptr)	Cluster ID	Global
3. RemoteHome Data & Local Sharer (RLptr)	Core ID	Temporary

Dynamic Apportioning

Pointer Type	Contents	Type of Home
1. LocalHome Data & Local Sharer (LLptr)	Core ID	Global
2. LocalHome Data & Remote Sharer (LRptr)	Cluster ID	Global
3. RemoteHome Data & Local Sharer (RLptr)	Core ID	Temporary

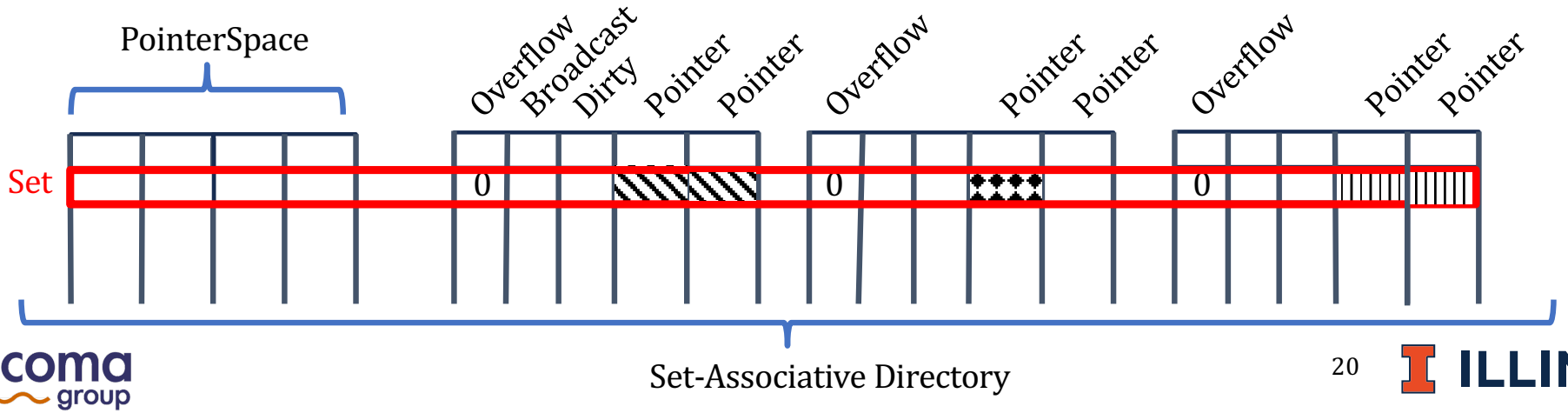
- A Global directory entry can have both Pointers of type 1 and 2
- A directory slice can have both Global and Temporary directory entries
- Dynamic Apportioning saves space by dynamically sharing HW based on application needs



SetOverflow: Efficient "Many-Sharer" Directory Entries

Goal: Dynamically support many sharers for some directory entries

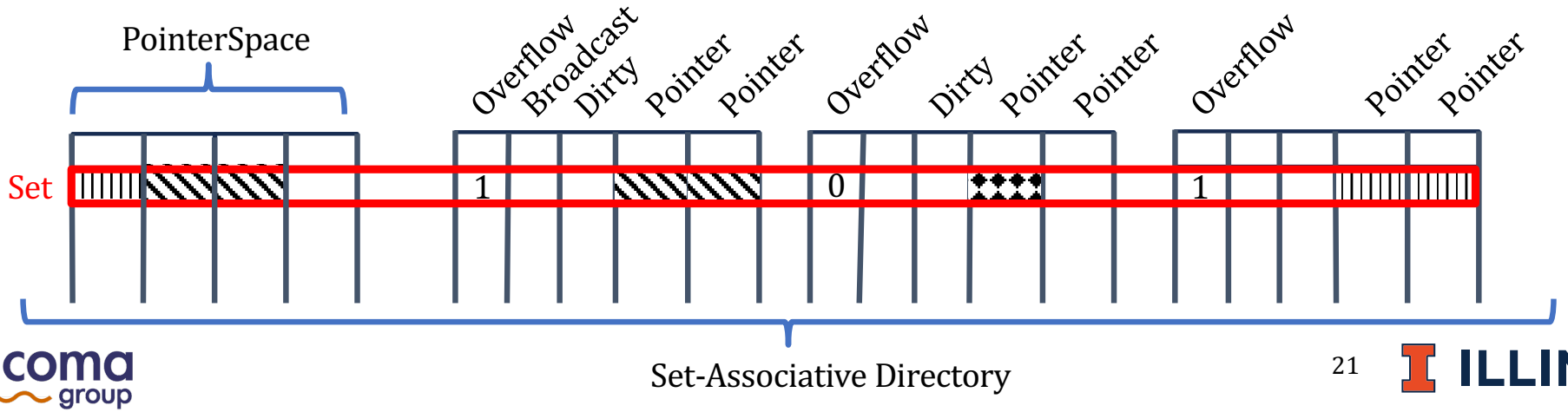
- Keep a group of additional sharer pointers per directory set (PointerSpace)
- When one way runs out of pointers: set O bit and take a pointer from PointerSpace



SetOverflow: Efficient "Many-Sharer" Directory Entries

Goal: Dynamically support many sharers for a few directory entries

- Keep a group of additional sharer pointers per directory set (PointerSpace)
- When one way runs out of pointers: set O bit and take a pointer from PointerSpace
- Fine-grain re-assignment of the PointerSpace pointers to entries in the set



Operation of the Dorado Scalable Directory Protocol

CORE READ MISS IN L2 AND LINE NOT MOD

A1: Local line and no local dir entry	Get line from local DRAM. Create dir LLptr and D=0.
A2: Remote line and no local dir entry	Access the Global home and do: {If dir e get lir add I Creat
A3: Local dir entry with any combination of LLptrs and LRptrs	Get li remot
A4: Local dir entry with RLptrs	Get li or on RLptr

CORE READ MISS IN

B1: Remote line and no local dir entry	Access t D=1, and in the Gl (S). Set cluster, if sharer in to memo entry. Ad owners i
B2: Owner is LLptr in local dir entry	Get line cache an add LLpt
B3: Owner is LRptr in local dir entry	In the c cache(s) line to th In local c
B4: Owner is RLptr in local dir entry	Get line RLptr an change t

CORE WRITE MISS IN ITS L2 AND LINE NOT MODIFIED ANYWHERE, OR WRITE HIT IN ITS L1/L2 ON A SHARED (S) LINE.

C1: Local line and no local dir entry	Get line from local DRAM. Create dir entry locally. Add LLptr and D=1.
C2: Remote line and no local dir entry	Access and do the following in the Global home cluster: {If dir entry does not exist, get line from DRAM, create dir entry, and add LRptr and D=1. Otherwise, do: 1)Invalidate all sharers in the Global home cluster and in third clusters,
C3: Local dir entr with all LLptrs	
C4: Local dir entr with all LRptrs	
C5: Local dir entr with a combinatio of LLptrs, LRptrs	
C6: Local dir entr with RLptrs	

CORE WRITE MISS IN L2 AND LINE MODIFIED SOMEWHERE.

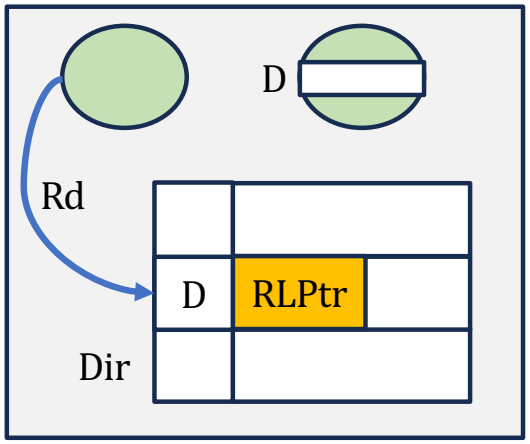
D1: Remote line and no local dir entry	Access the Global home, where the dir has an entry with D=1, and do: {Write back the line and invalidate the owner cache (which is either in the Global home cluster or in a third cluster). If owner was in a third cluster, remove dir entry in the third cluster. In the Global home, update dir entry to have only the requester's LRptr and keep D=1}. Bring the line to local cluster. Create local dir entry. Add RLptr and D=1. If the third cluster had multiple owners in MS state, all of them are invalidated.
D2: Owner is LLptr in local dir entry	Get line from the owner's L2. Invalidate owner cache. Update local dir entry to have only the requester's LLptr and keep D=1.
D3: Owner is LRptr in local dir entry	In the owner remote cluster, do: {1)Get the line from an owner, 2)Invalidate the owner(s), and 3)Remove the dir entry}. Bring the line to local cluster. Update local dir entry to have only the requester's LLptr and keep D=1.
D4: Owner(s) is/are RLptr in local dir entry	Get line from an owner's local L2. Invalidate the owner(s). Update local dir entry to have only the requester's RLptr and keep D=1.

i-acoma group

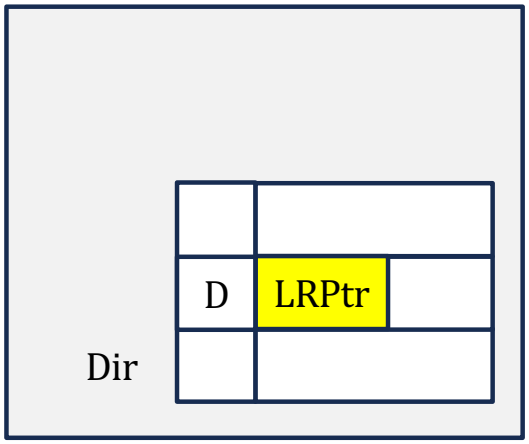
S

Operation of the Dorado Scalable Directory Protocol

Read miss to modified data where the local cluster has a Temporary home



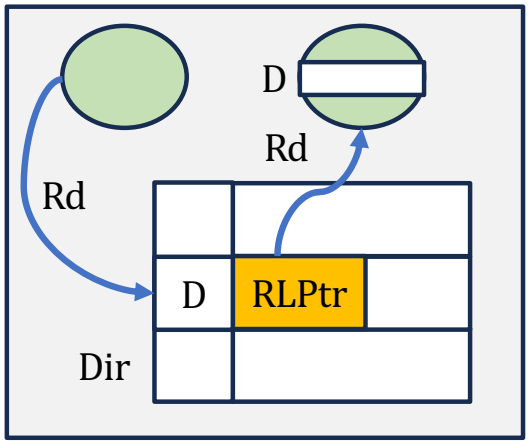
Local Cluster



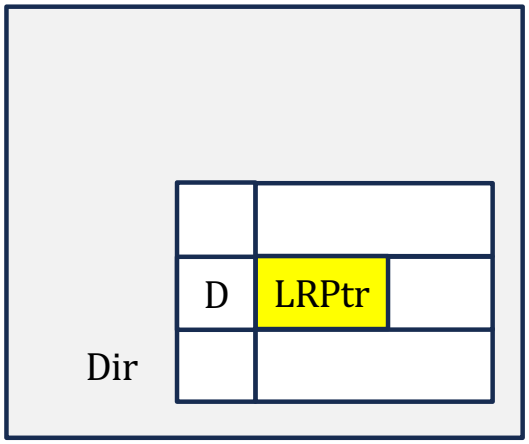
Global Home Cluster

Operation of the Dorado Scalable Directory Protocol

Read miss to modified data where the local cluster has a Temporary home



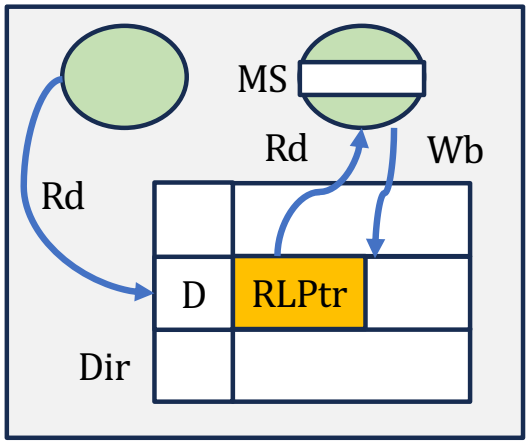
Local Cluster



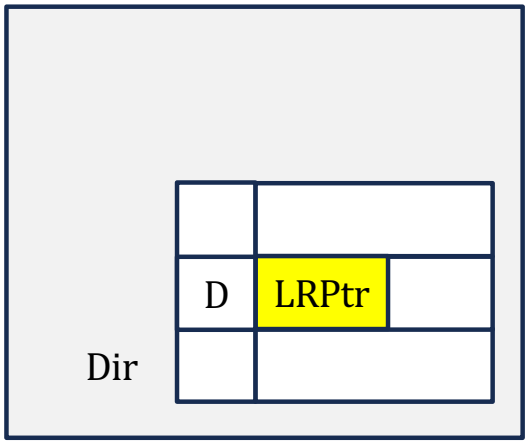
Global Home Cluster

Operation of the Dorado Scalable Directory Protocol

Read miss to modified data where the local cluster has a Temporary home



Local Cluster



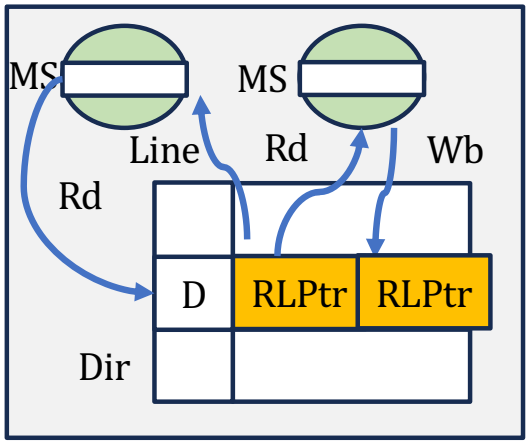
Global Home Cluster

MS: ModifiedShared state

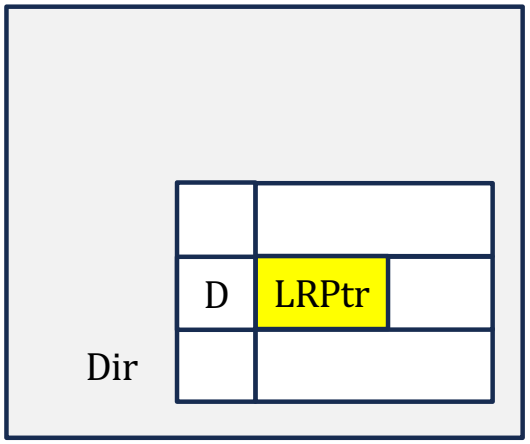
Operation of the Dorado Scalable Directory Protocol

Read miss to modified data where the local cluster has a Temporary home

No need to inform the Global directory when local caches Wr/Rd data in State D or MS



Local Cluster



Global Home Cluster

MS: ModifiedShared state

Evaluation Setup

Architecture

System	1024 cores at 3 GHz
Organization	32 clusters of 32 cores each
Cores	6-issue out-of-order, 352-entry ROB, TSO memory consistency

Workloads

Graphs from GraphBIG	BFS, DFS, CC, PR
Informatics	MUMmer
Data Analytics	MapRed
Key-value Stores	Redis-read, Redis-write
ML Serving from MLPerf Inference	DLRM, CNN
FaaS from FunctionBench	Image processing, Video processing
Microservices from DeathStarBench	SocNet

Speed-up of Protocols (for Same-area Directory)

Dir2B	Vanilla flat (i.e., not clustered) directory protocol	2 ptrs/dir entry (10b each)

Speed-up of Protocols (for Same-area Directory)

Dir2B	Vanilla flat (i.e., not clustered) directory protocol	2 ptrs/dir entry (10b each)
TLH-Dir4B	Dir2B + TLH (reduces ptr size due to clustering)	4 ptrs/dir entry (5b each)

Speed-up of Protocols (for Same-area Directory)

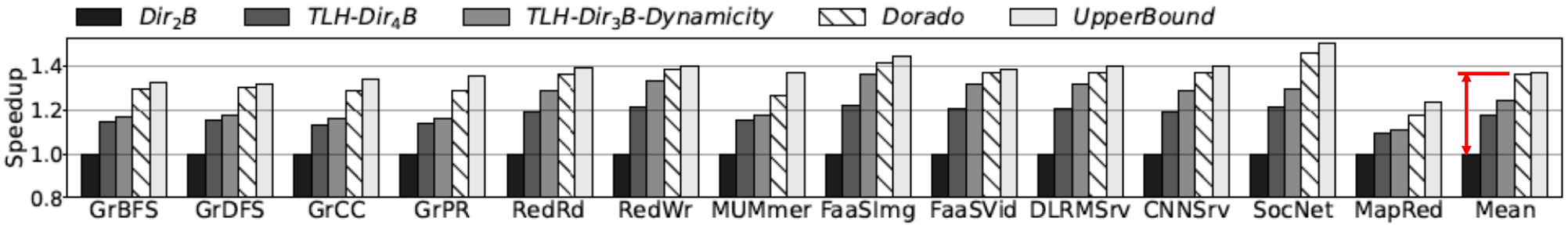
Dir2B	Vanilla flat (i.e., not clustered) directory protocol	2 ptrs/dir entry (10b each)
TLH-Dir4B	Dir2B + TLH (reduces ptr size due to clustering)	4 ptrs/dir entry (5b each)
TLH-Dir3B-Dynamicity	TLH-Dir4B + Dynamic Apportioning	3 ptrs/dir entry (6b each)

Speed-up of Protocols (for Same-area Directory)

Dir2B	Vanilla flat (i.e., not clustered) directory protocol	2 ptrs/dir entry (10b each)
TLH-Dir4B	Dir2B + TLH (reduces ptr size due to clustering)	4 ptrs/dir entry (5b each)
TLH-Dir3B-Dynamicity	TLH-Dir4B + Dynamic Apportioning	3 ptrs/dir entry (6b each)
Dorado	TLH-Dir3B-Dynamicity + SetOverflow	2 ptrs/entry (6b each) + 12 PointerSpace ptrs/set

Speed-up of Protocols (for Same-area Directory)

Dir2B	Vanilla flat (i.e., not clustered) directory protocol	2 ptrs/entry (10b each)
TLH-Dir4B	Dir2B + TLH (reduces ptr size due to clustering)	4 ptrs/entry (5b each)
TLH-Dir3B-Dynamicity	TLH-Dir4B + Dynamic Apportioning	3 ptrs/entry (6b each)
Dorado	TLH-Dir3B-Dynamicity + SetOverflow	2 ptrs/entry (6b each) + 12 PointerSpace ptrs/set
UpperBound	TLH-Dir3B-Dynamicity + full bit vector/entry	63 ptrs/entry (1b each): 32 cores +31 clusters



- Each of the three contributions of Dorado is effective on every application
- Dorado achieves avg speedup of 1.36x (within 1% of the perf of UpperBound)

Conclusions

- Dorado: a new cluster-based cache coherence protocol for 1,000+ cores
- Introduces three new techniques:
  - Two-Level Homes, Dynamic Apportioning, and SetOverflow
- Attains an avg speedup of 1.36x over a same-area limited-pointer protocol
- Stays within 1% of the performance of a full bit vector protocol that uses 2.75x more directory storage

Dorado: Clustered Hardware Cache Coherence for 1,000+ Cores

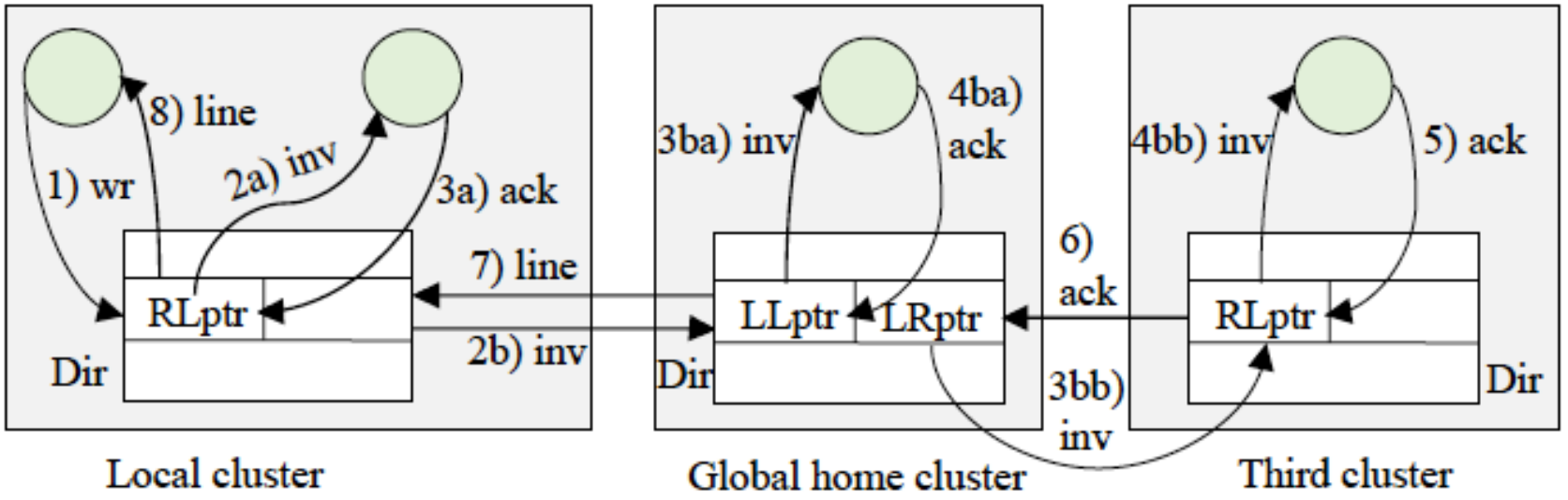
Jovan Stojkovic, Abraham Farrell, Gerasimos Gerogiannis,
Zhangxiaowen Gong, Christopher J. Hughes, Josep Torrellas

University of Illinois Urbana-Champaign + Intel

ISCA, June 2026

Operation of the Dorado Scalable Directory Protocol

Write miss transaction to unmodified data where the local cluster has the Temporary home



Speed-up of “Many-Sharer” Designs (for Same-area Directory)

Dir2B	Vanilla flat directory protocol

Speed-up of “Many-Sharer” Designs (for Same-area Directory)

Dir2B	Vanilla flat directory protocol
TLH-Dir3B-Dynamicity	Dir2B + TLH + Dynamic Apportioning

Speed-up of "Many-Sharer" Designs (for Same-area Directory)

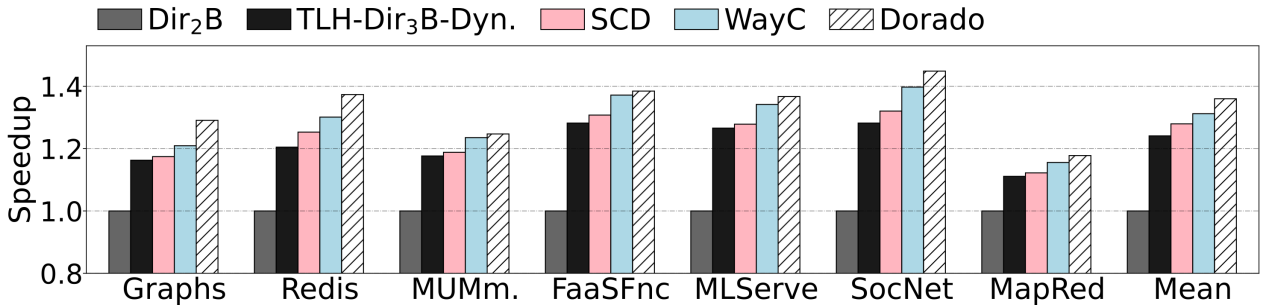
Dir2B	Vanilla flat directory protocol
TLH-Dir3B-Dynamicity	Dir2B + TLH + Dynamic Apportioning
SCD	TLH-Dir3B-Dynamicity + SCD [Sanchez]

Speed-up of "Many-Sharer" Designs (for Same-area Directory)

Dir2B	Vanilla flat directory protocol
TLH-Dir3B-Dynamicity	Dir2B + TLH + Dynamic Apportioning
SCD	TLH-Dir3B-Dynamicity + SCD [Sanchez]
WayC	TLH-Dir3B-Dynamicity + WayCombining [Titos-Gil]

Speed-up of "Many-Sharer" Designs (for Same-area Directory)

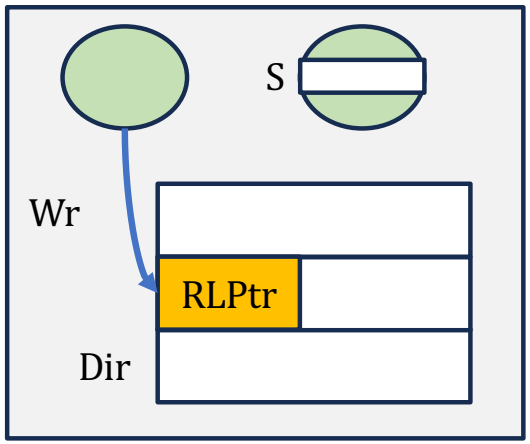
Dir2B	Vanilla flat directory protocol
TLH-Dir3B-Dynamicity	Dir2B + TLH + Dynamic Apportioning
SCD	TLH-Dir3B-Dynamicity + SCD [Sanchez]
WayC	TLH-Dir3B-Dynamicity + WayCombining [Titos-Gil]
Dorado	TLH-Dir3B-Dynamicity + SetOverflow



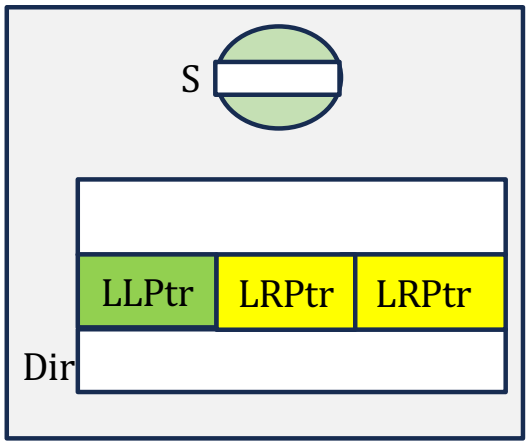
- Modest speed-ups: SCD incurs directory conflicts; WayC is relatively inflexible
- SetOverflow (Dorado): 10.5% speedup over TLH-Dir3B-Dyn thanks to fine-grained assignment of the extra sharer pointers to multiple lines in the set.

Operation of the Dorado Scalable Directory Protocol

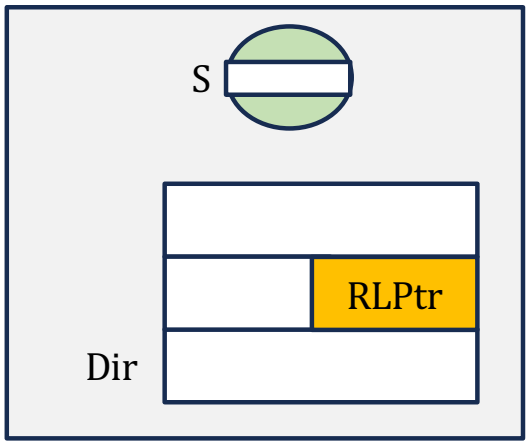
Write miss to unmodified data where the local cluster has a Temporary home



Local Cluster



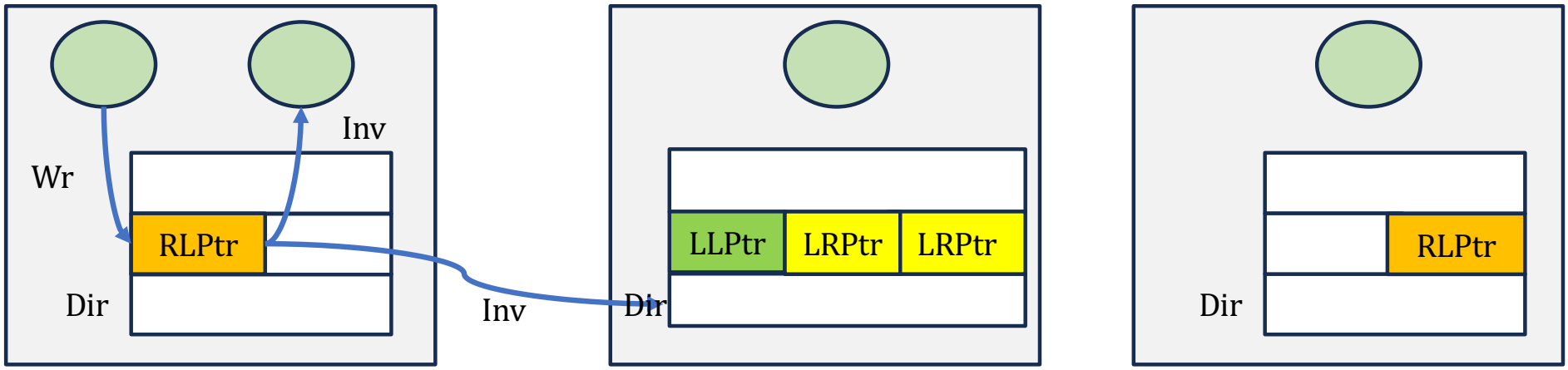
Global Home Cluster



Third Cluster

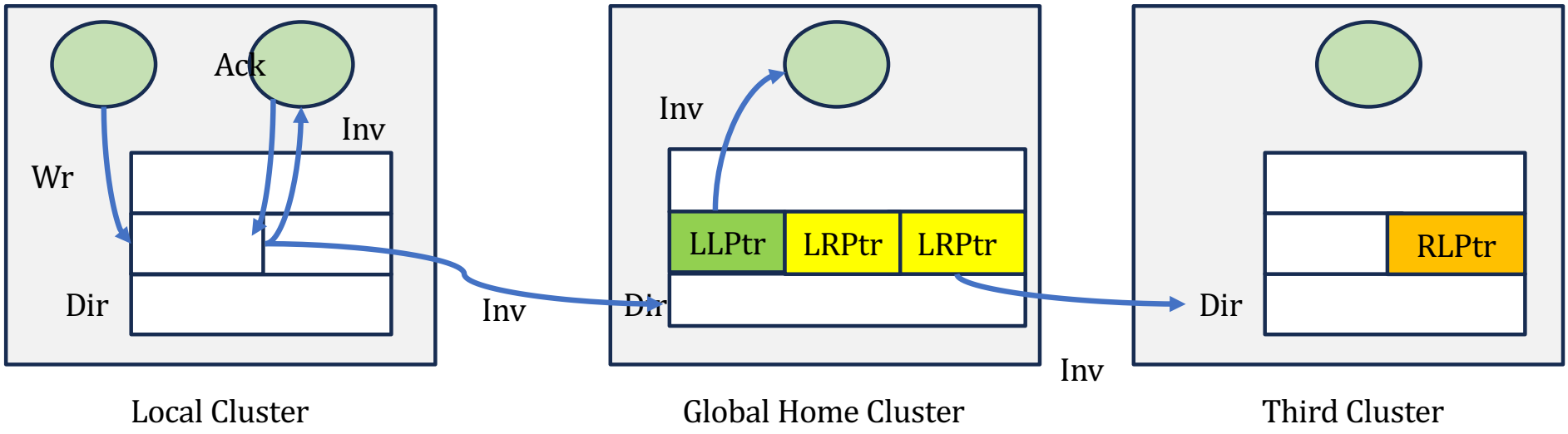
Operation of the Dorado Scalable Directory Protocol

Write miss to unmodified data where the local cluster has a Temporary home



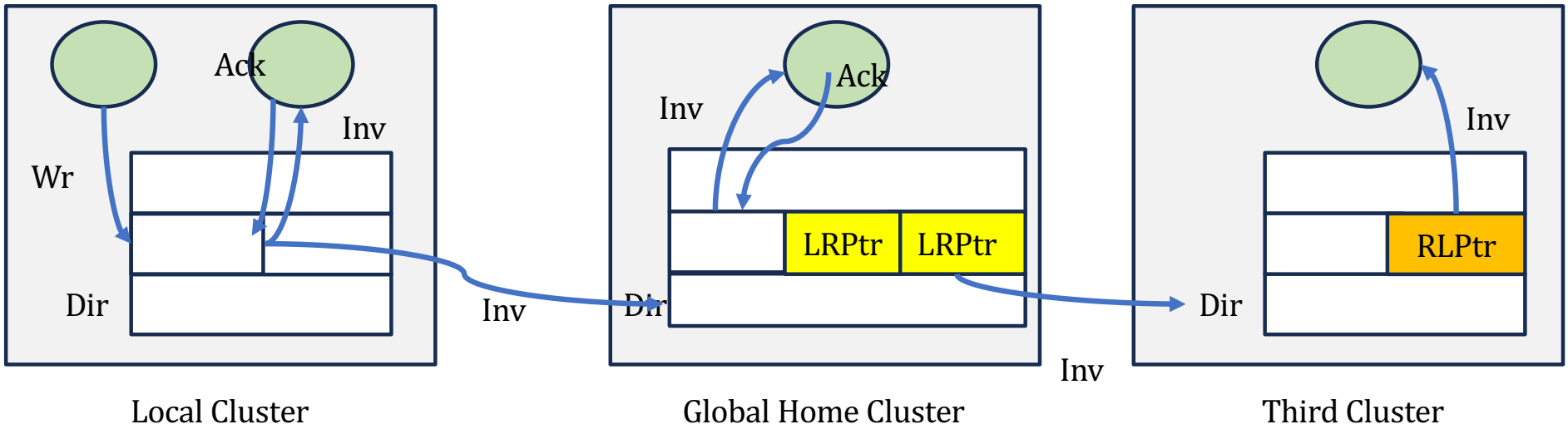
Operation of the Dorado Scalable Directory Protocol

Write miss to unmodified data where the local cluster has a Temporary home



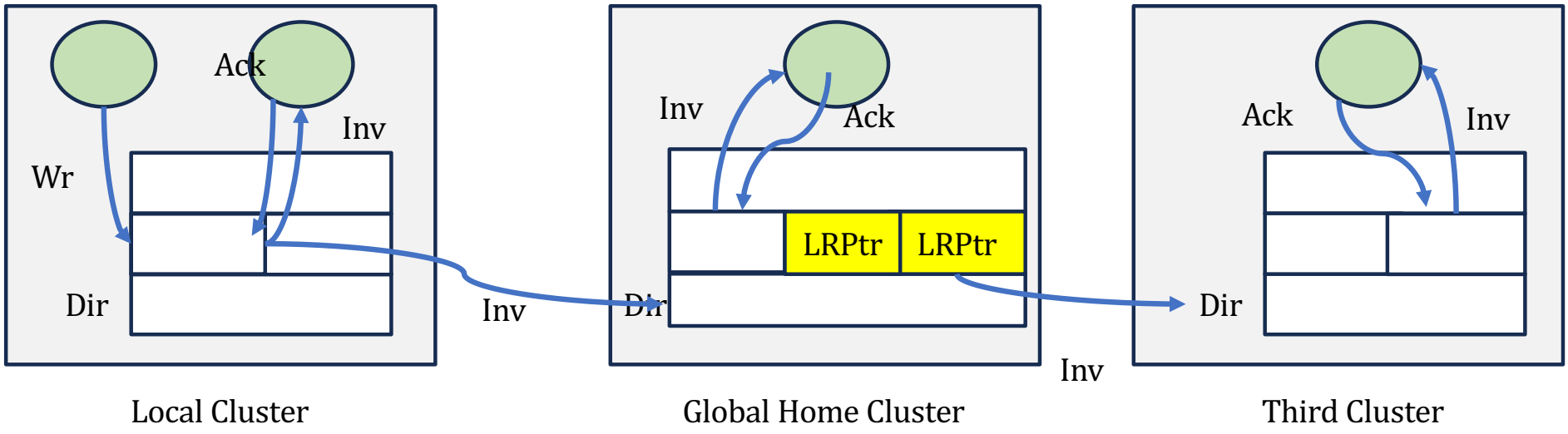
Operation of the Dorado Scalable Directory Protocol

Write miss to unmodified data where the local cluster has a Temporary home



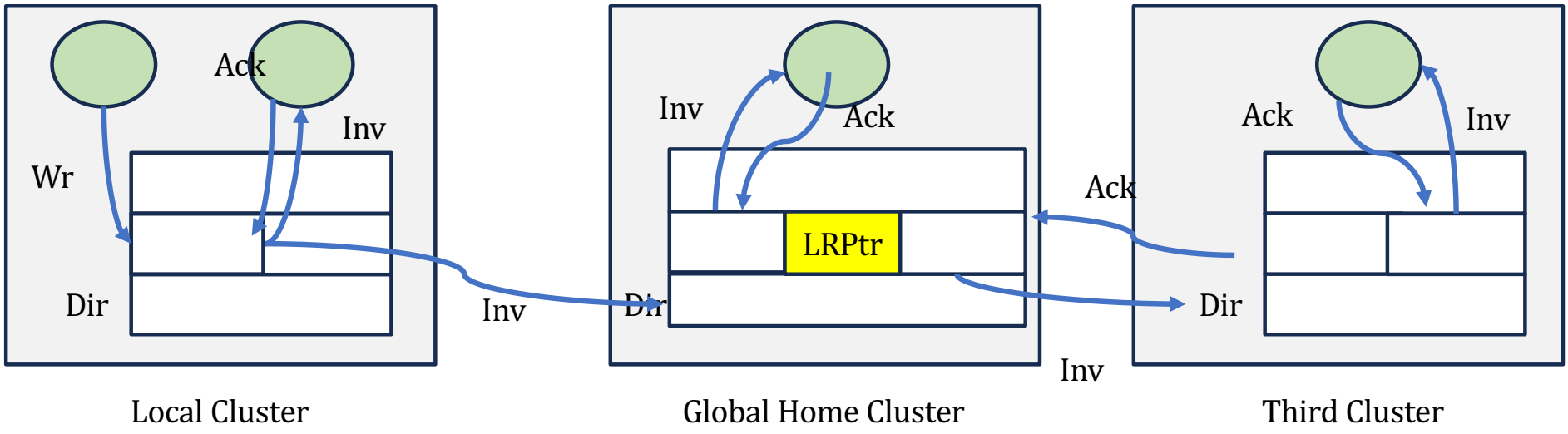
Operation of the Dorado Scalable Directory Protocol

Write miss to unmodified data where the local cluster has a Temporary home



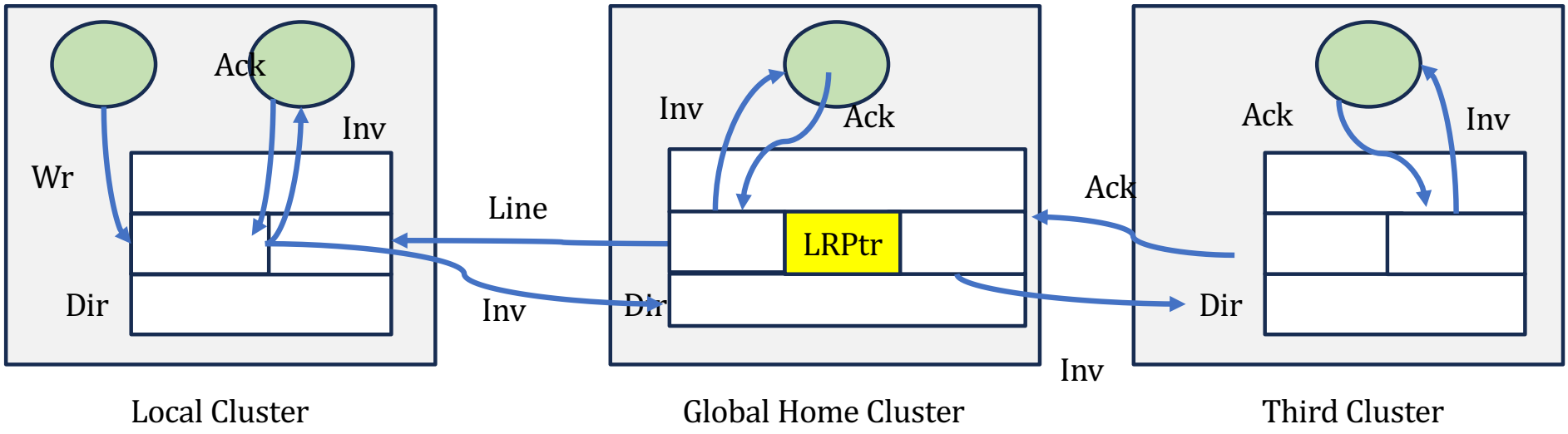
Operation of the Dorado Scalable Directory Protocol

Write miss to unmodified data where the local cluster has a Temporary home



Operation of the Dorado Scalable Directory Protocol

Write miss to unmodified data where the local cluster has a Temporary home



Operation of the Dorado Scalable Directory Protocol

Write miss to unmodified data where the local cluster has a Temporary home