ASPIRE

Exploiting ***As***ynchronous ***P***arallelism in ***I***terative Algorithms using a ***Re***laxed Consistency based DSM

Keval Vora, Sai Charan Koduru, Rajiv Gupta

OOPSLA' 14 – Portland, Oregon
October 24, 2014

UCR
UNIVERSITY OF CALIFORNIA, RIVERSIDE

Motivation

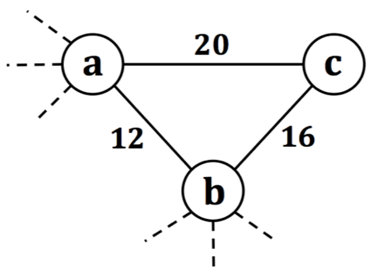
- Iterative algorithms
  - PDE Solvers: Heat Simulation
  - Graph Analysis: PageRank, Community Detection
- Real-world datasets are typically large
  - Social Networks, Genome Graphs
- Processing on distributed memory machines
  - Performance
  - Programmability

Outline

- Iterative Algorithms
- Overview of ASPIRE
- Existing Weak Memory Models
- Relaxed Consistency Protocol
- Evaluation
- Conclusion

Iterative Algorithms

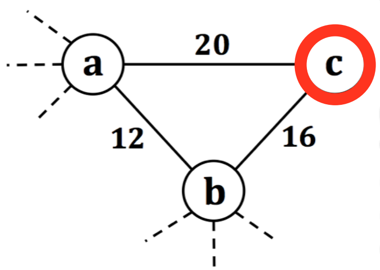
- Data Centric
  - Computation written for a single element
  - Terminate when values converge
  - Highly parallel execution
- Network Bound
  - Computation is simple



Fetch(c)
Fetch(a)
Fetch(b)
c’ = f (c, a, b)
Store(c, c’)

Iterative Algorithms

- Data Centric
  - Computation written for a single element
  - Terminate when values converge
  - Highly parallel execution
- Network Bound
  - Computation is simple



Fetch(c)
Fetch(a)
Fetch(b)
c' = f(c, a, b)
Store(c, c')

Execution Models

- Bulk Synchronous Parallel (BSP) [CACM'90]
  - Disjoint computation and communication
  - Computation based on previous iteration
- Asynchronous Parallelism [Baudet 1978]
  - Overlap computation and communication
  - Computation based on current iteration
  - Known to be faster than BSP

ASPIRE

- Improve asynchronous execution
  - Make them faster
- Tolerate network latencies
  - Tardis: remote fetch is ~2.3 times of local fetch
- Relaxing consistency
  - Allow use of stale values
- Without affecting convergence
  - Minimize use of stale values

ASPIRE

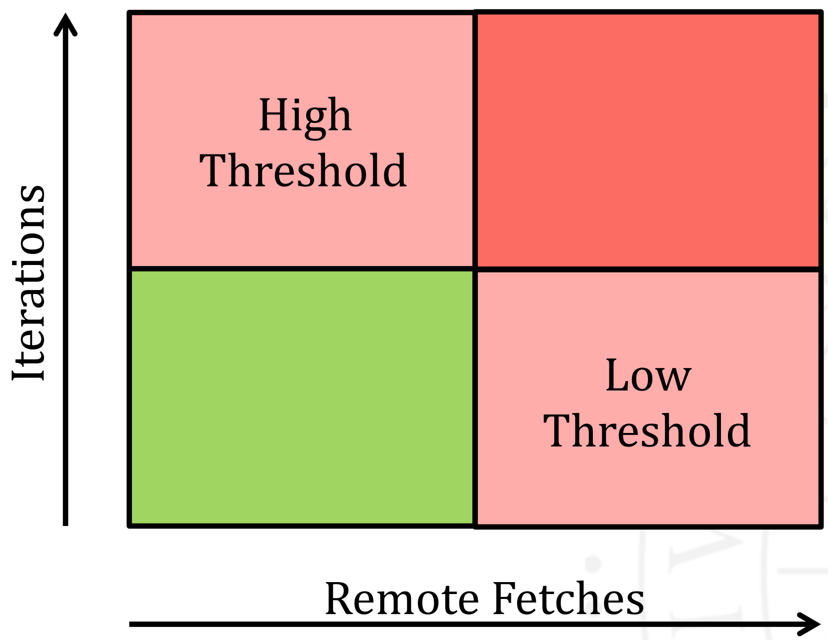
- Improve asynchronous execution
  - Make them faster

Challenge: Relax consistency without delaying convergence

- Relaxing consistency
  - Allow use of stale values
- Without affecting convergence
  - Minimize use of stale values

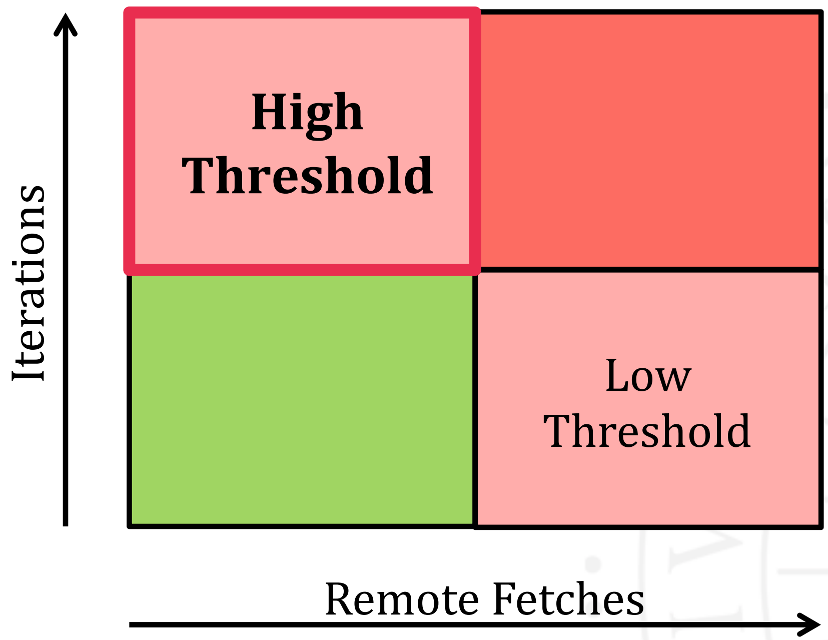
Weak Memory Models

- Delta Consistency [SPAA'97] [PPoPP'03]
  - Controls staleness using static threshold



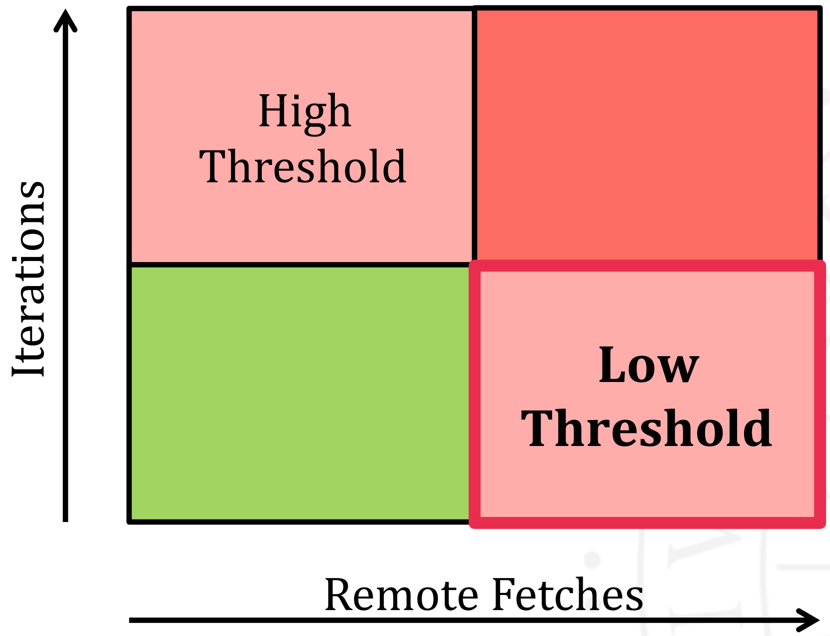
Weak Memory Models

- Delta Consistency [SPAA'97] [PPoPP'03]
  - Controls staleness using static threshold



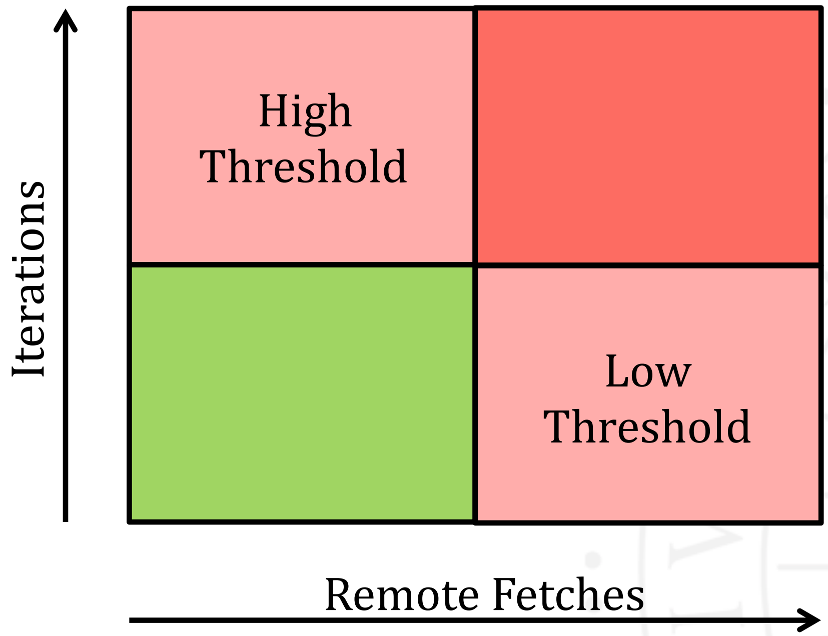
Weak Memory Models

- Delta Consistency [SPAA'97] [PPoPP'03]
  - Controls staleness using static threshold



Weak Memory Models

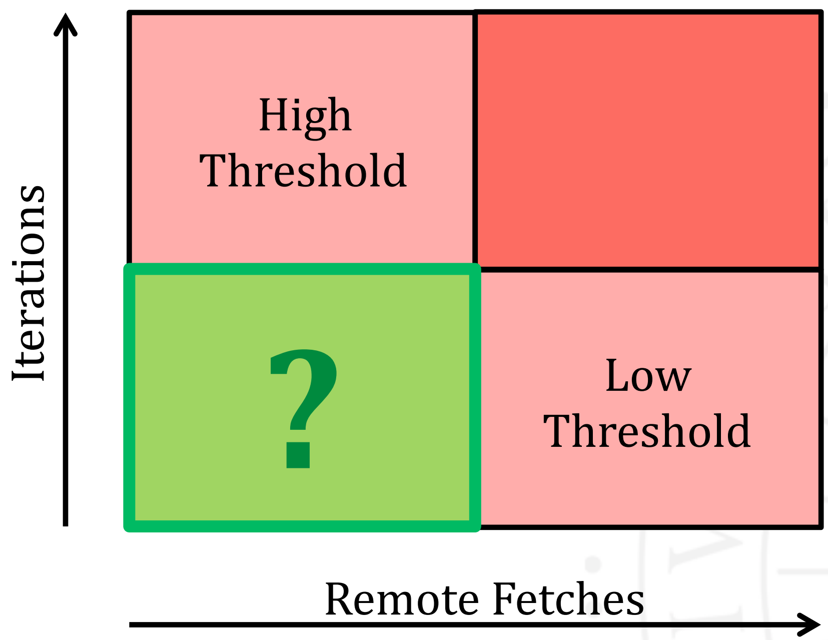
- Delta Consistency [SPAA'97] [PPoPP'03]
  - Controls staleness using static threshold



Delayed updates affect convergence

Weak Memory Models

- Delta Consistency [SPAA'97] [PPoPP'03]
  - Controls staleness using static threshold



Delayed updates affect convergence

Relaxed Consistency Protocol

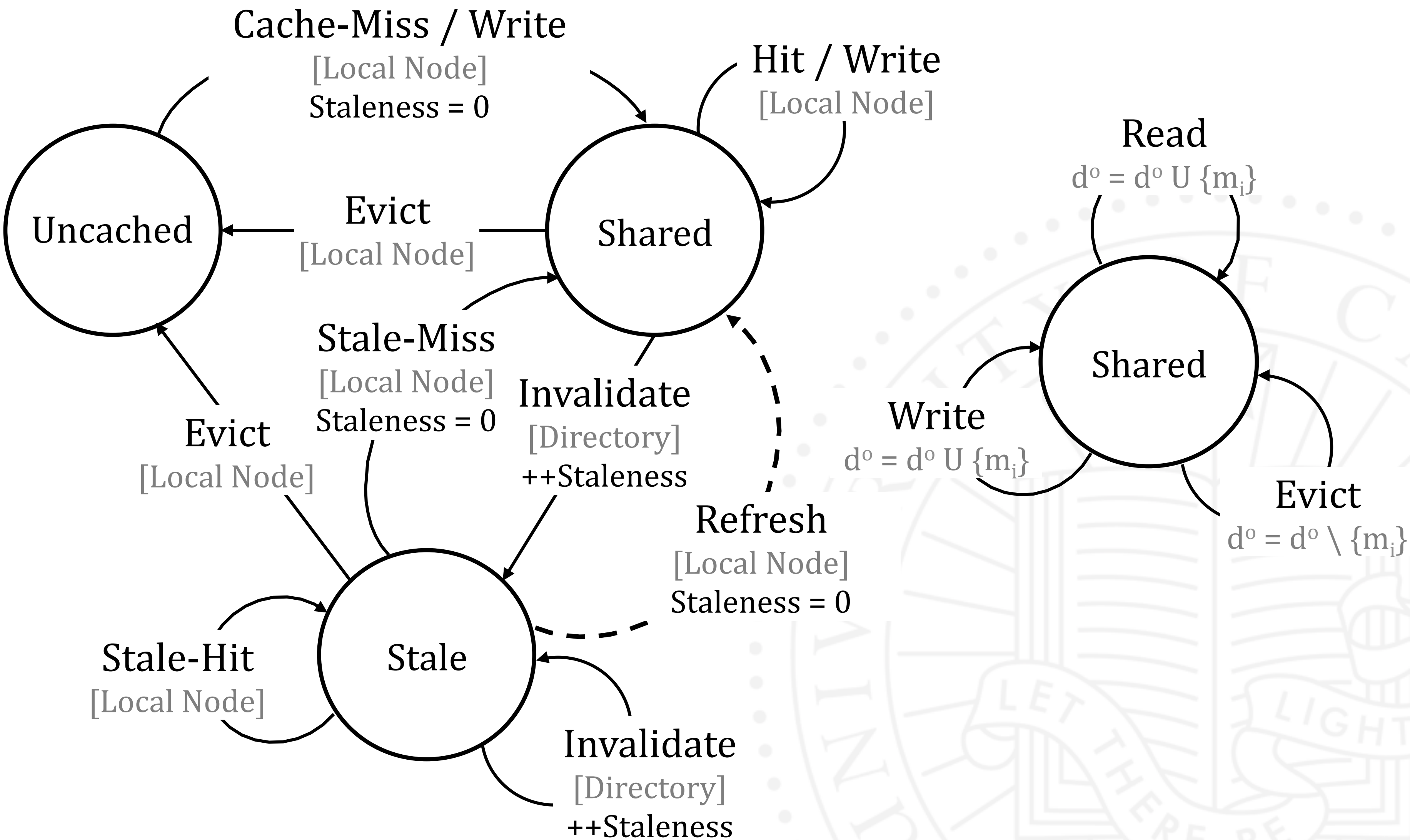
- Tracks staleness to exploit it
  - Cached objects have a staleness value
- Best efforts to minimize stale objects
  - Refresh cached objects based on access pattern

- Provides programming support
  - Local writes must be immediately visible
  - Once an object is read by a thread, no earlier writes to it can be read by the same thread

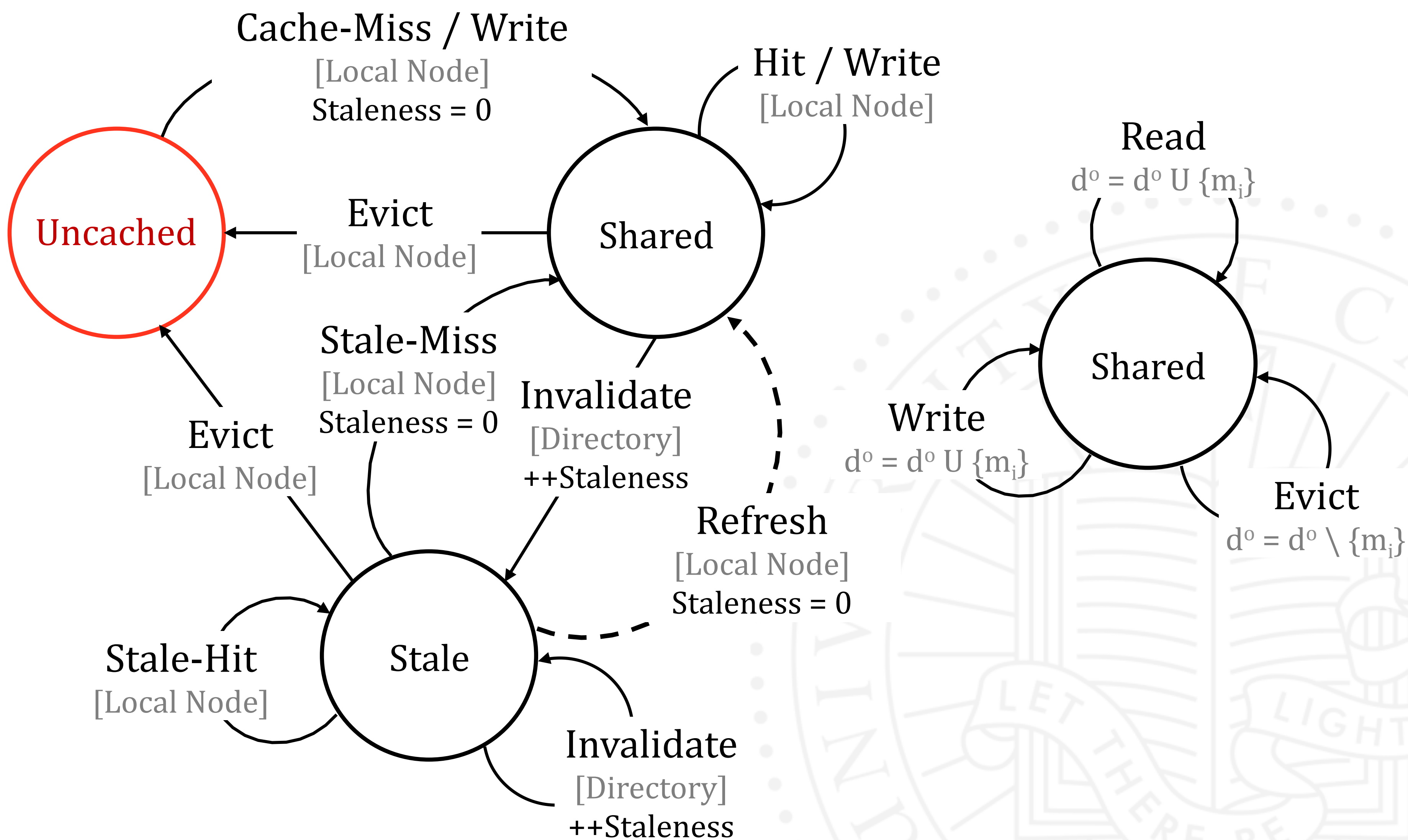
Relaxed Consistency Protocol

- Current-hit
  - object in cache; staleness = 0
- Stale-hit
  - object in cache; 0 < staleness <= t
- Stale-miss
  - object in cache; staleness > t
- Cache-miss
  - object not in cache

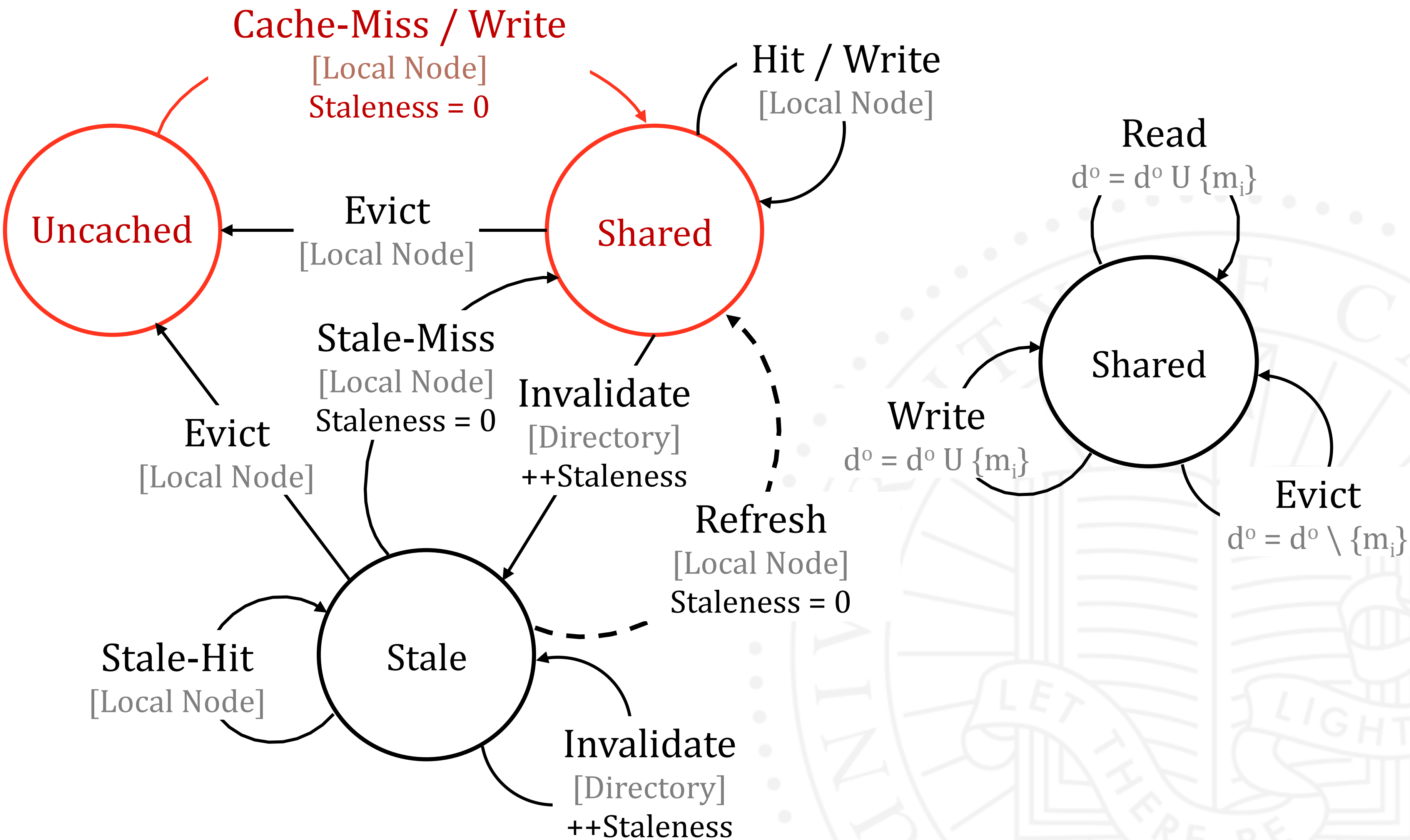
Relaxed Consistency Protocol



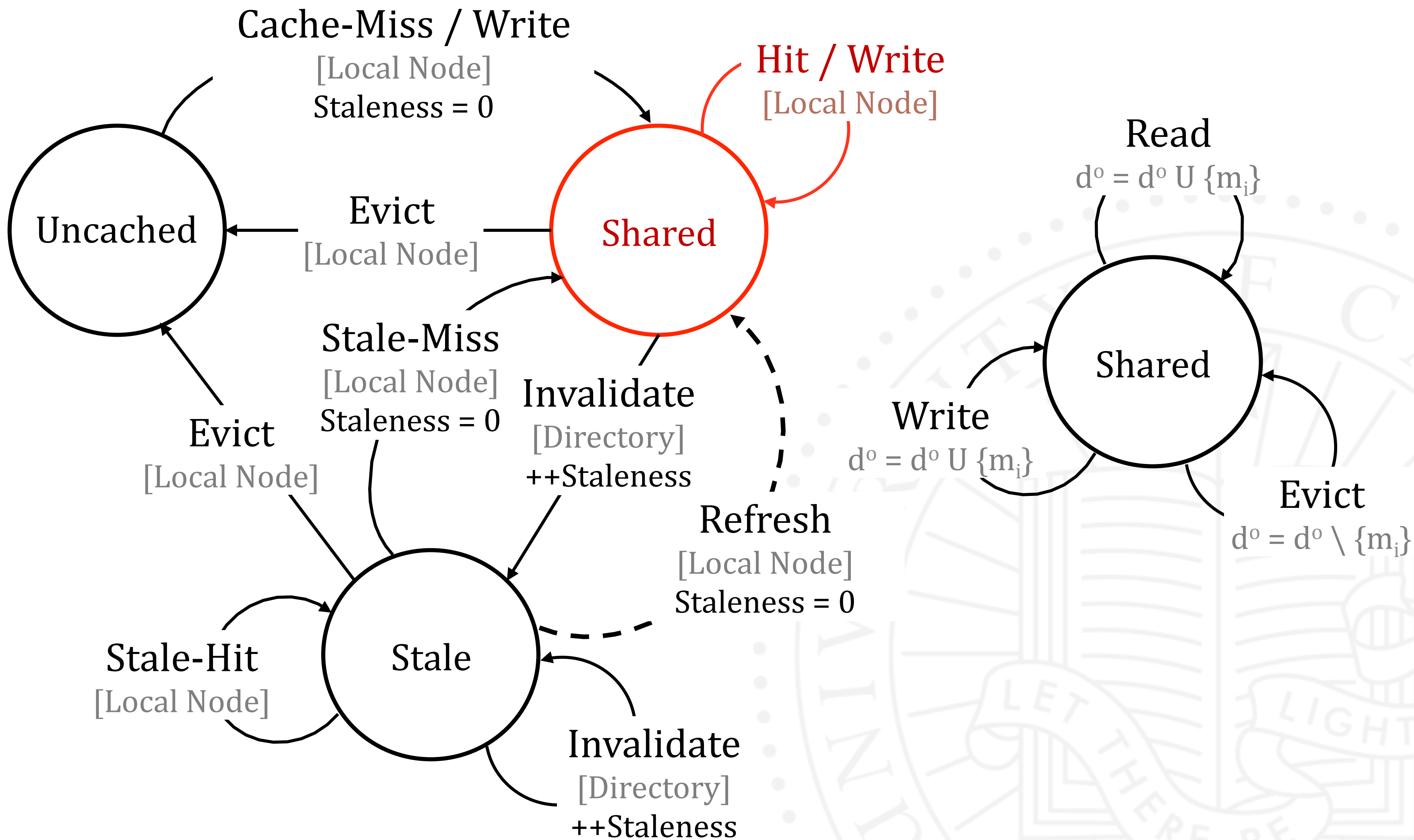
Relaxed Consistency Protocol



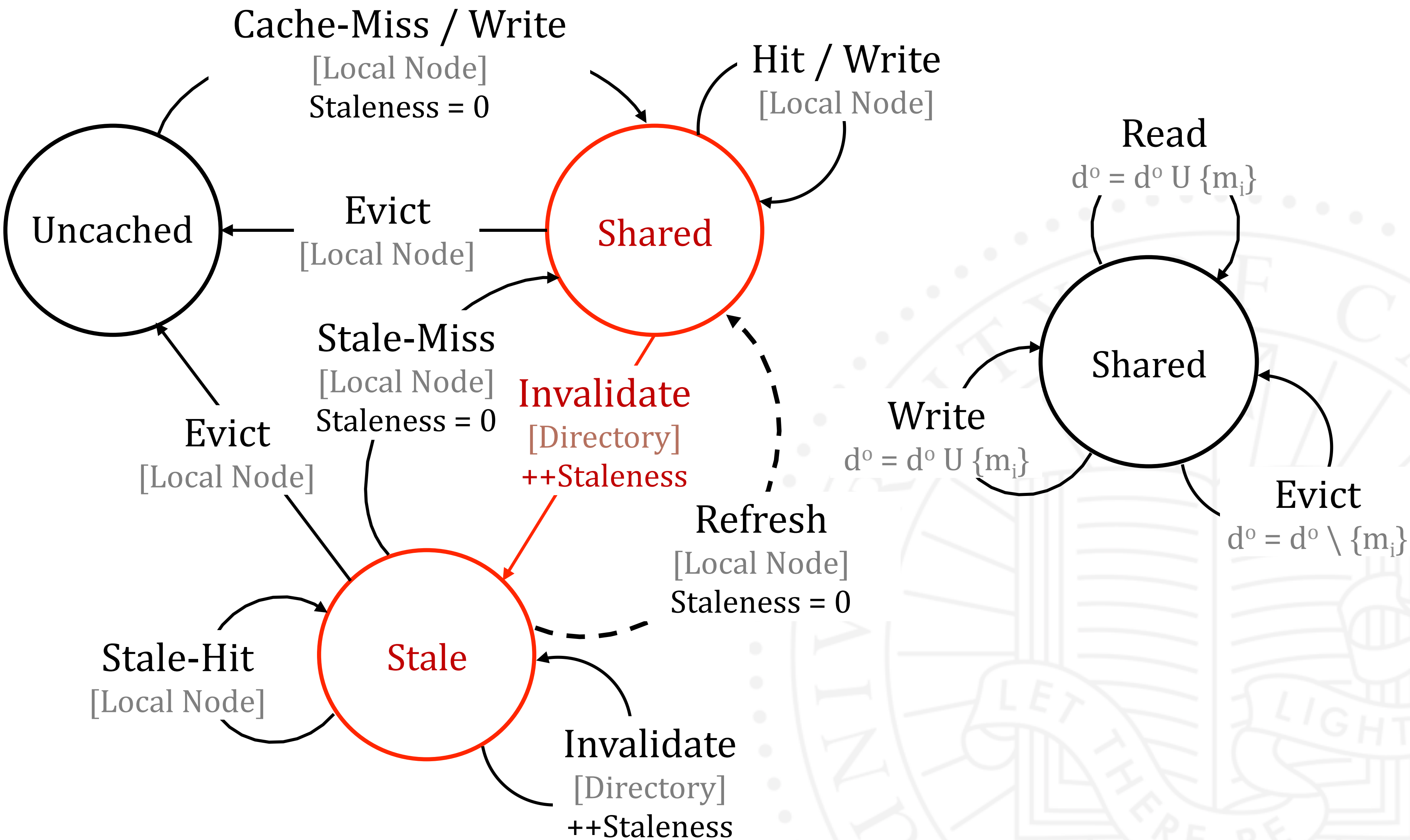
Relaxed Consistency Protocol



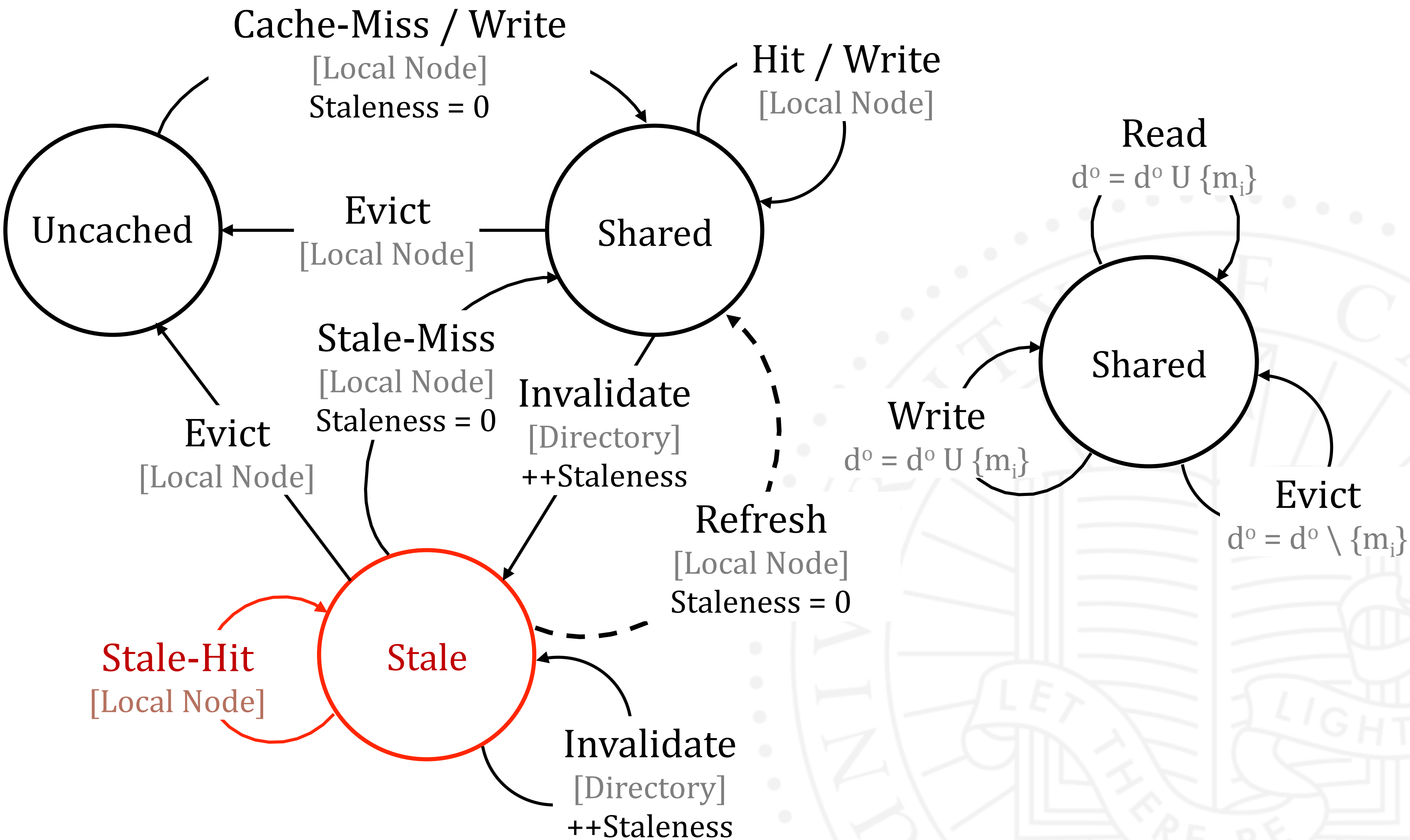
Relaxed Consistency Protocol



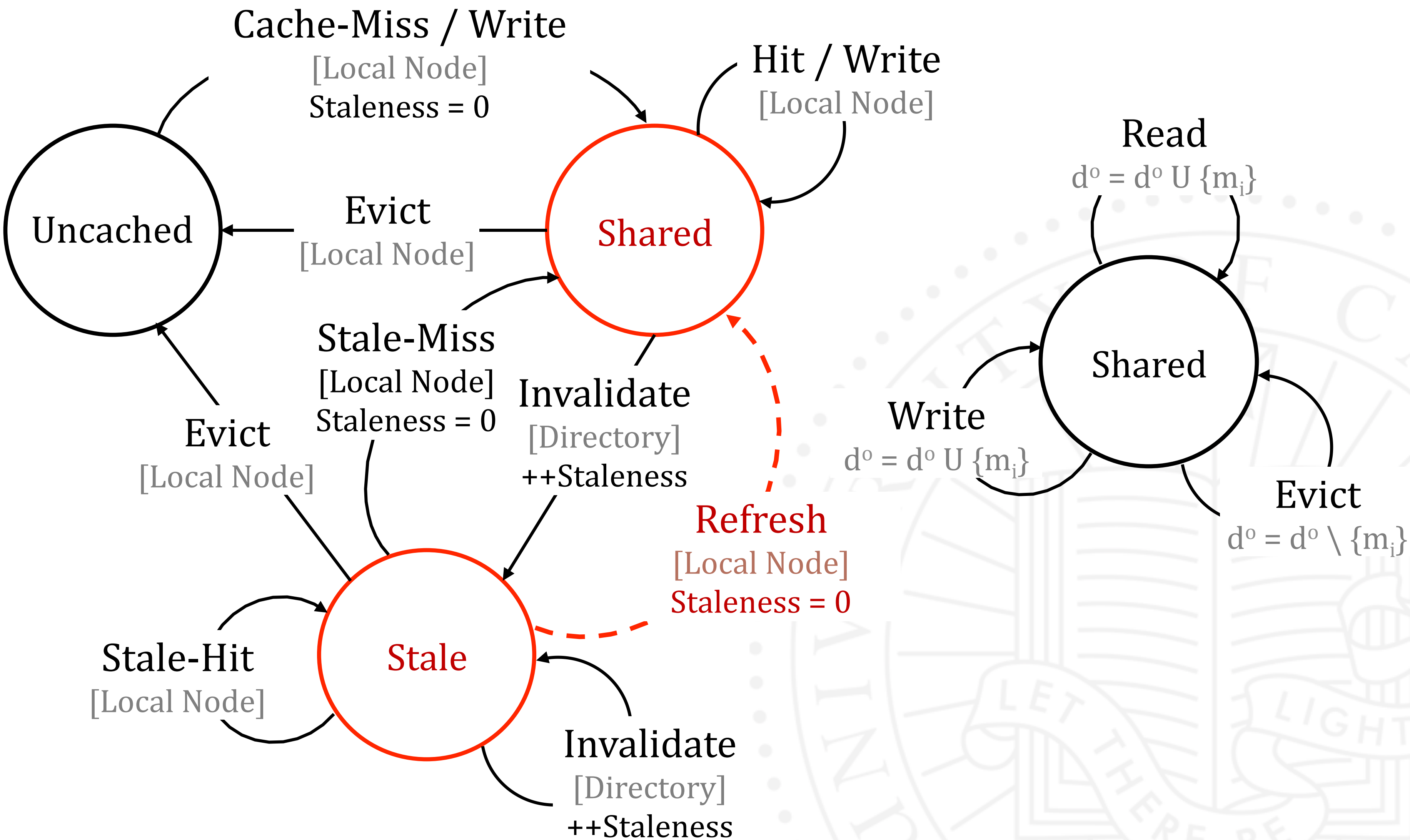
Relaxed Consistency Protocol



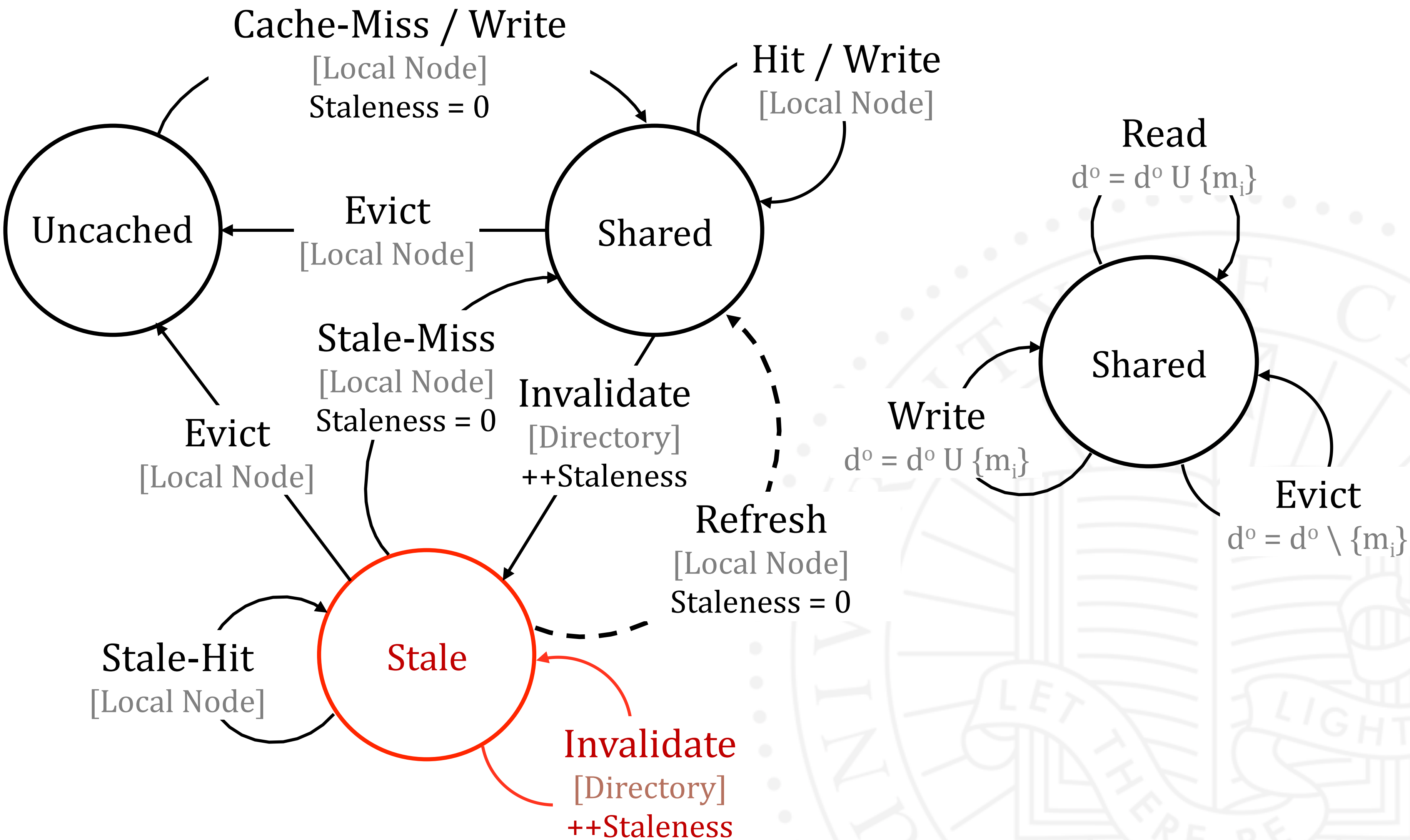
Relaxed Consistency Protocol



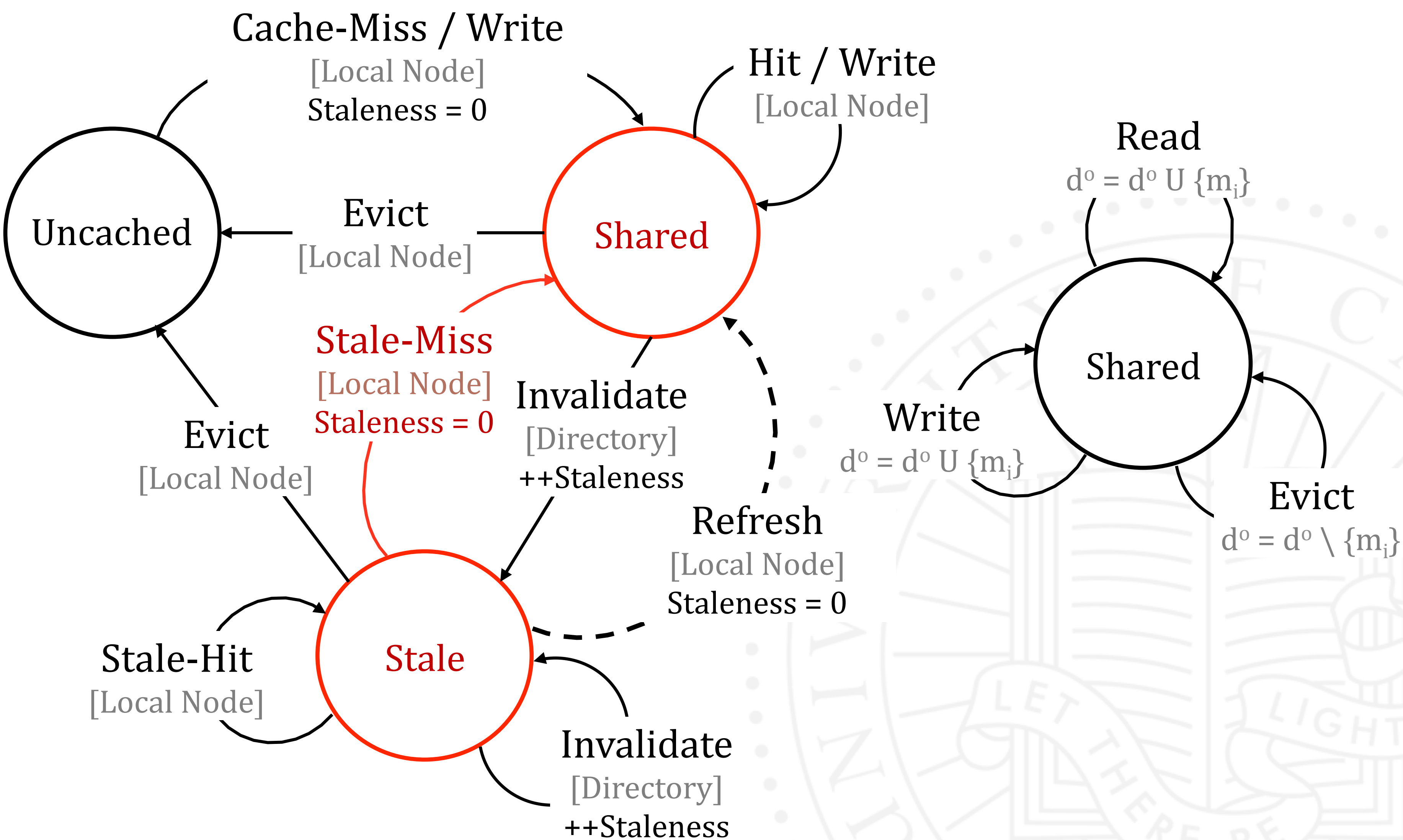
Relaxed Consistency Protocol



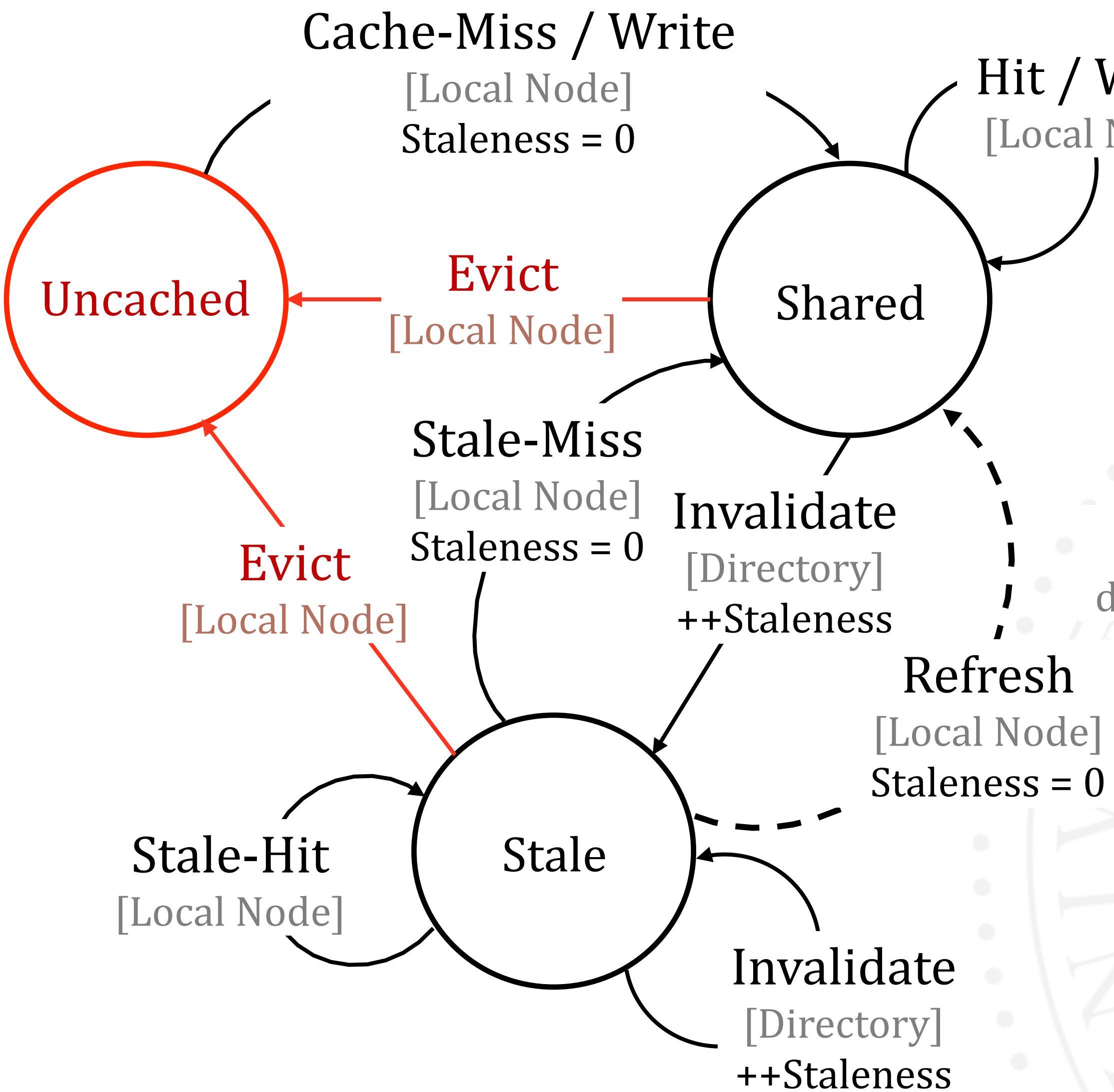
Relaxed Consistency Protocol

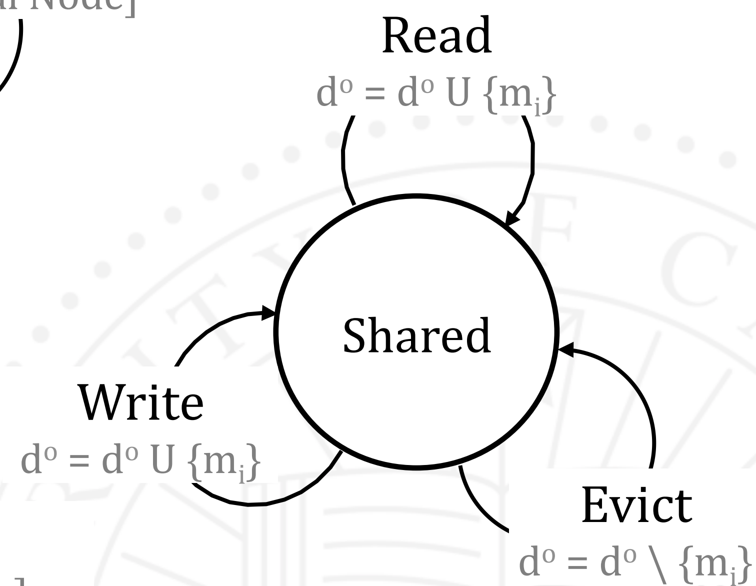


Relaxed Consistency Protocol

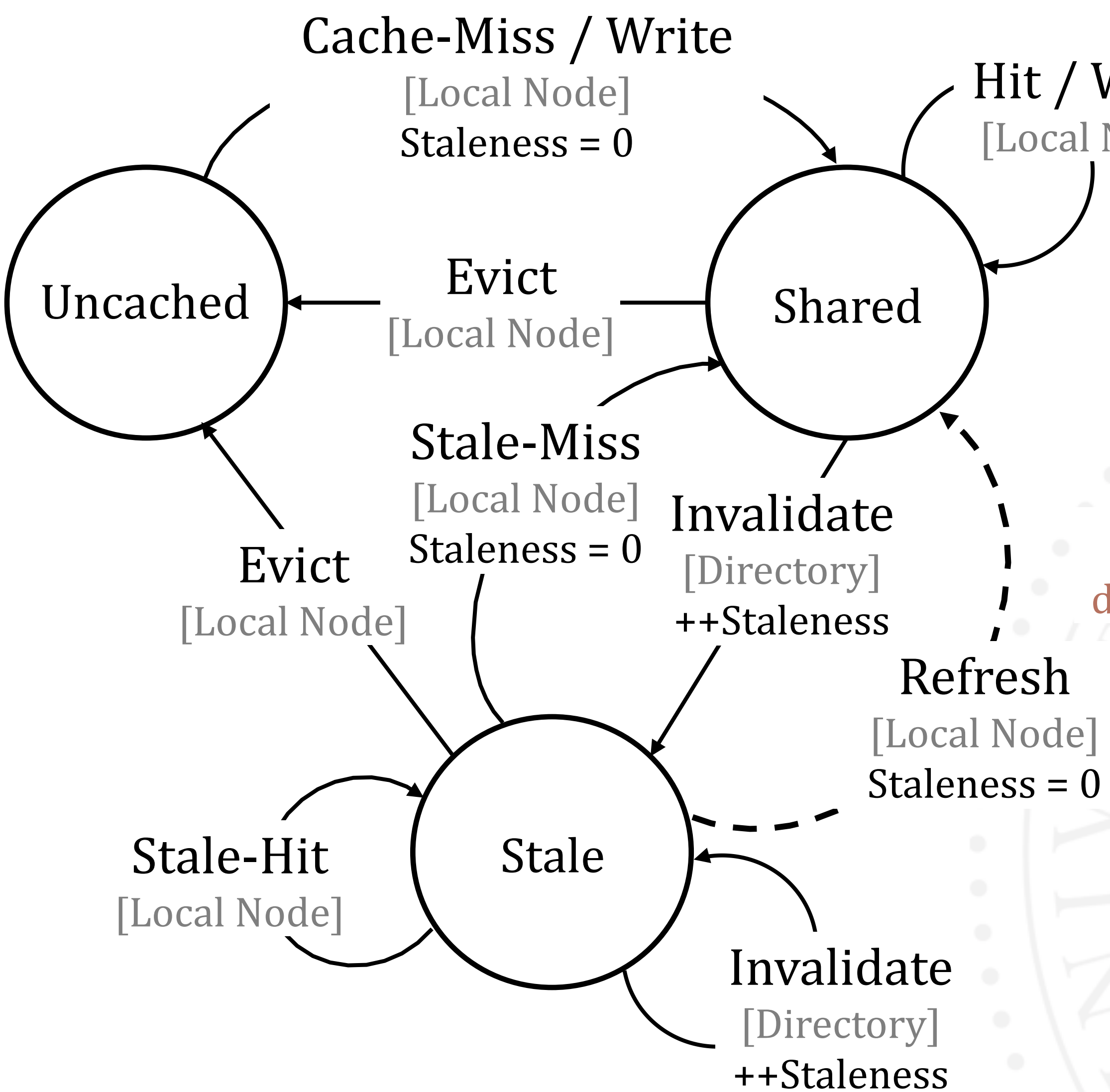


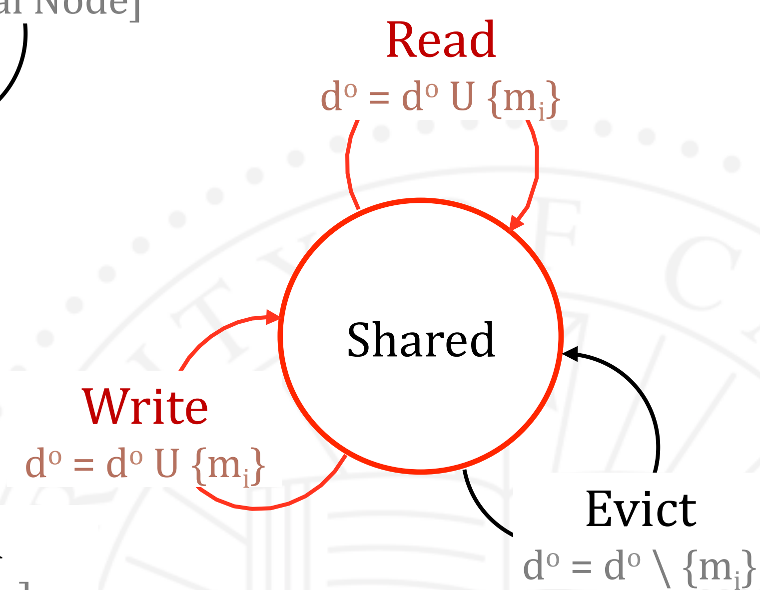
Relaxed Consistency Protocol



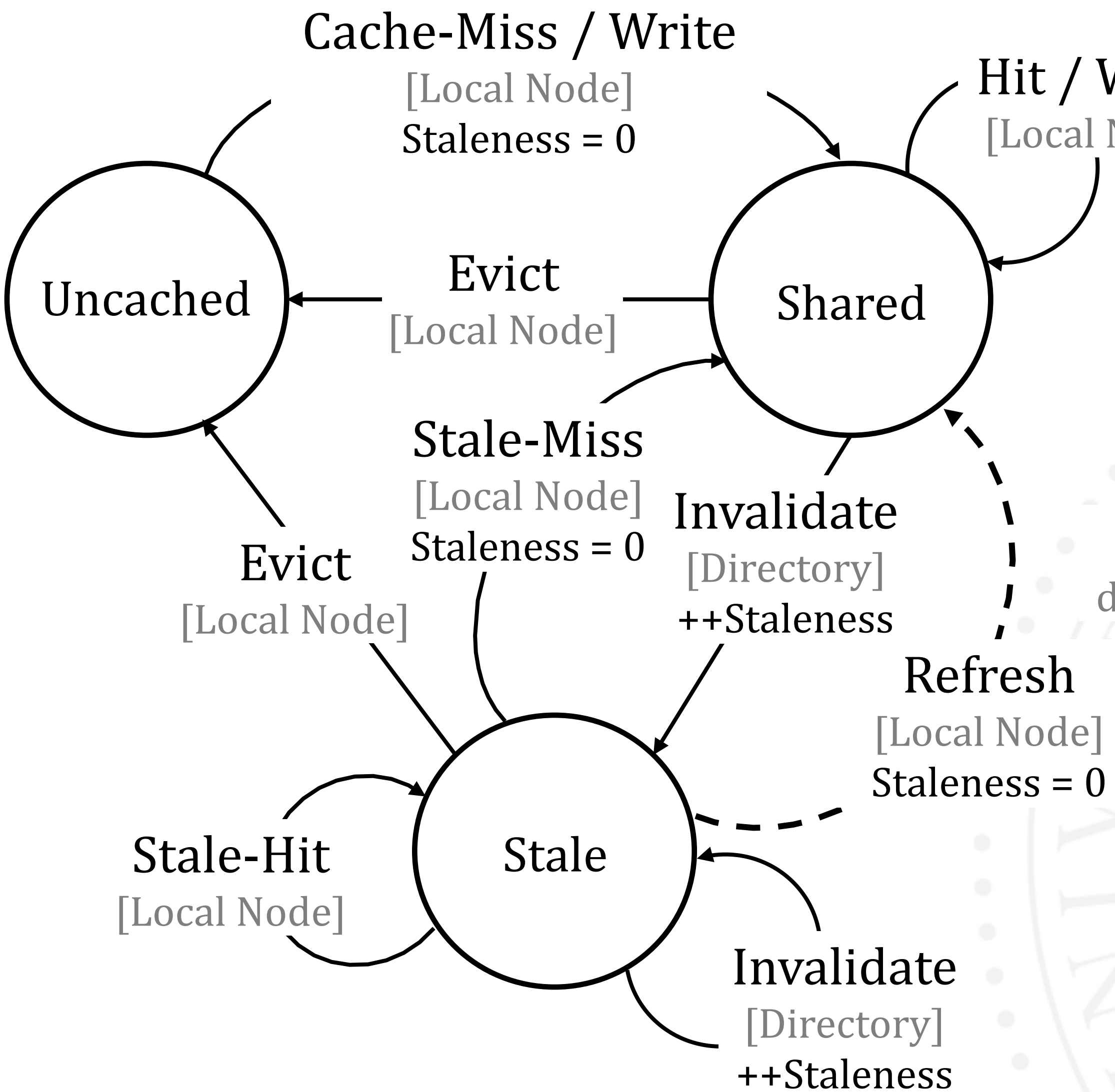


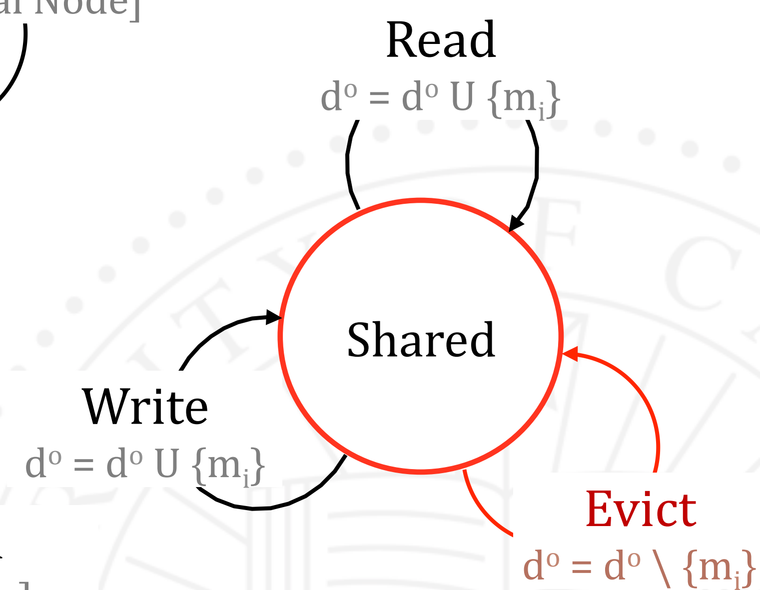
Relaxed Consistency Protocol





Relaxed Consistency Protocol





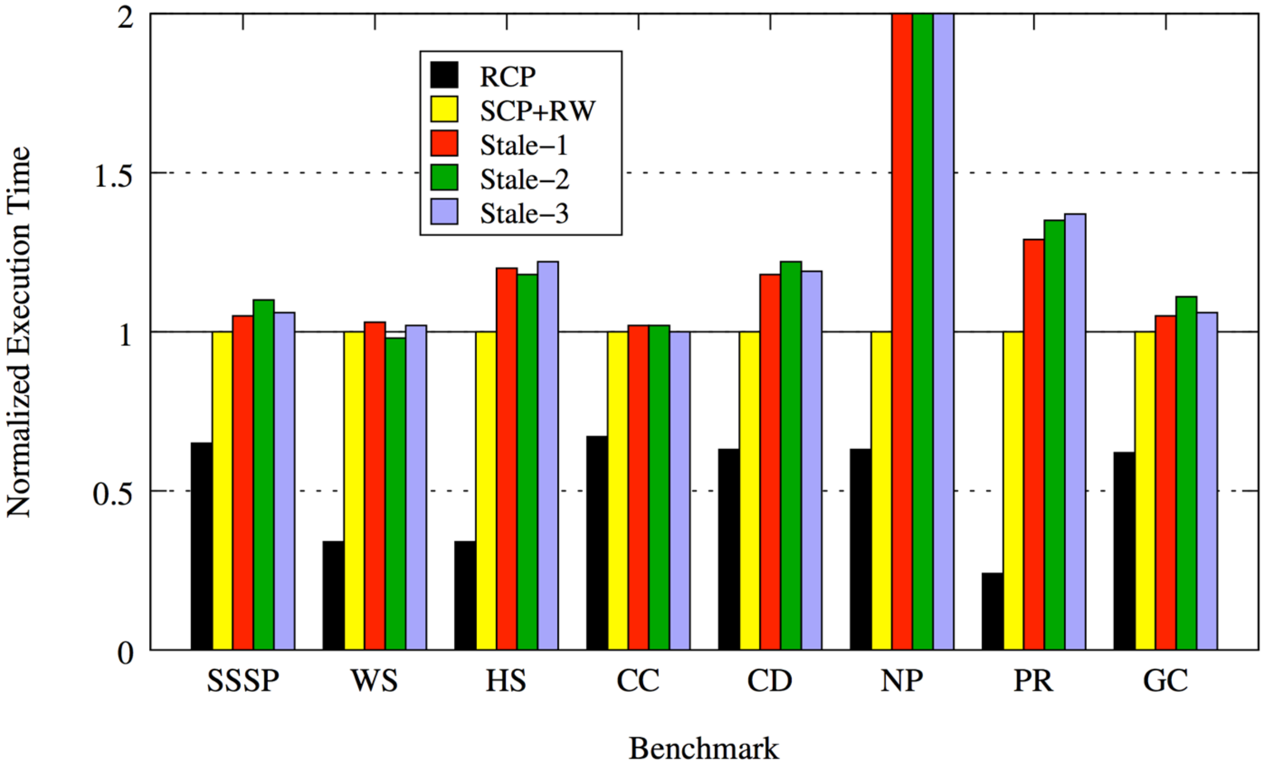
Implementation

- Similar to dyDSM [Koduru et al. 2013]
  - Object based
  - Protocol relaxes strict consistency
  - Graphs are distributed using METIS [SISC 99]
- Runtime
  - Single Writer Model
  - Refresh strategy using producer-consumer model
  - Termination semantics

Experimental Setup

- Tardis @ UCR
  - 16-node cluster running CentOS 6.3
- 8 benchmarks
  - WS, HS, PR, SSSP, CC, CD, NP, GC
- 10 real-world inputs
  - SNAP dataset collection
  - UFL sparse matrix collection

Execution Time

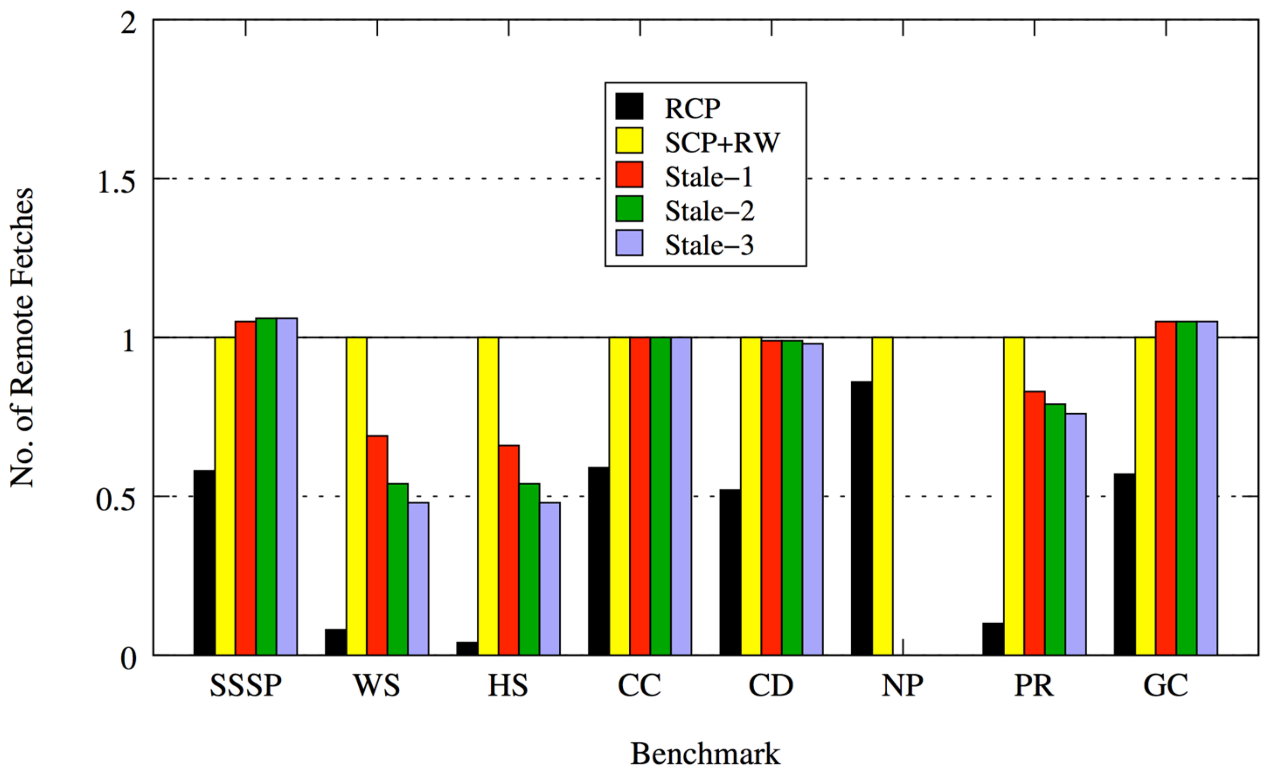


Pokec:
30M edges
1.6M vertices

AtmosModl:
10M edges
1.4M vertices

RCP 48.7% faster than SCP+RW and 56% faster than best Stale-n

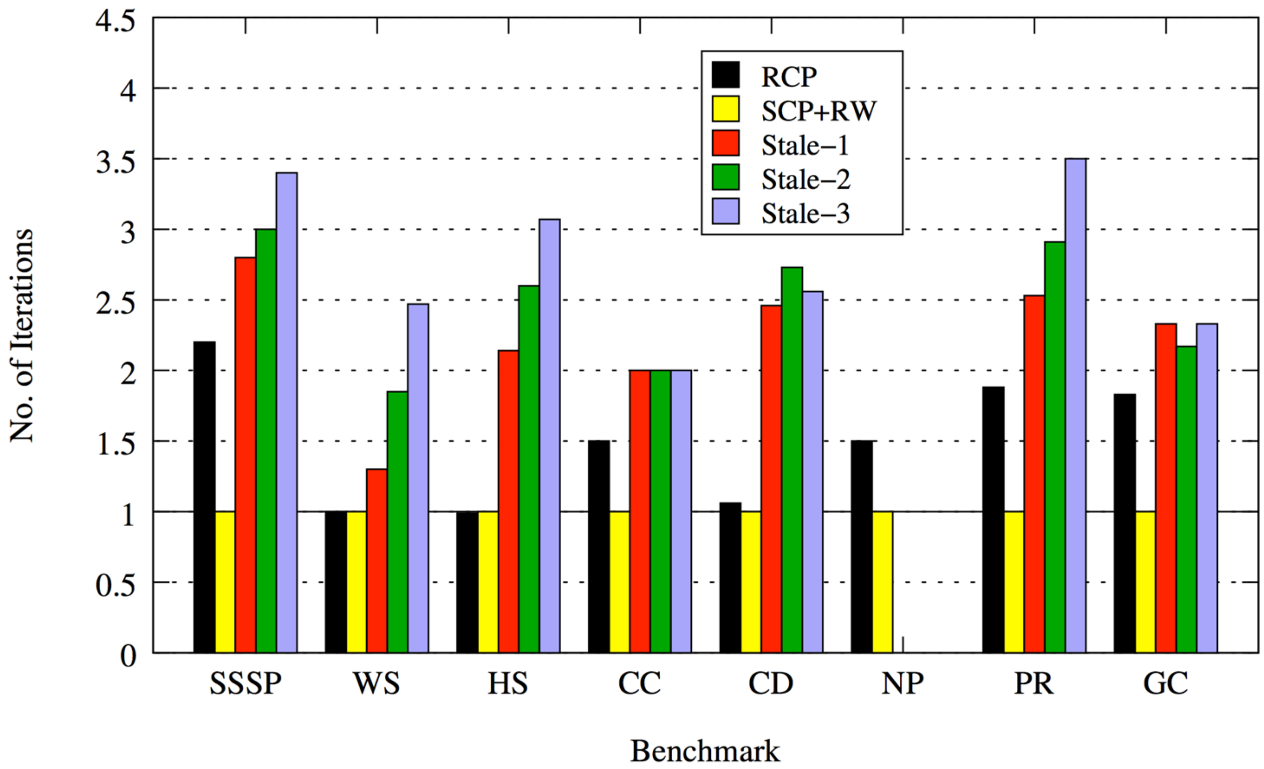
Remote Fetches



RCP blocks on 41.8% of remote fetches (7.5% for PR, WS & HS)

Best Stale-n blocks on 85.6% of remote fetches

Iterations

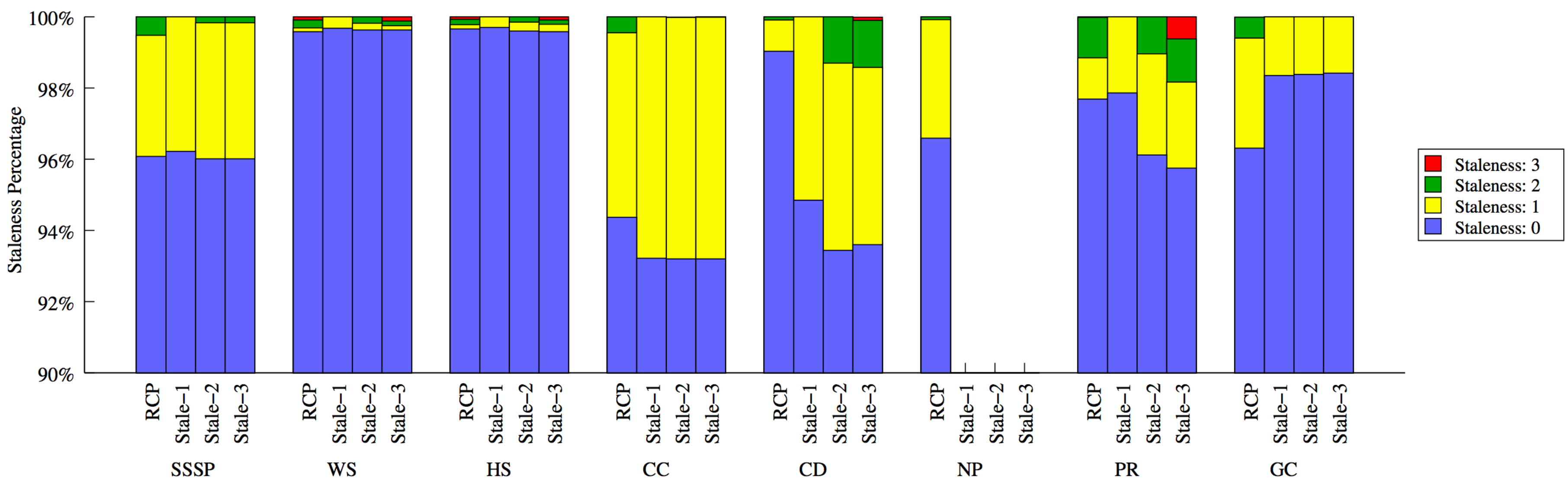


RCP requires 49.5% more iterations

Stale-2/Stale-3 require 146/176% more iterations

Staleness Percentage

97.4% of values have staleness 0; 2.2% of values have staleness 1



Graph Processing Frameworks

- Distributed memory
  - GraphLab [VLDB’12], Pregel [SIGMOD’10], PowerGraph [OSDI’12]
- Shared memory
  - Ligra [PPoPP’13], Grace [CIDR’13], Galois[PLDI’07]
- Out-of-core
  - GraphChi [OSDI’12], X-Stream [SOSP’13]

GraphLab

		SSSP	PR	GC	CC	NP
Orkut	**RCP**	161.88	822.95	92.79	90.31	2.35
	GraphLab	239.4	829.3	248.66	102.02	140.5
LiveJournal	**RCP**	21.73	343.96	17.44	22.09	133.43
	GraphLab	15.7	295.1	X	66.99	150.2
Pokec	**RCP**	9.47	169.47	8.81	7.1	1.74
	GraphLab	8.7	159.9	173.47	40.52	76.4
HiggsTwitter	**RCP**	2.5	15.64	3.59	4.1	0.48
	GraphLab	5.5	X	263.45	16.21	32.5
RoadNetCA	**RCP**	49.47	7.70	0.93	56.96	16.51
	GraphLab	60.3	88.4	50.22	220.9	37.4
RoadNetTX	**RCP**	44.21	5.05	0.53	50.94	15.83
	GraphLab	18.3	78	60.76	115.6	X

- RCP compares favorably with GraphLab
- RCP is orthogonal to these frameworks
  - Can be used in their asynchronous engines

Other Experiments

- Different inputs
- Overhead study
- Design choices
    - Invalidates v/s updates
    - Sensitivity to object size
    - Sensitivity to writes
    - Sensitivity to communication delay
- Different cluster sizes

Conclusion

- Relaxing consistency is useful
  - With controlled use of staleness
  - Using refresh strategy
- Efficient than prior DSMs based on
  - Strict consistency
  - Delta consistency
- Processing Frameworks
  - Easier to code (Pregel)
  - Compares favorably (GraphLab)

Thanks

- GRASP
  - http://grasp.cs.ucr.edu
- Acknowledgements