A Relaxed Consistency based DSM for Asynchronous Parallelism

Keval Vora, Sai Charan Koduru, Rajiv Gupta

UNIVERSITY OF CALIFORNIA, RIVERSIDE

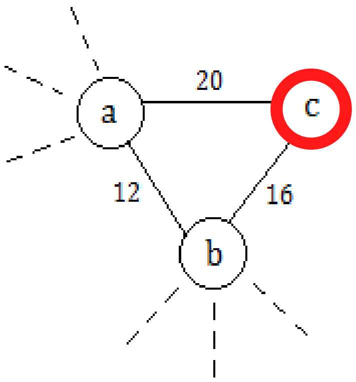
SoCal PLS - Spring 2014

Motivation

- Graphs are popular
  - Graph Mining: Community Detection, Coloring
  - Graph Analytics: PageRank, Shortest Paths
- Real-world graphs are large
  - Orkut: 234M edges, 3M vertices
  - LiveJournal: 68M edges, 4.8M vertices
- Processing on distributed memory machines
  - Performance
  - Programmability

Graph Algorithms

- Vertex Centric
  - Computation written for a single vertex
  - Highly parallel execution
- Iterative
  - Terminate when values converge
- Network Bound
  - Computation is simple



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

Our Work

- Improve asynchronous execution
  - Make them faster
- Relax consistency to tolerate latencies
  - Tardis: remote fetch is ~2.3 times of local fetch
  - Allow use of stale values

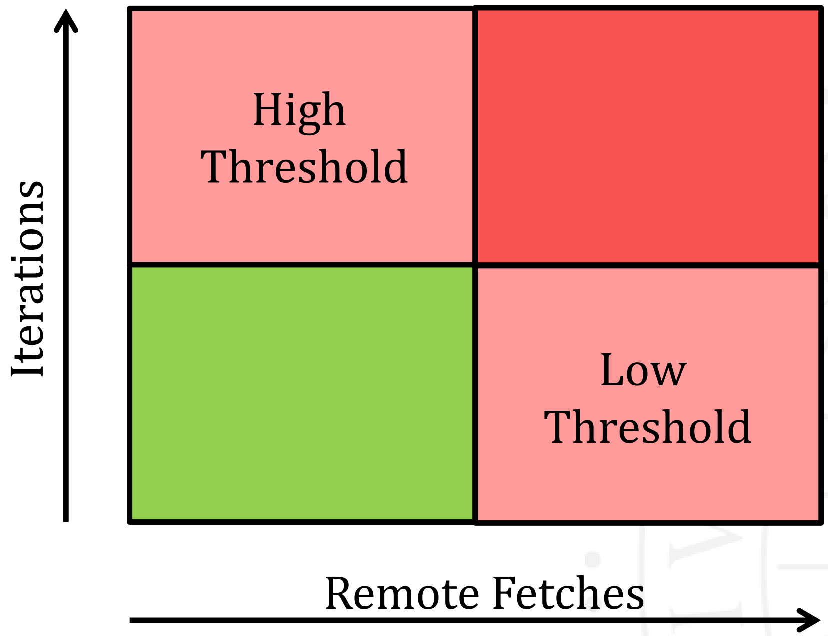
Our Work

- Improve asynchronous execution
  - Make them faster
- Relax consistency to tolerate latencies
  - Tardis: remote fetch is ~2.3 times of local fetch
  - Allow use of stale values

Challenge: Tolerate latencies without delaying convergence

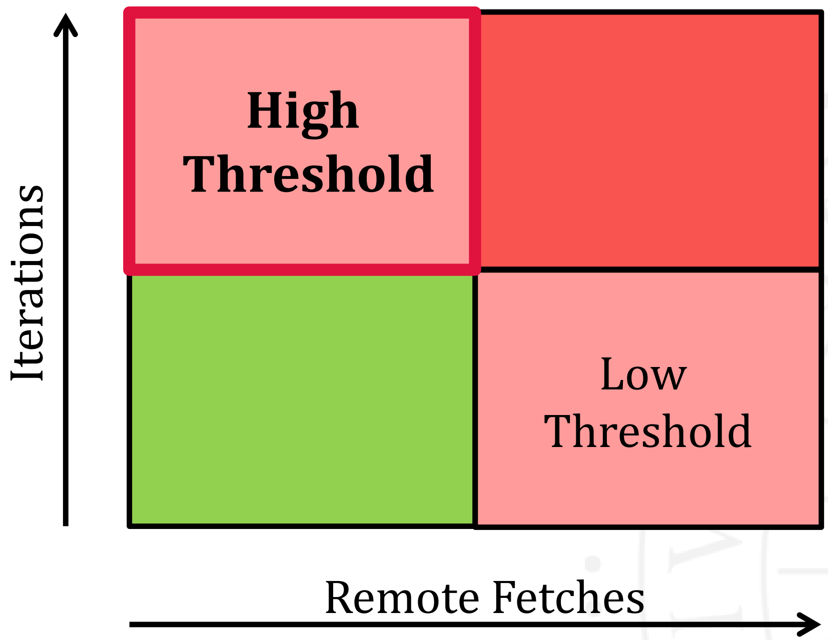
Weak Memory Models

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



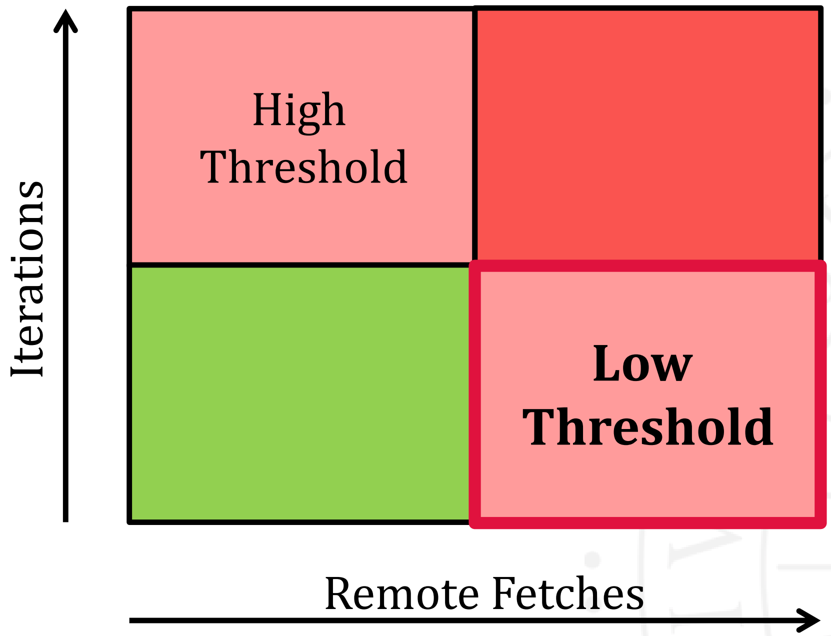
Weak Memory Models

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



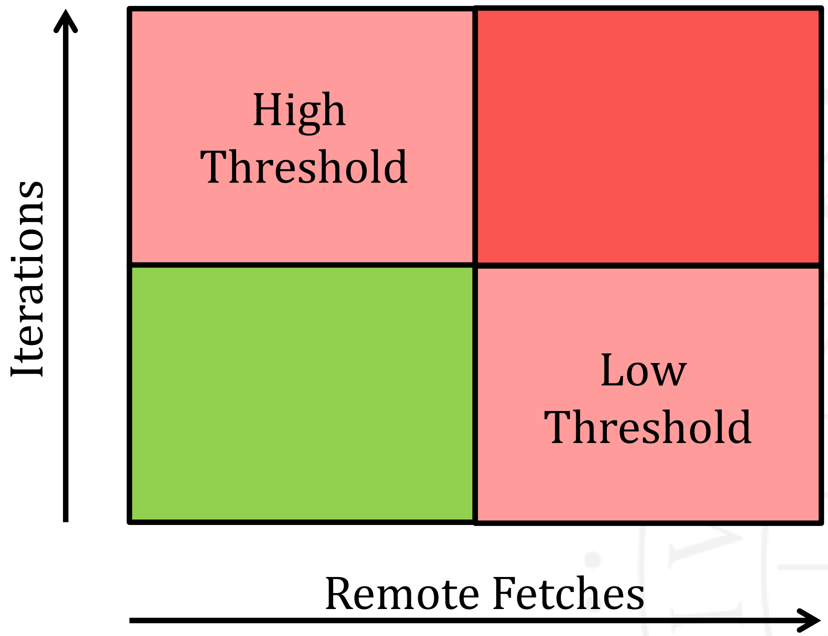
Weak Memory Models

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



Weak Memory Models

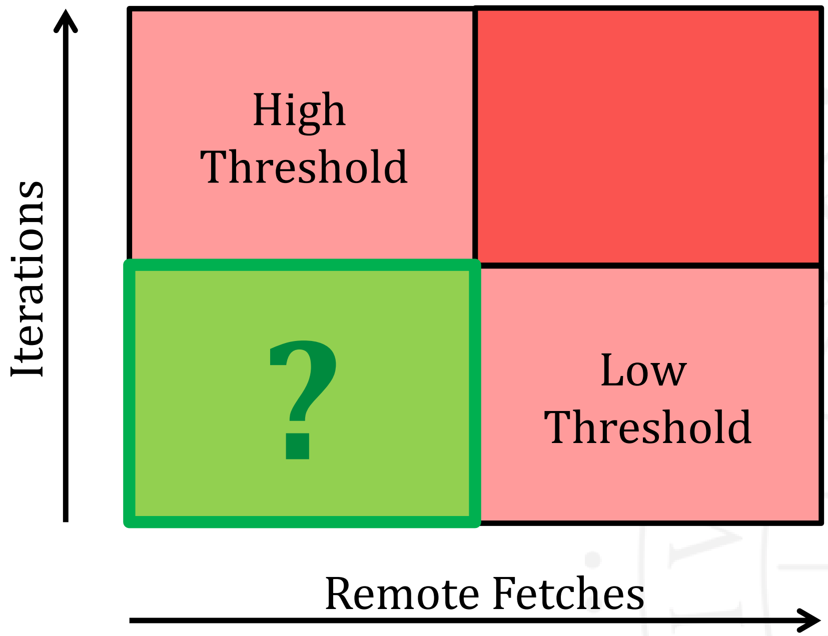
- Delta Consistency [SPAA'97] [PPoPP'03]
  - Controls staleness using static threshold



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

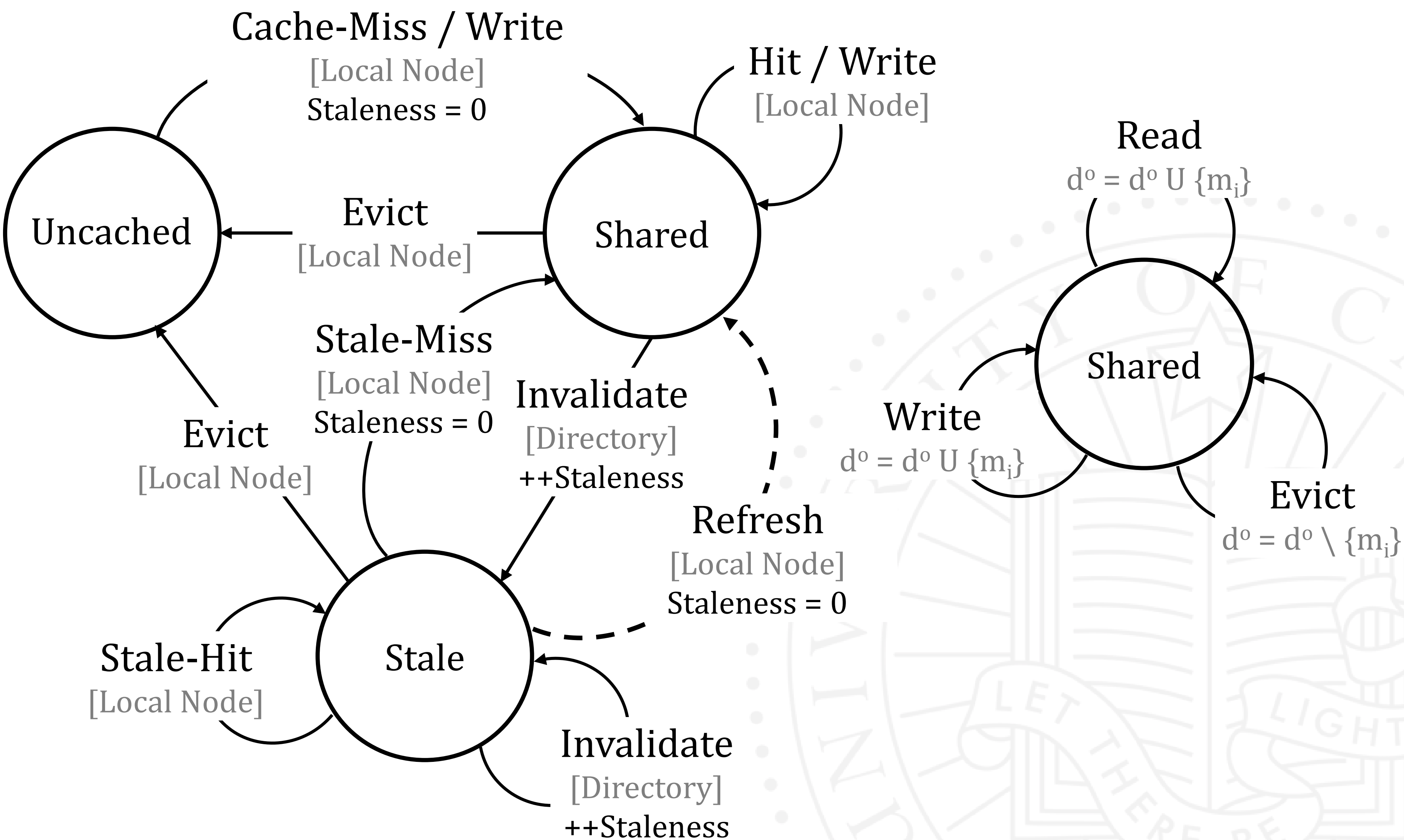
- Tracks staleness to exploit it
  - Cached objects have a staleness value
- Best efforts to minimize stale objects
  - Refresh cached objects based on access pattern

- 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

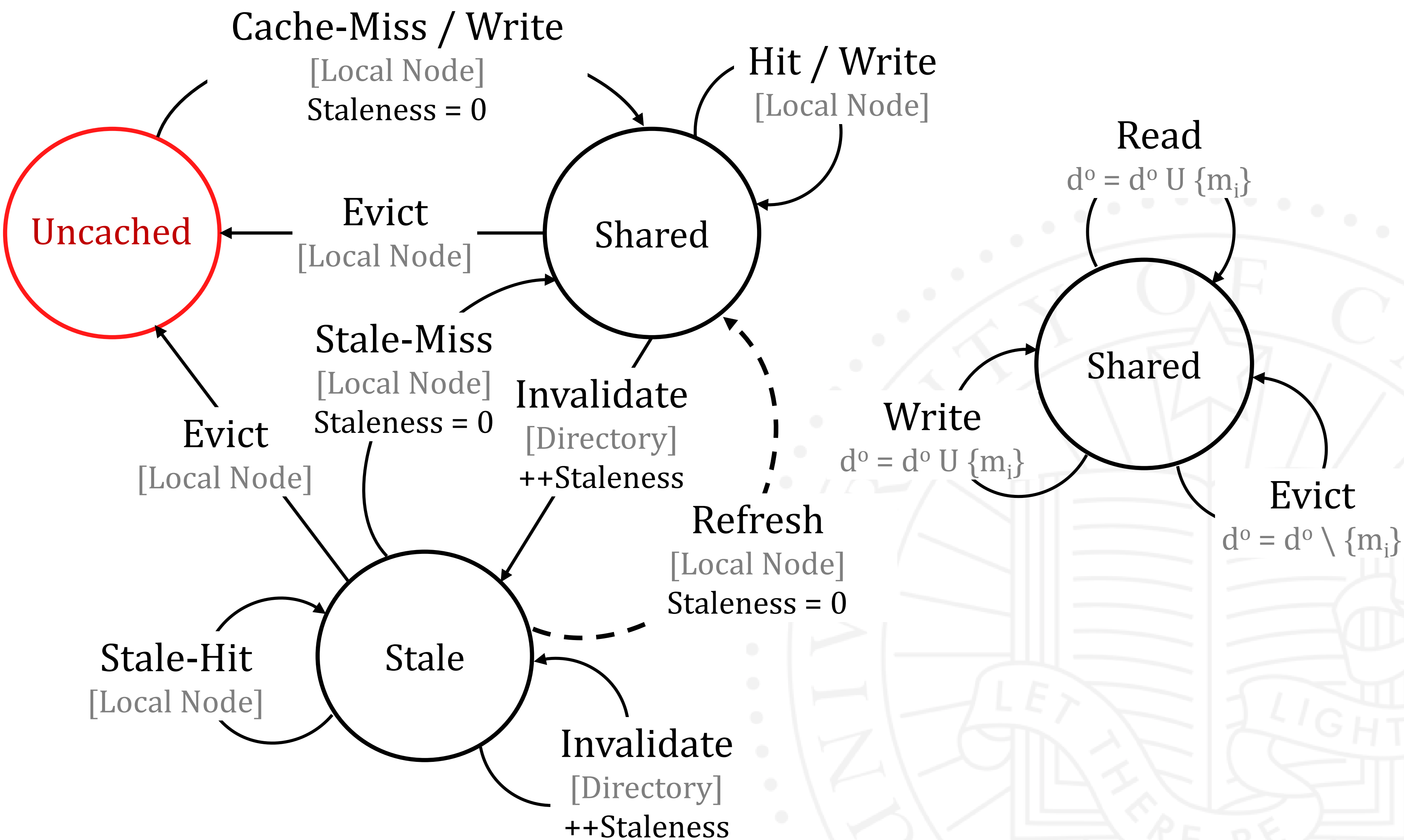
Relaxed Consistency Protocol

- 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

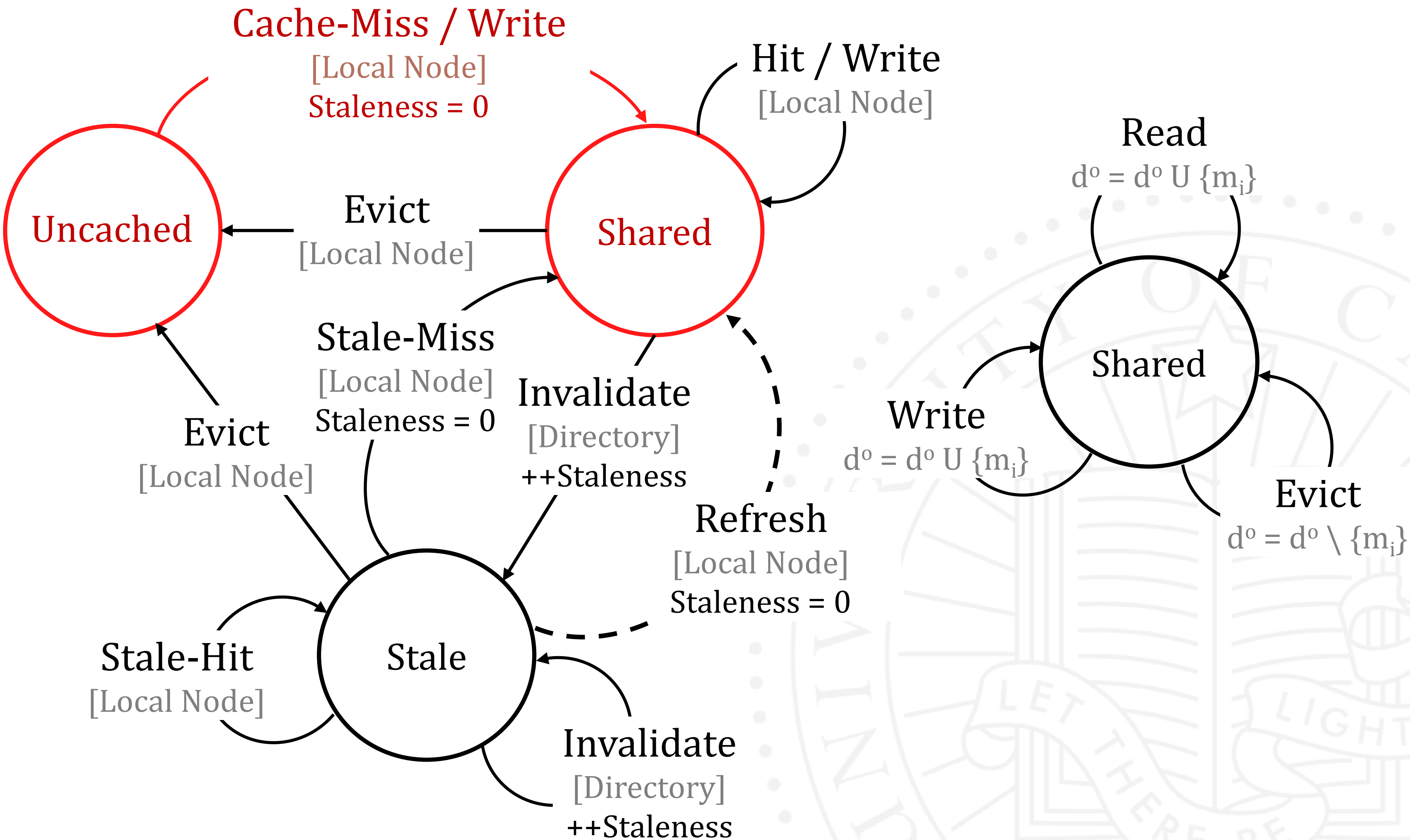
Relaxed Consistency Protocol



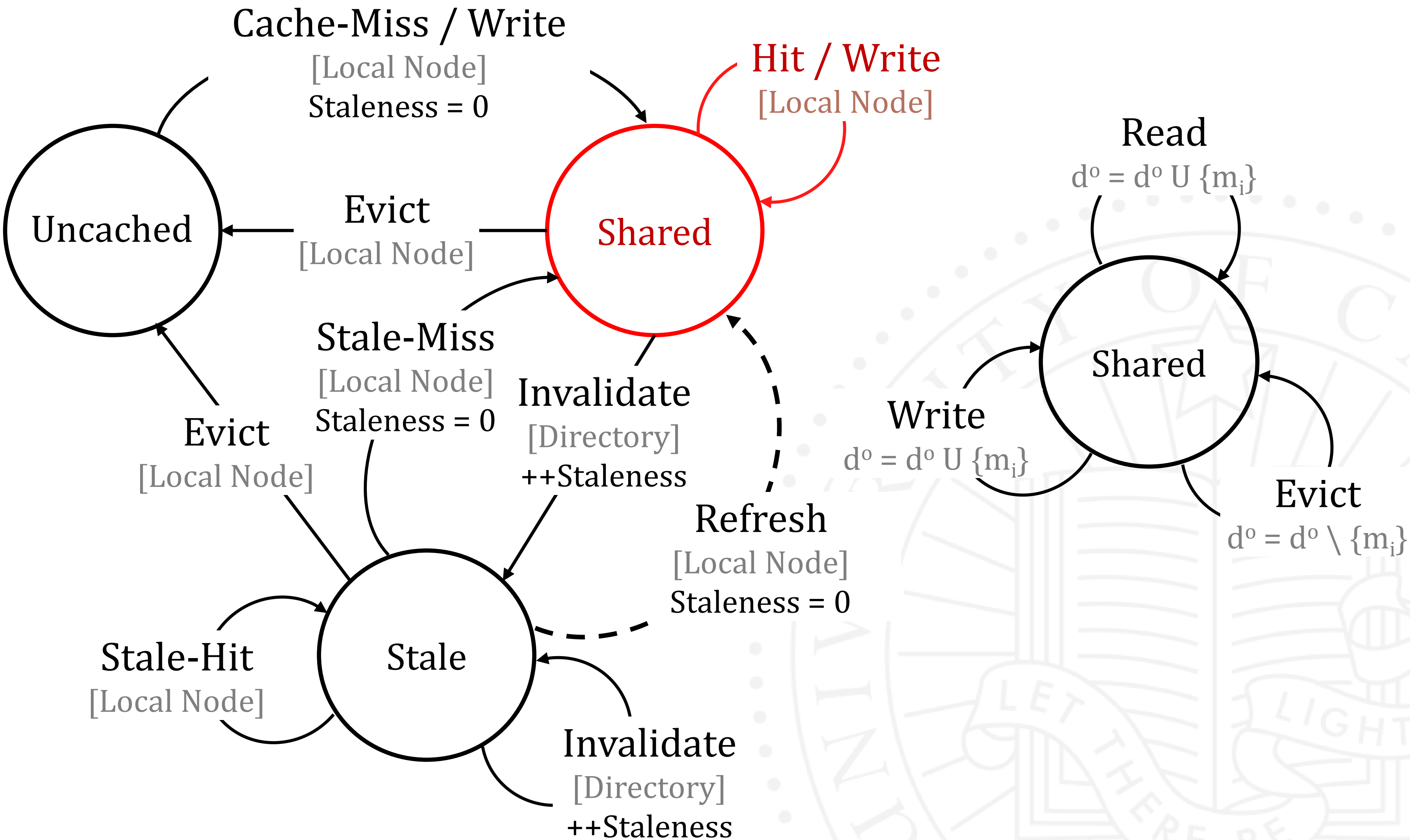
Relaxed Consistency Protocol



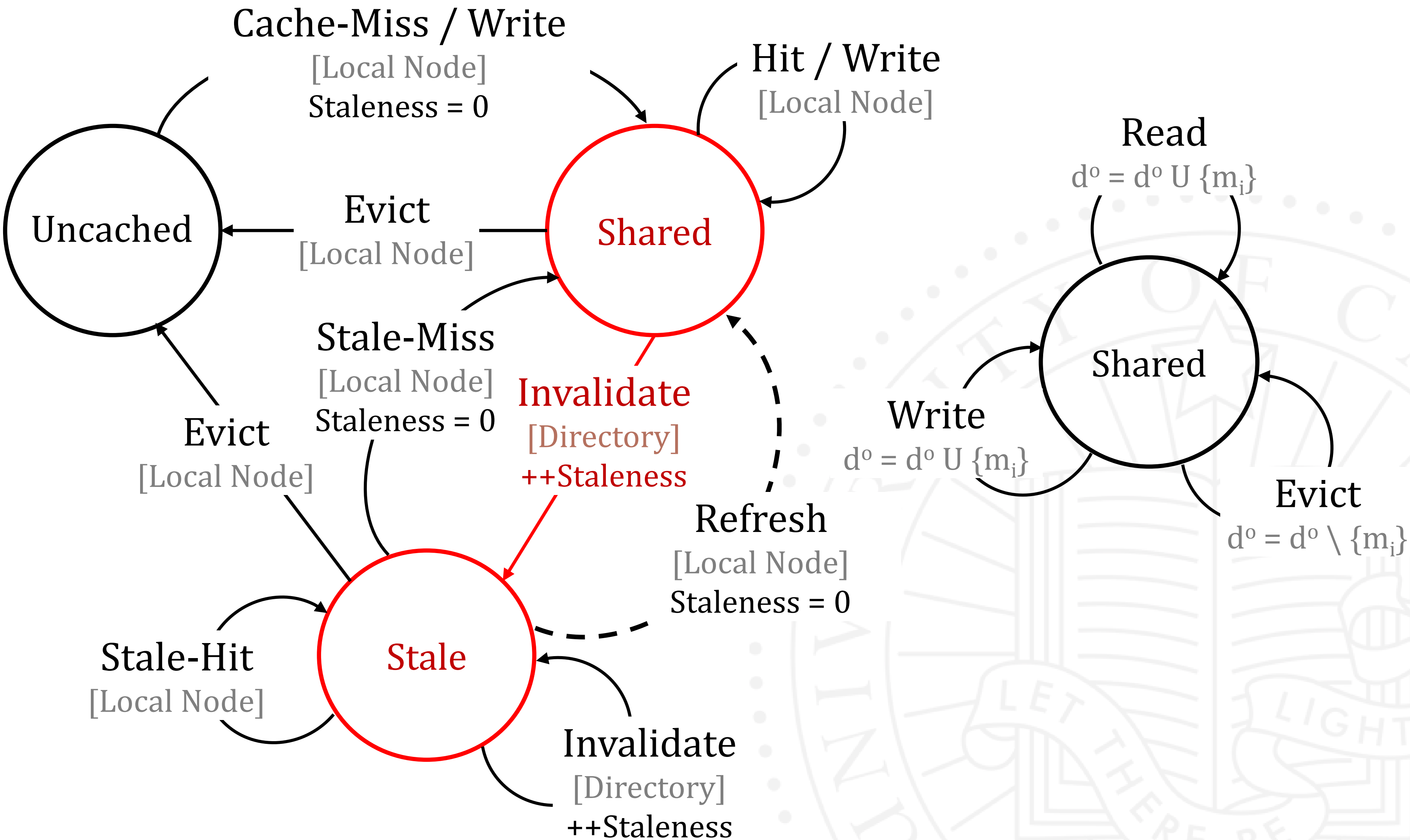
Relaxed Consistency Protocol



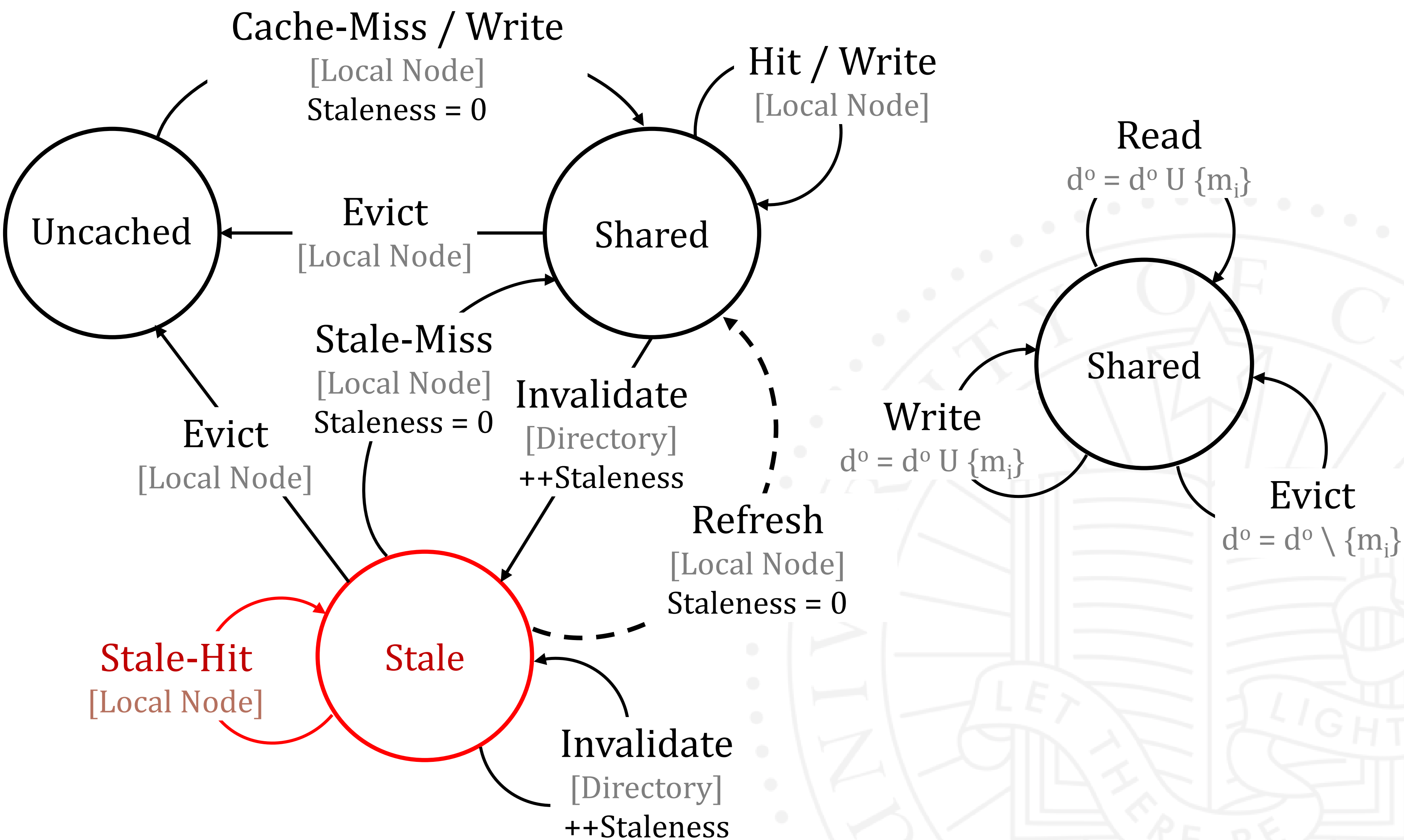
Relaxed Consistency Protocol



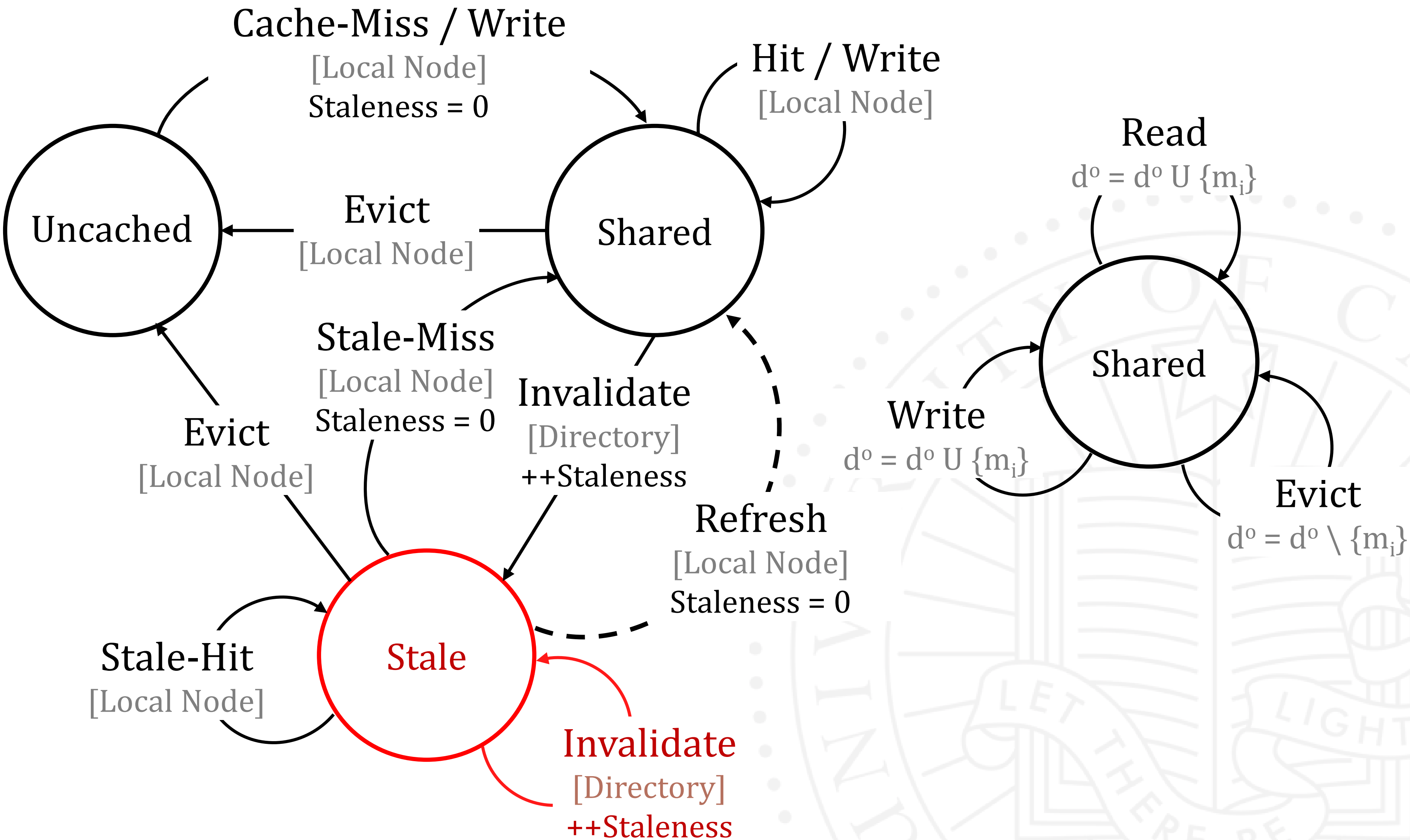
Relaxed Consistency Protocol



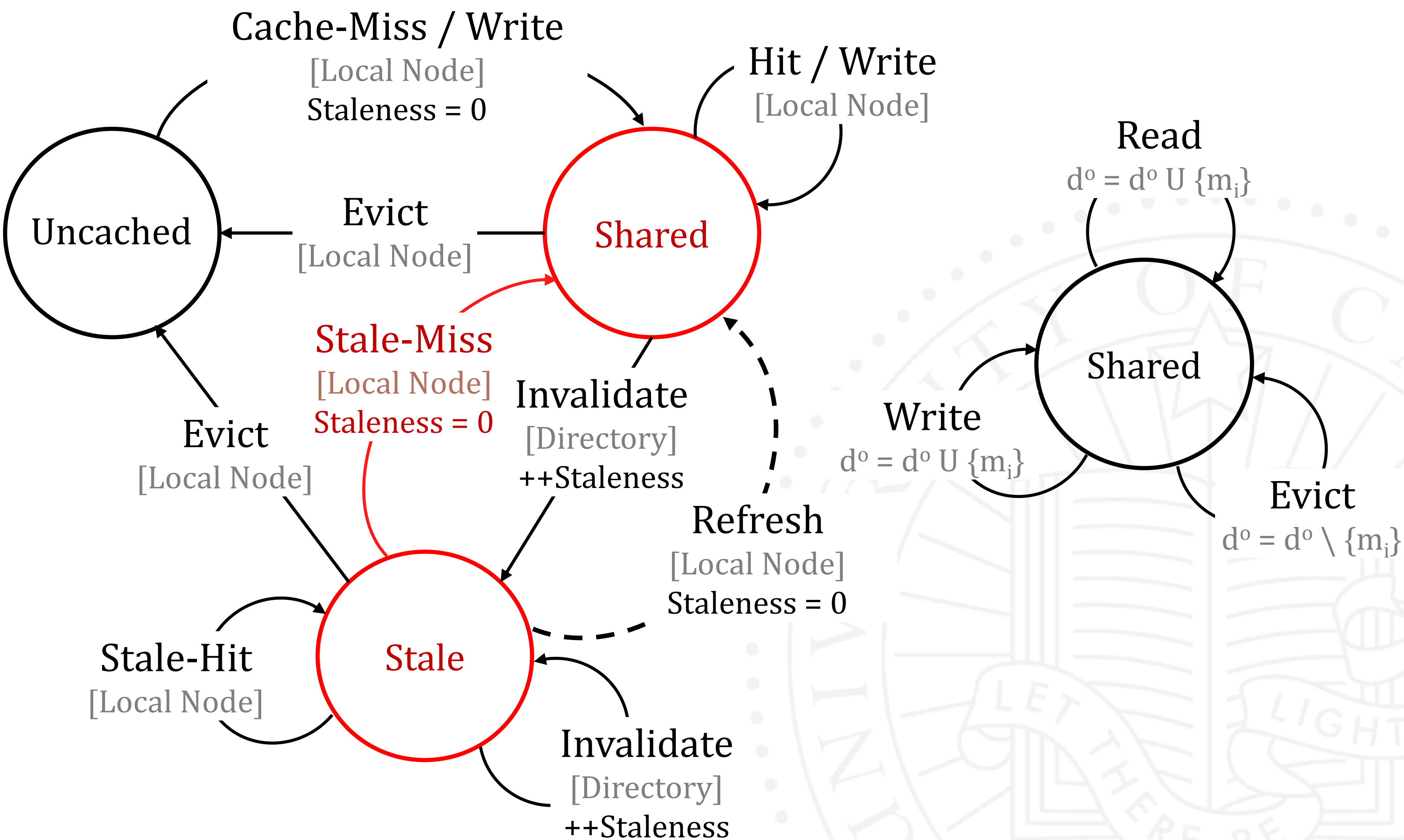
Relaxed Consistency Protocol



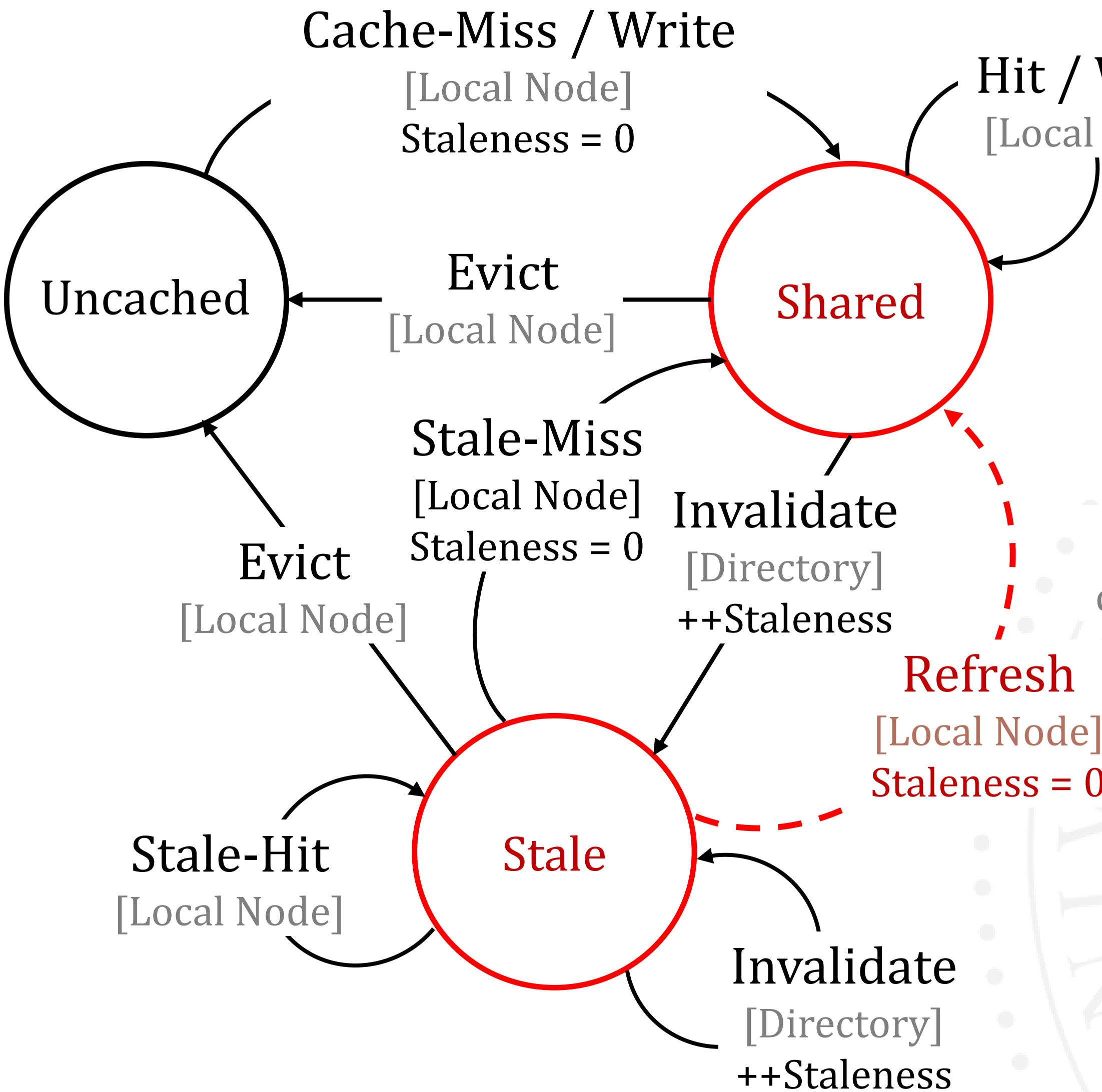
Relaxed Consistency Protocol

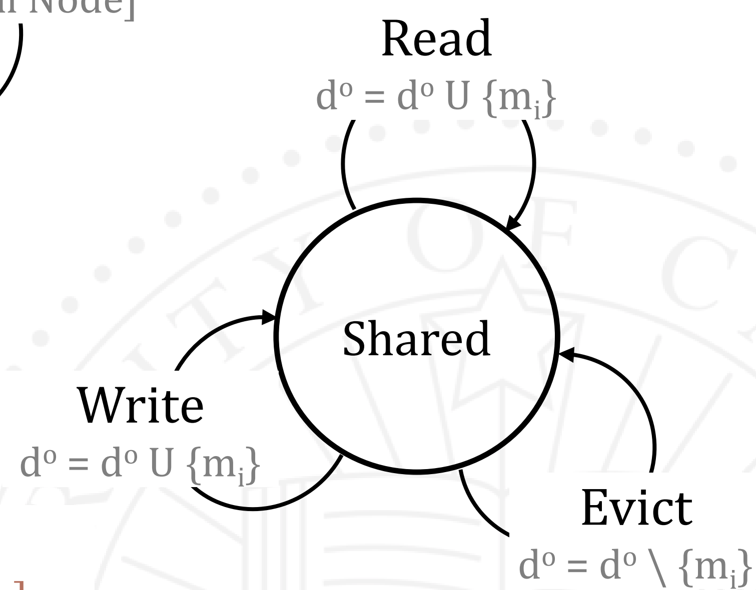


Relaxed Consistency Protocol

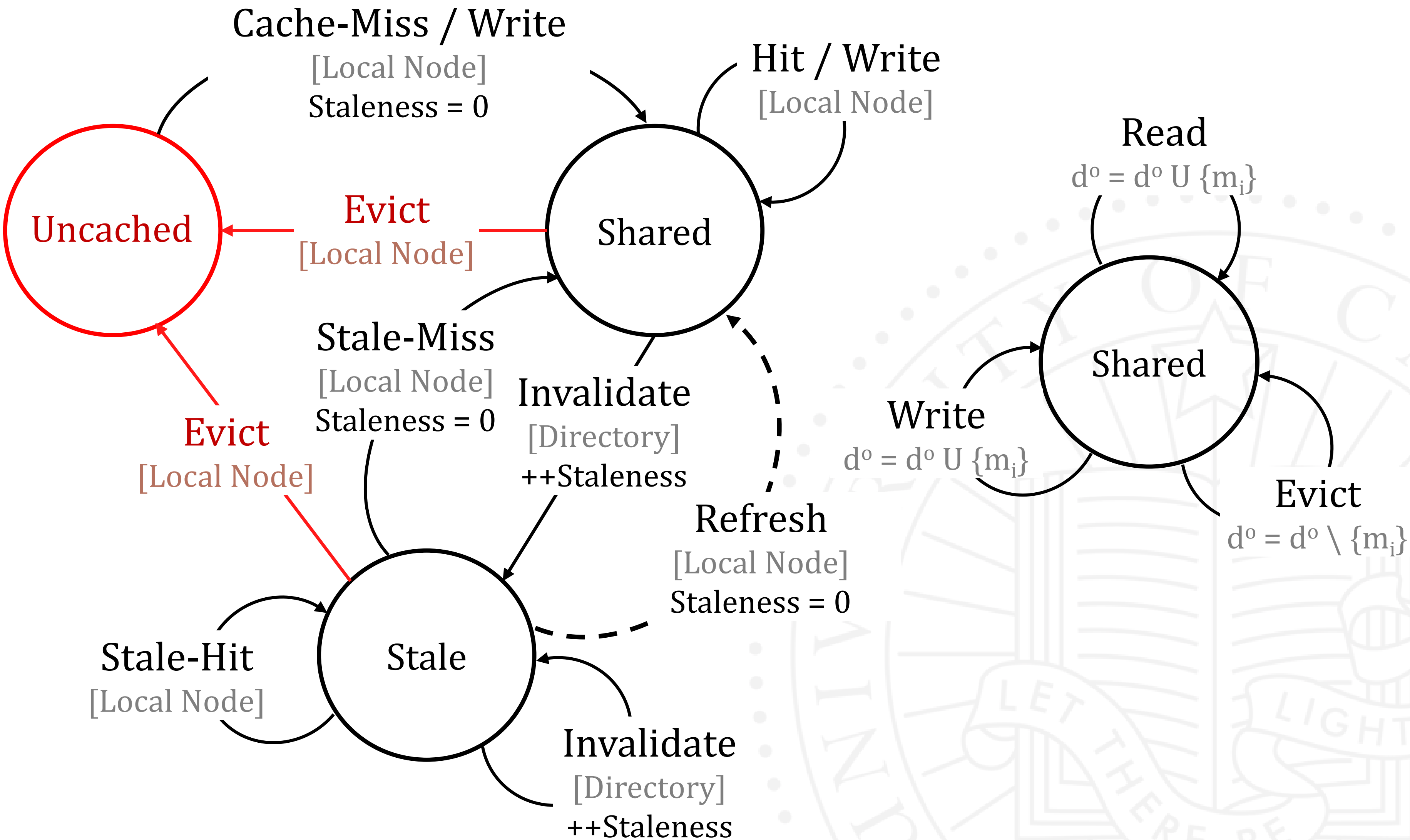


Relaxed Consistency Protocol

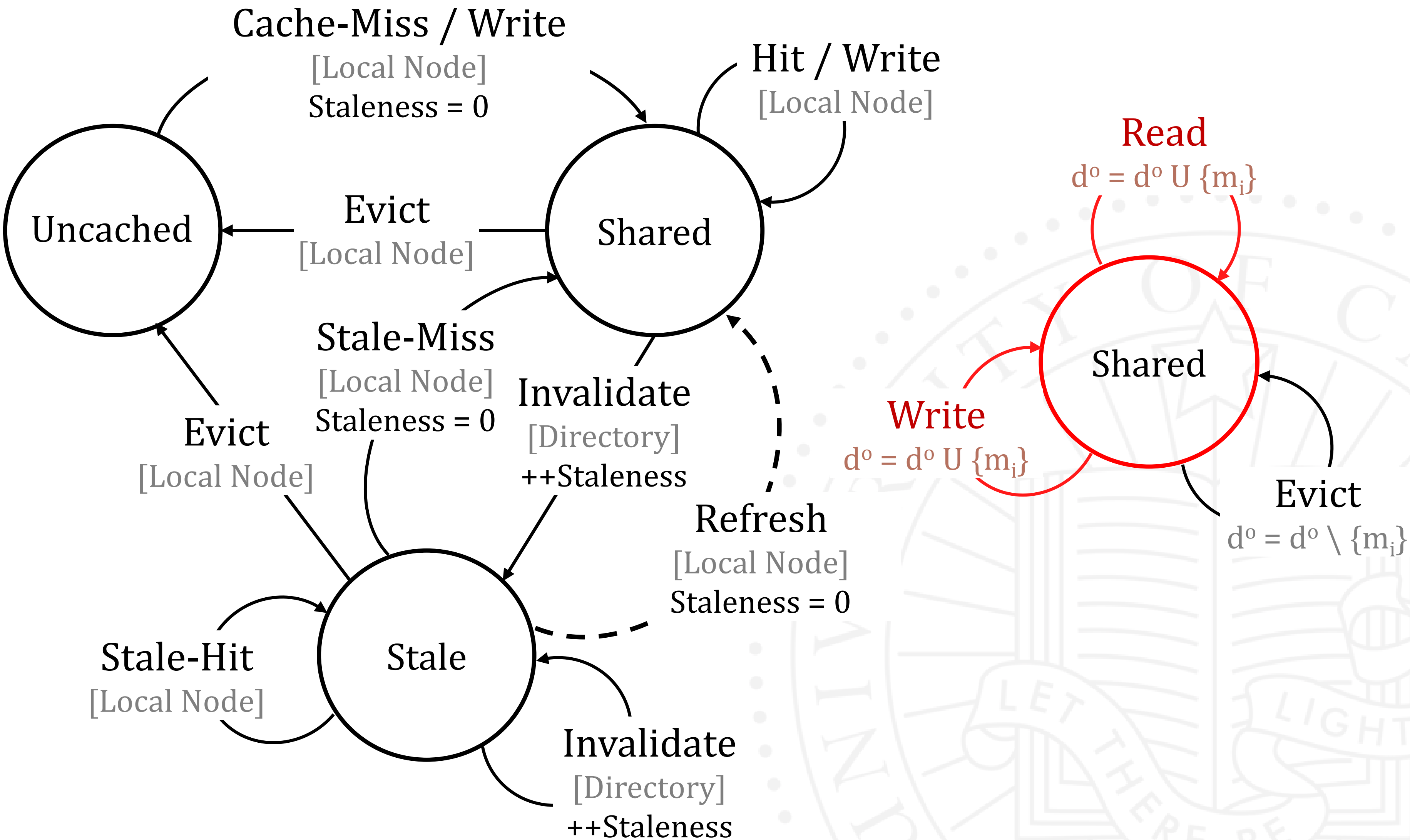




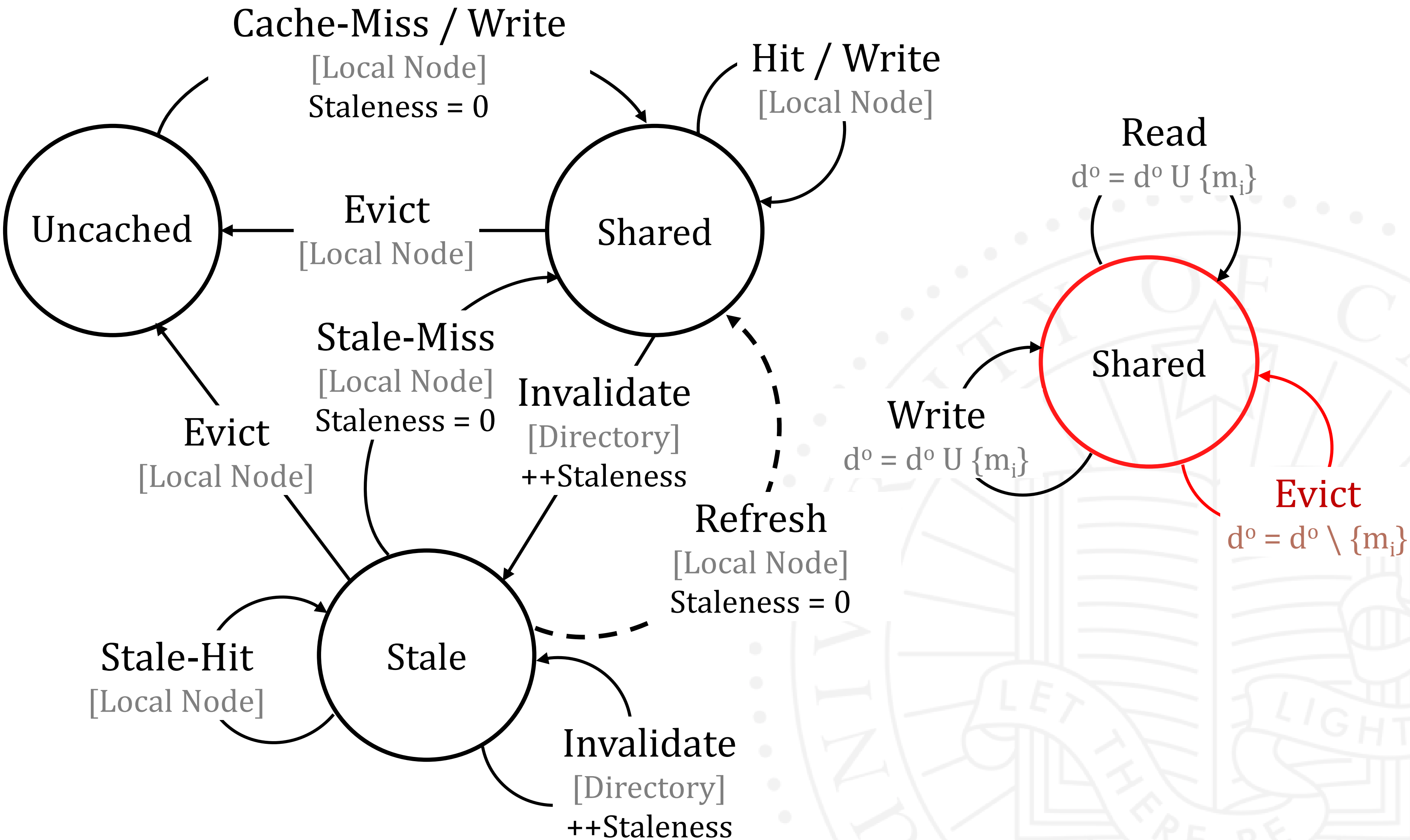
Relaxed Consistency Protocol



Relaxed Consistency Protocol



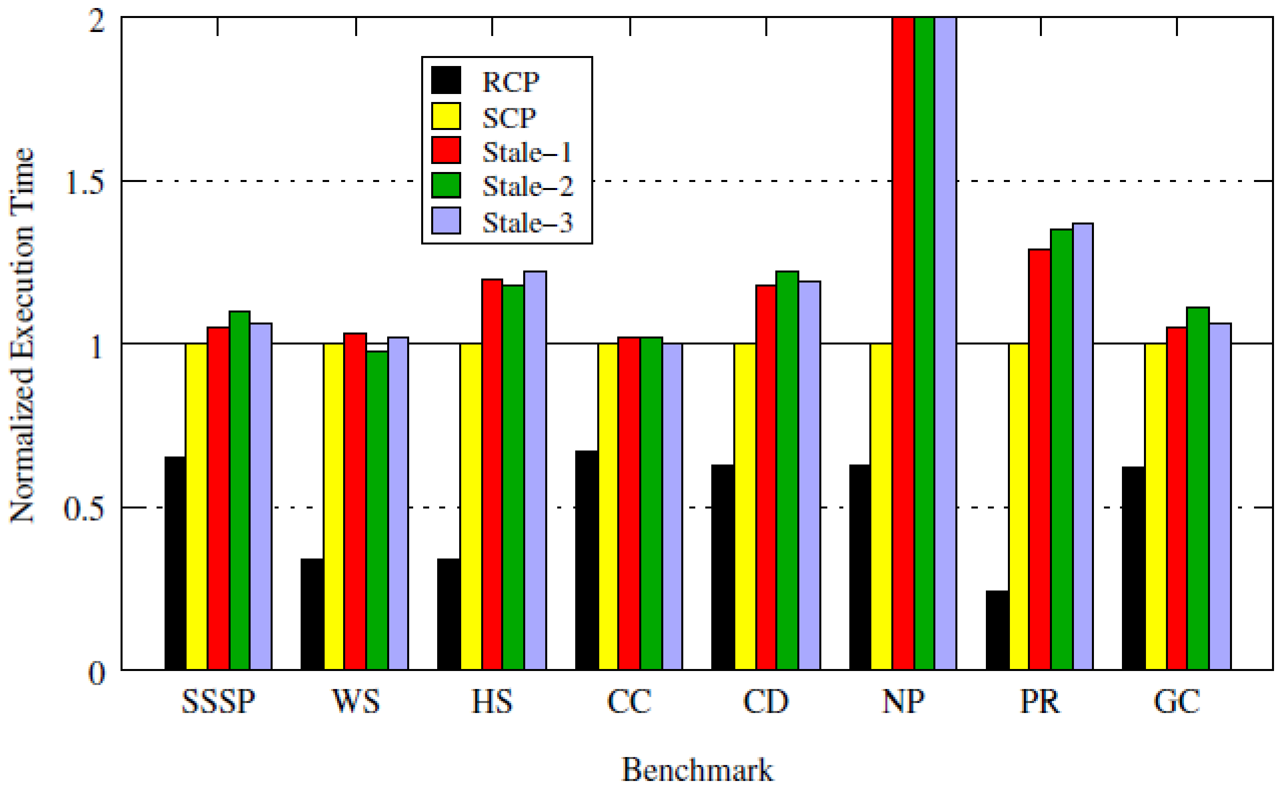
Relaxed Consistency Protocol



Implementation

- Similar to dyDSM [Koduru et al. 2013]
  - Object based
  - Protocol relaxes strict consistency
  - Graphs are partitioned using METIS [SISC 99]
- Runtime
  - Single Writer Model
  - Refresher threads block on refresh-queues
  - Compute threads populate refresh-queues

Performance

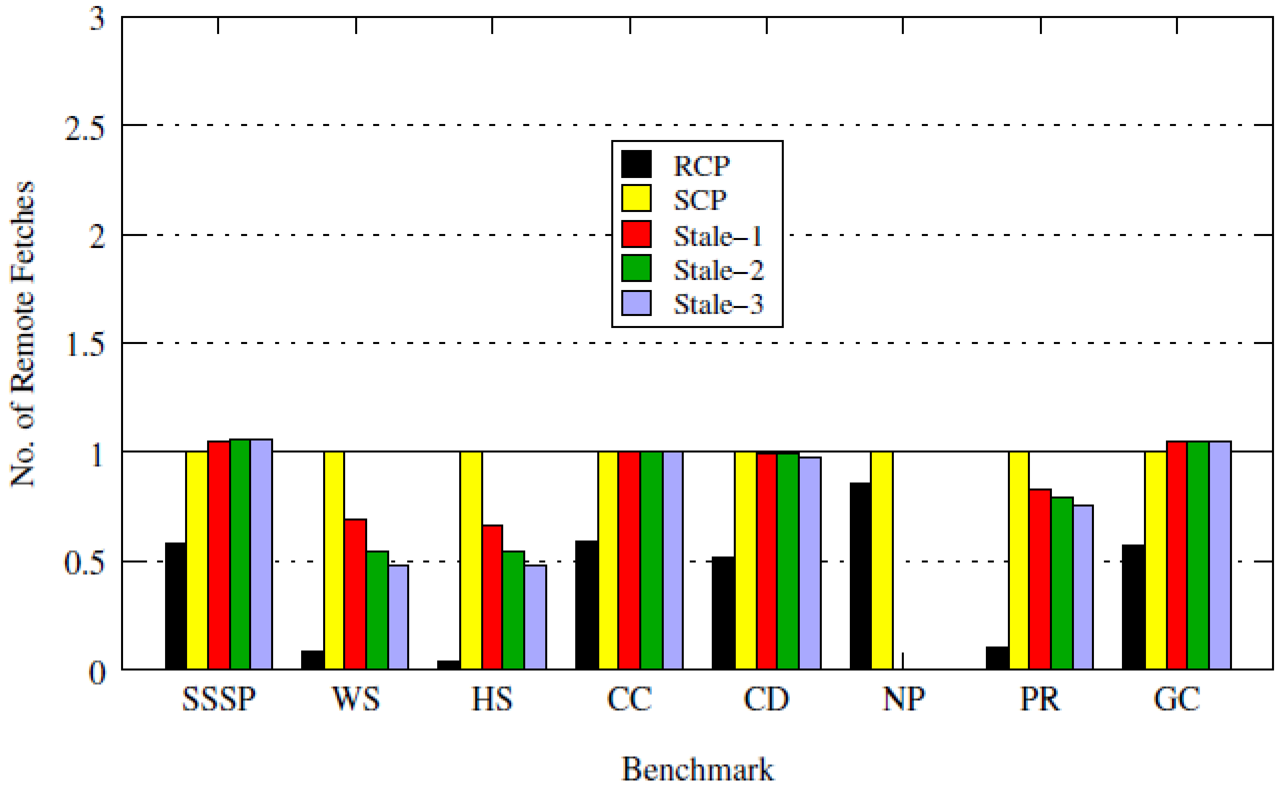


Pokec:
30M edges
1.6M vertices

AtmosModl:
10M edges
1.4M vertices

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

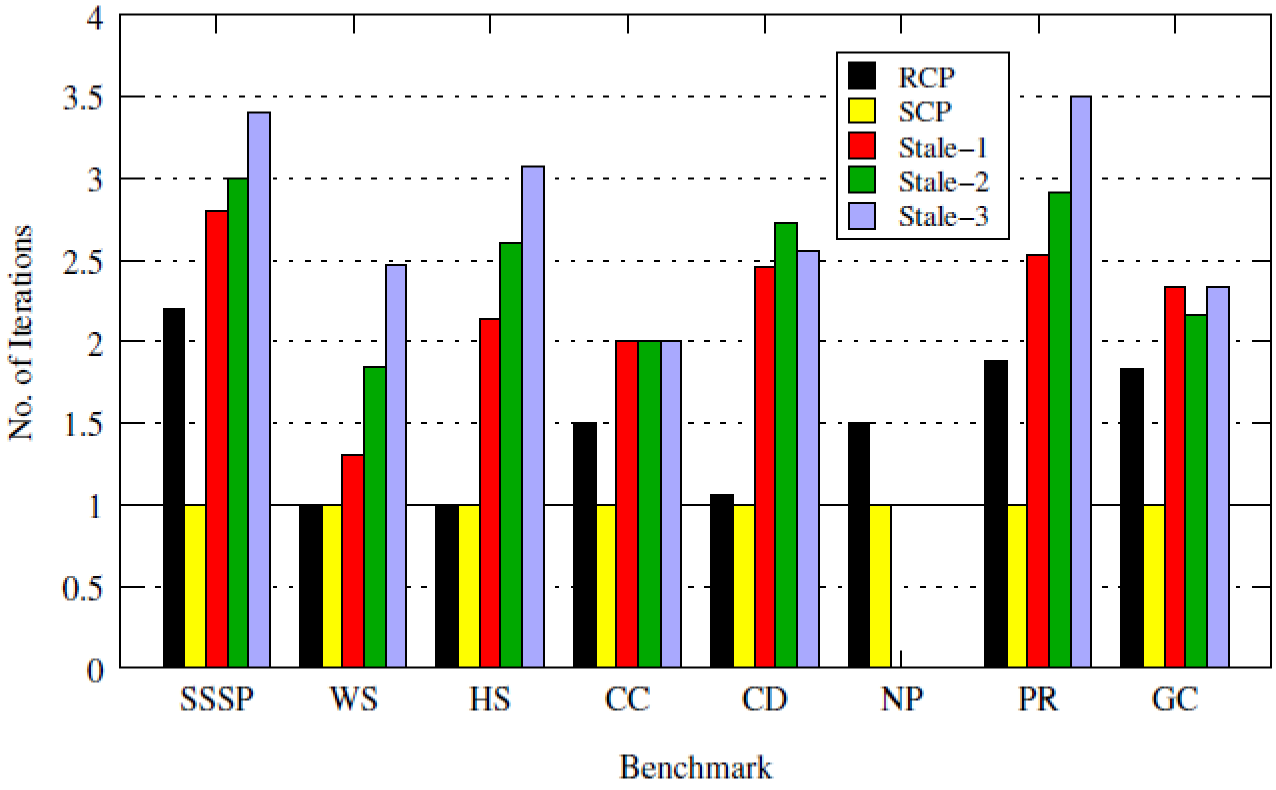
Performance



RCP blocks for 41% of remote fetches

Best Stale-n blocks for 85% of remote fetches

Performance

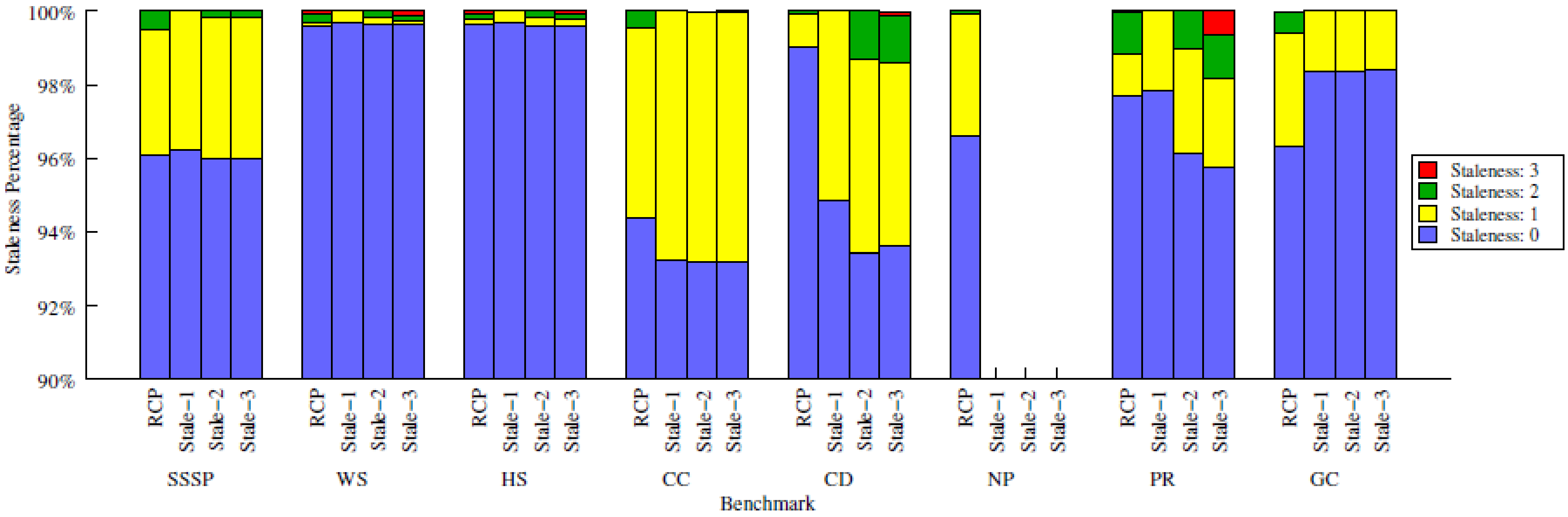


RCP requires 49% more iterations

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

Performance

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



GraphLab

		SSSP	PR	GC	CC
Orkut	RCP	161.88	822.95	92.79	90.31
	GL	239.4	829.3	248.66	102.02
Live-Journal	RCP	21.73	343.96	17.44	22.09
	GL	15.7	295.1	x	66.99
Pokec	RCP	9.47	169.47	8.81	7.1
	GL	8.7	159.9	173.47	40.52
Higgs-Twitter	RCP	2.50	15.64	3.60	4.10
	GL	5.5	x	263.45	16.21

- RCP performs better for non power-law graphs
- RCP is orthogonal to GraphLab

Conclusion

- Relaxing consistency is useful
  - With controlled use of staleness
- Prior DSMs:
  - Efficient (delta coherence & strict consistency)
- Graph Processing Frameworks
  - Easier to code (Pregel, GraphLab & PowerGraph)