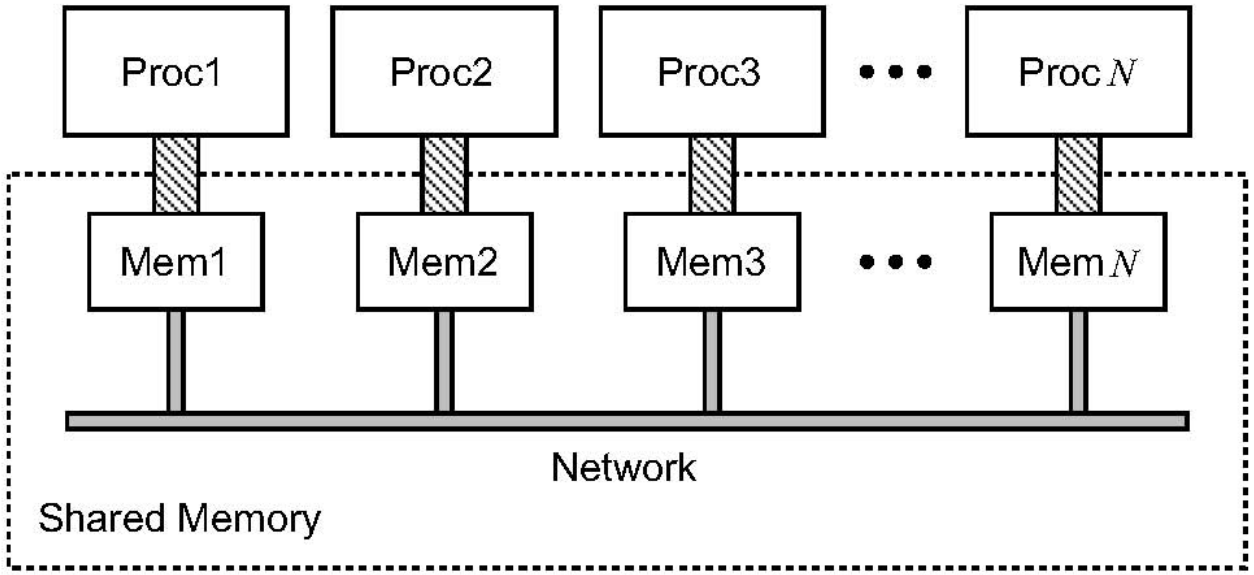
Release Consistency: Treadmarks

Haibo Chen

Credit: slides adapted from Jun Lee

Review: DSM (distributed shared memory)

- A **software system** for parallel computation
  - Shares distributed memories
  - Easier programming
    - Provide a single global address space

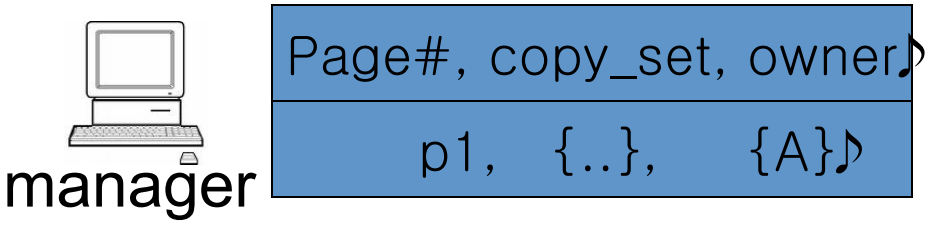


Recap: Ivy implementation

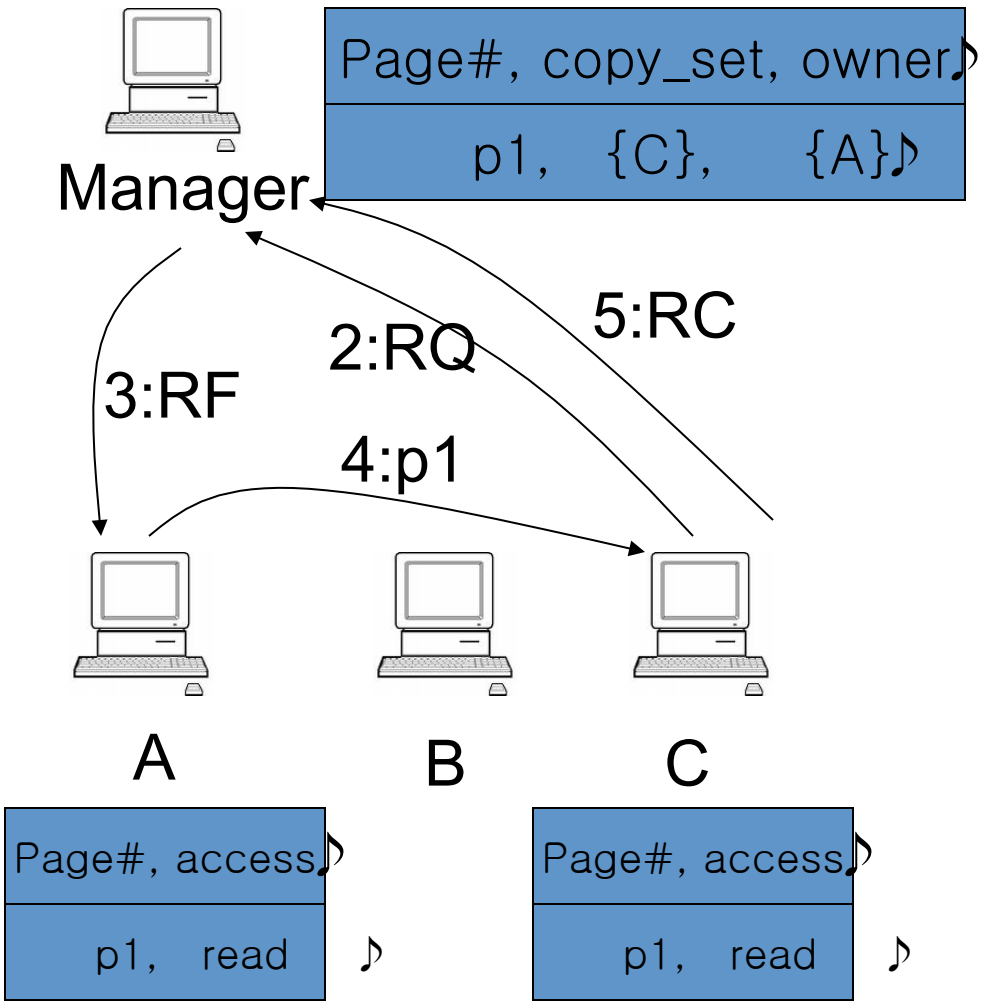
- The ownership of a page moves across nodes
  - Latest writer becomes the owner
  - Why?

- Challenge:
  - how to find the owner of a page?
  - how to ensure one owner per page?
  - How to ensure all cached pages are invalidated?

Recap-Ivy: centralized manager

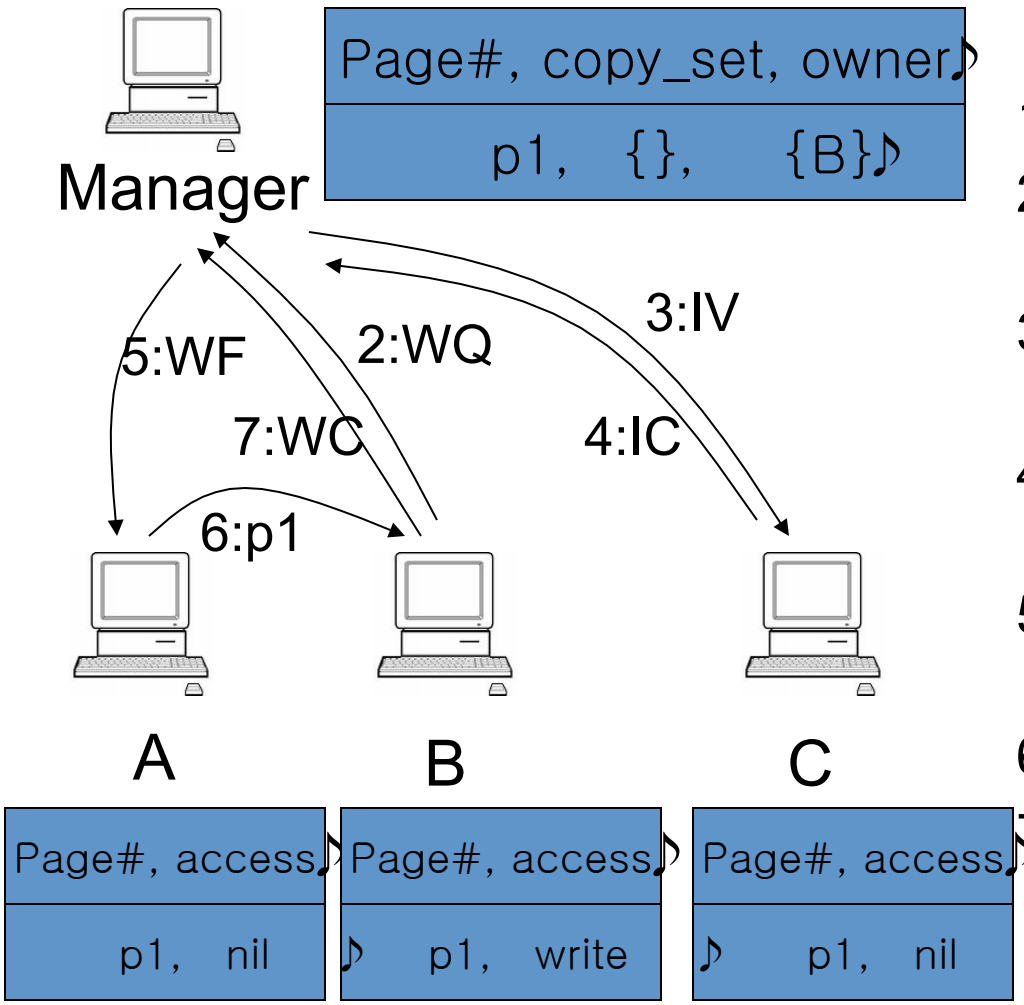


Recap-Ivy: read



1. Page fault for p1 on C
2. C sends RQ(read request) to M
3. M sends RF(read forward) to A, M adds C to copy_set
4. A sends p1 to C, C marks p1 as read-only
5. C sends RC(read confirmation) to M

Recap-Ivy: write



1. Page fault for p1 on B
2. B sends WQ(write request) to M
3. M sends IV(invalidate) to copy_set = {C}
4. C sends IC(invalidate confirm) to M
5. M clears copy_set, sends WF(write forward) to A
6. A sends p1 to B, clears access
7. B sends WC(write confirmation) to M

♪♪

Ivy invariants?

- Every page has exactly one current owner
- Current owner has a copy of the page
- If mapped r/w by owner, no other copies
- If mapped r/o by owner, identical to other copies
- Manager knows about all copies

DSM (distributed shared memory)

- No widely available DSM implementations

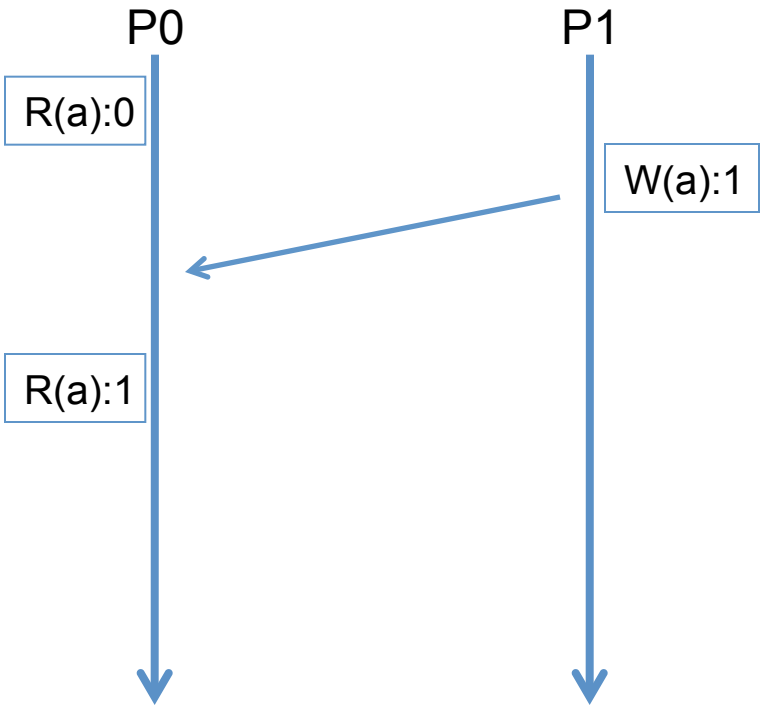
- Issues with IVY
  - In-house research platforms
  - Kernel modifications
  - Poor performance
    - Mimicking consistency protocols of hardware
    - False sharing

Treadmarks

- **Objectives**
  - Commercially available workstations and OS
    - Standard Unix system on DECstation
  - Efficient user-level DSM implementation
    - Reduce communication overhead
- **Design**
  - LRC (lazy release consistency)
  - Multiple writer protocol
  - Lazy diff creation

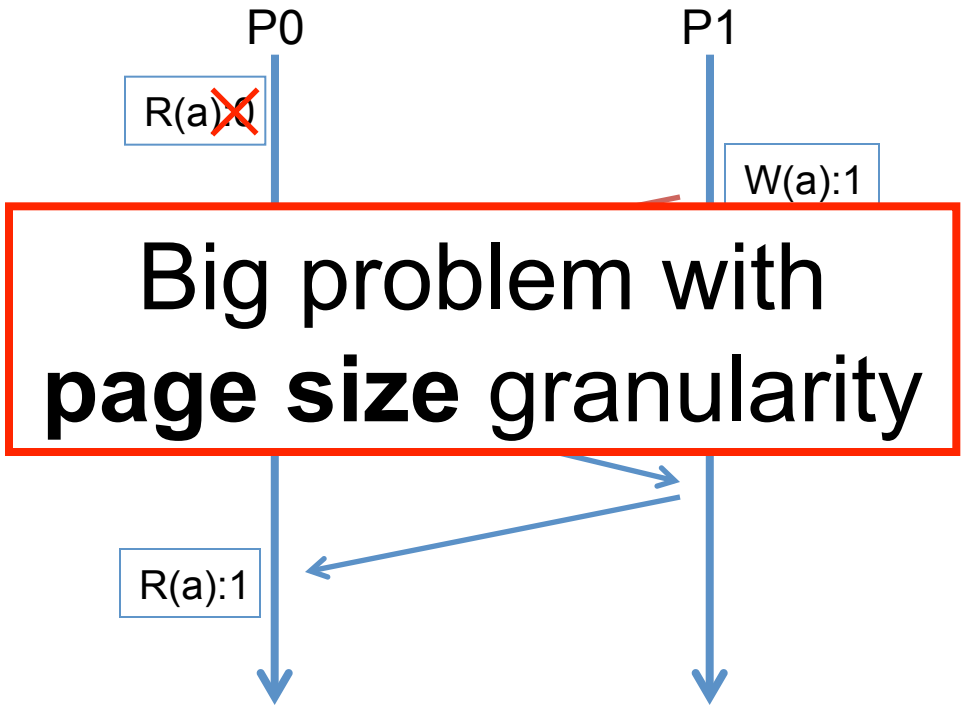
Consistency protocol (SC)

- Sequential Consistency
  - Every write visible “immediately”
  - Single writer



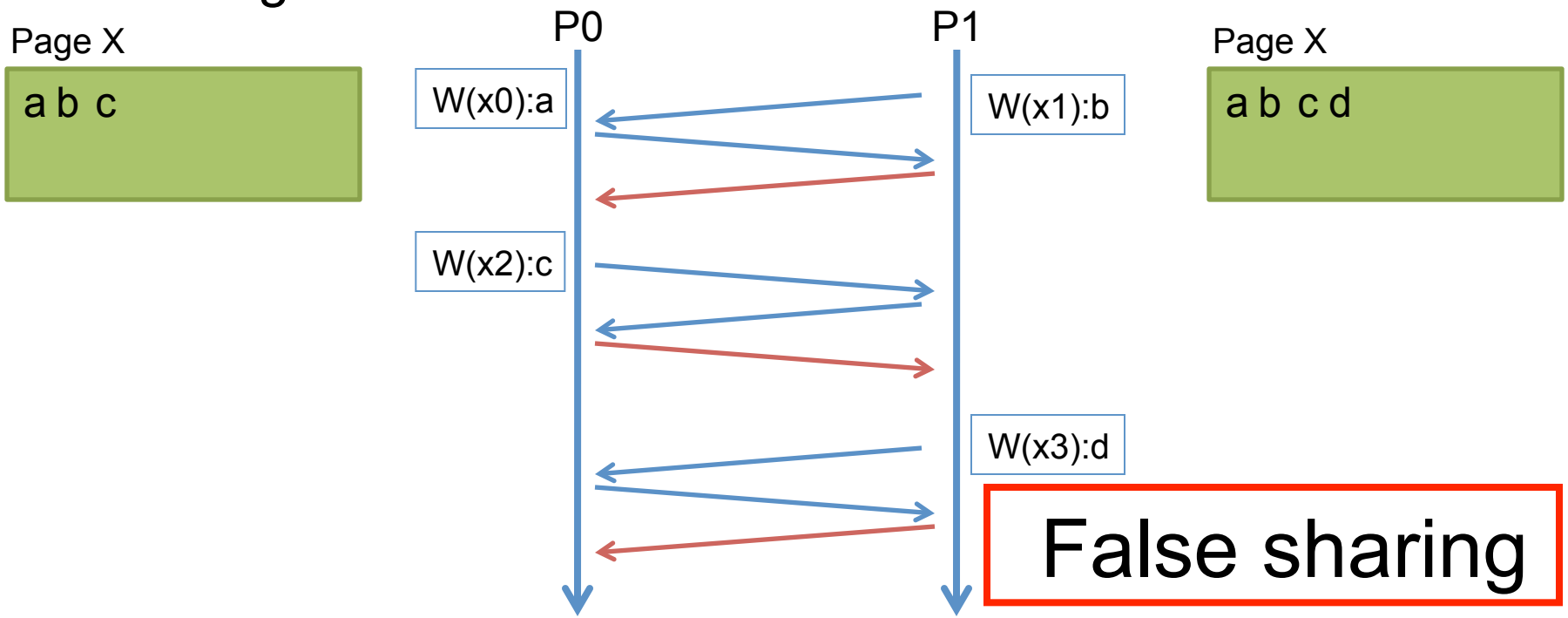
Consistency protocol (SC)

- Sequential Consistency
  - Every write visible “immediately”
  - Single writer



Consistency protocol (SC)

- Sequential Consistency
  - Every write visible “immediately”
  - Single writer



Consistency protocol (RC)

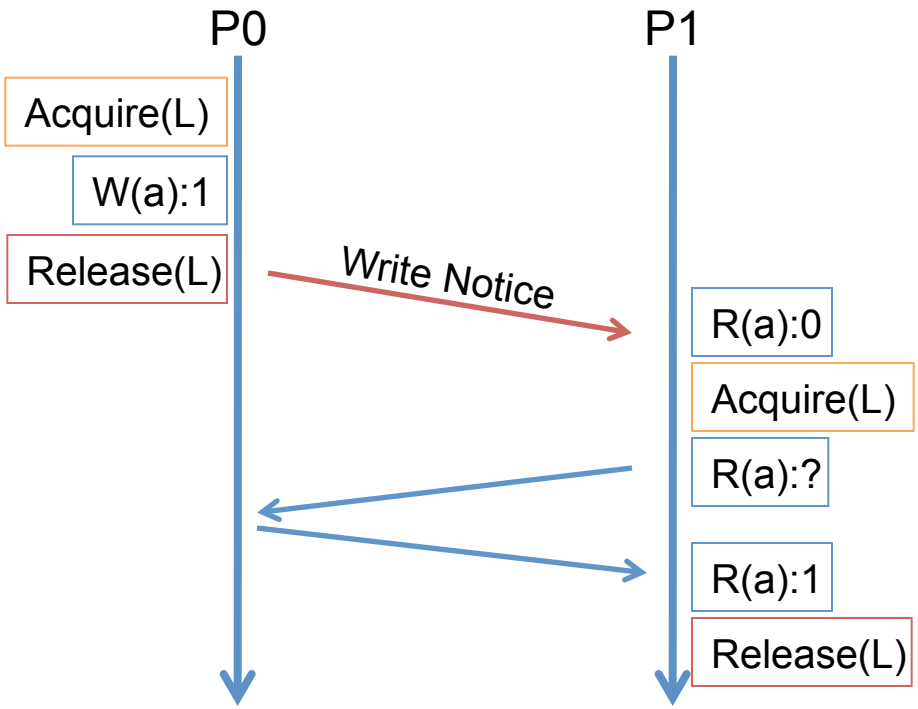
- Release Consistency
  - Relaxed memory consistency model
    - delay making its changes visible to other processors until certain synchronization accesses occurs
  - Synchronization points
    - Acquire(), Release() (similar to locks, barriers)
  - Two types
    - ERC (eager), LRC (lazy)

Consistency protocol (RC)

- Release Consistency
  - Acquire() and release() are sequentially consistent
    - Release() is performed after all previous operations have completed
  - Acquire() and release() pair between conflicting accesses
    - SC and RC produce the same results.

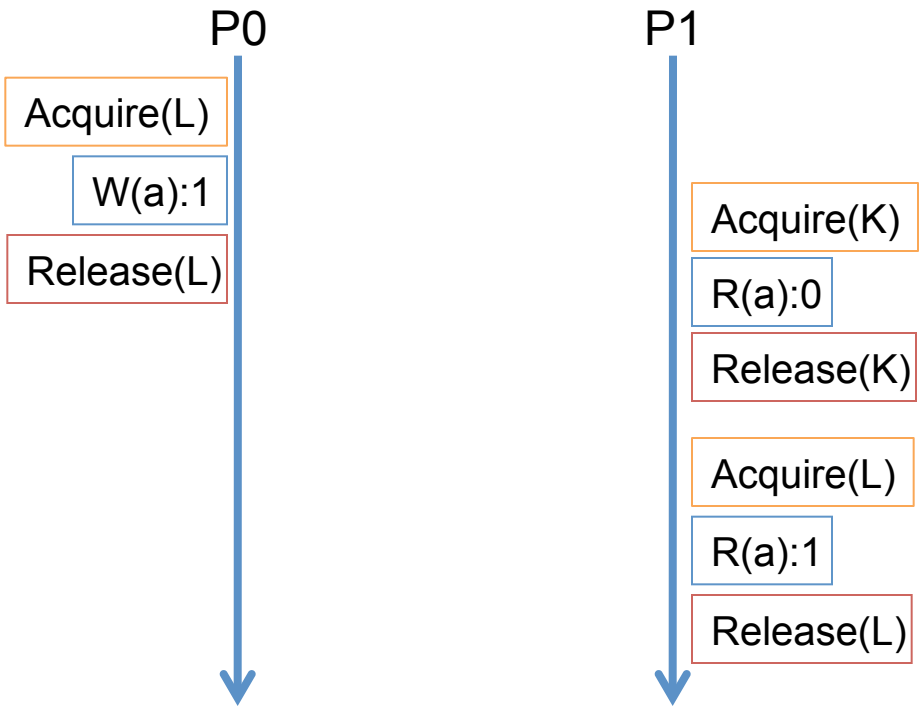
Consistency protocol (RC)

- ERC
  - Write information is delivered at the release point



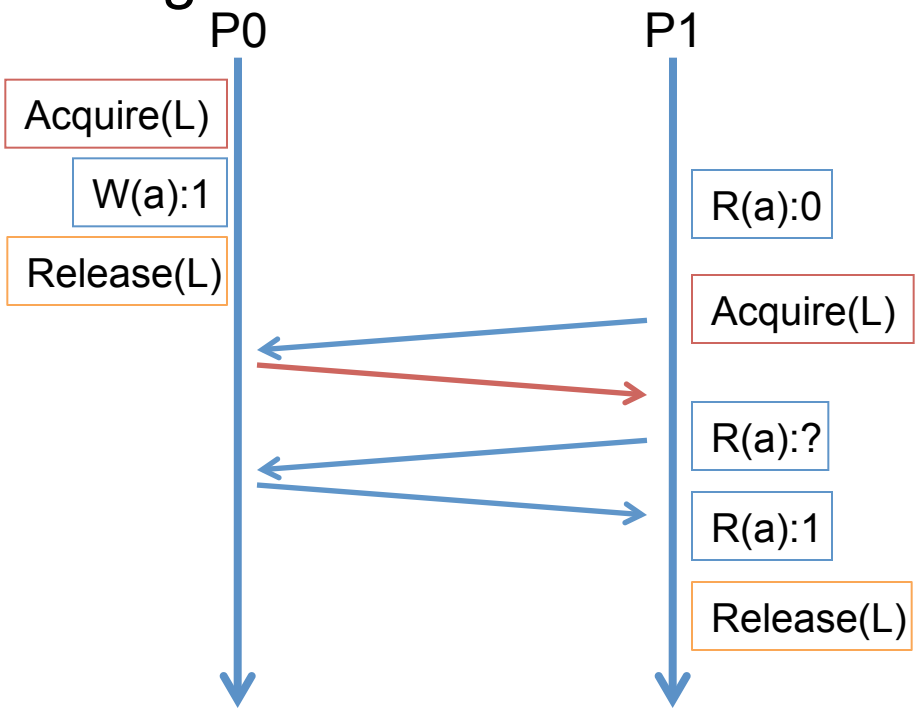
Consistency protocol (RC)

- ERC
  - Write information is delivered at the release point



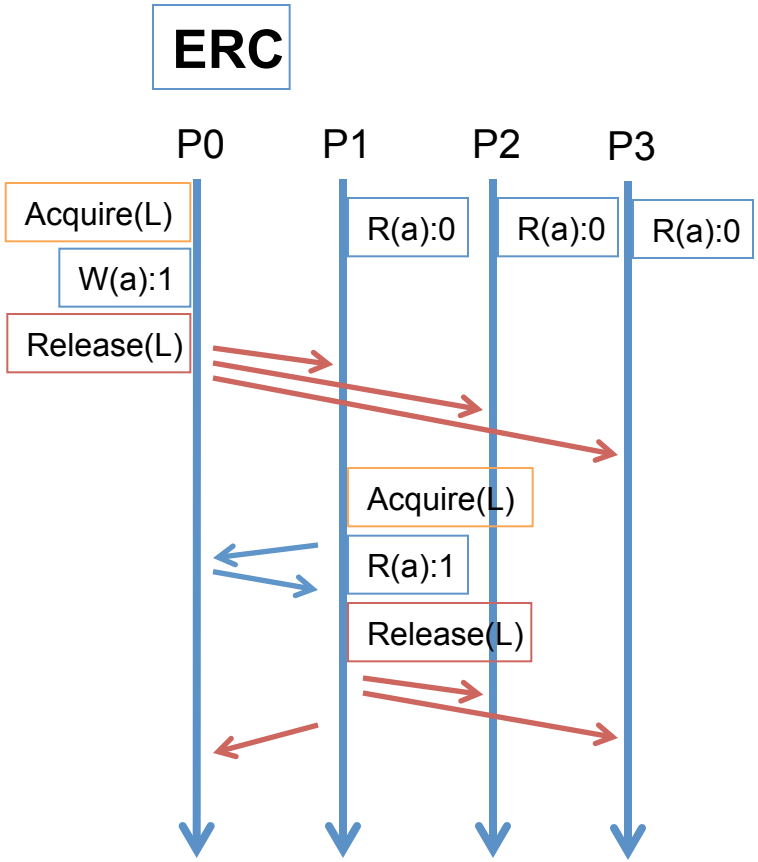
Consistency protocol (RC)

- LRC
  - The delivery is postponed until the acquire
  - Fewer messages than ERC



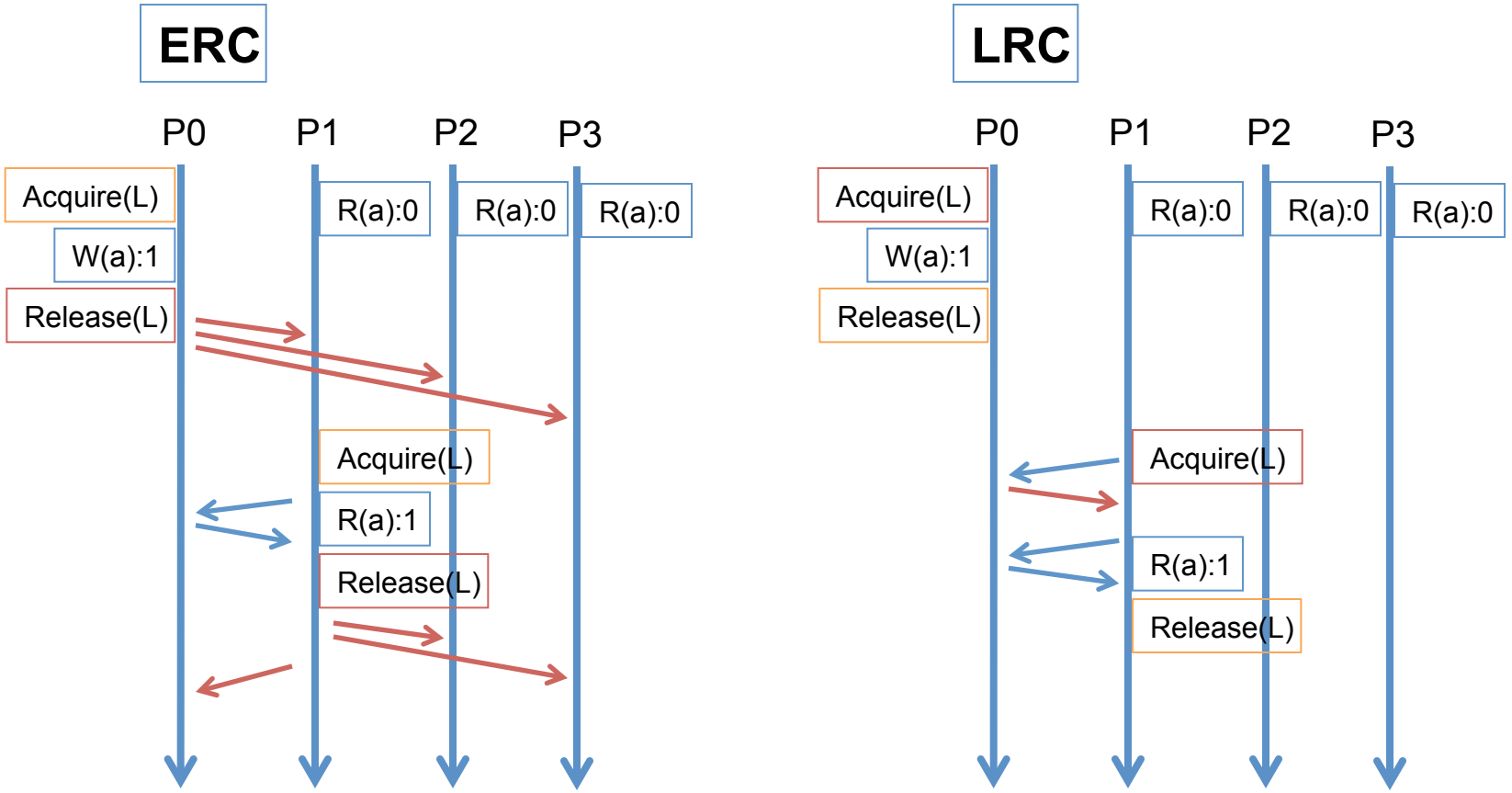
Consistency protocol (RC)

- ERC vs. LRC

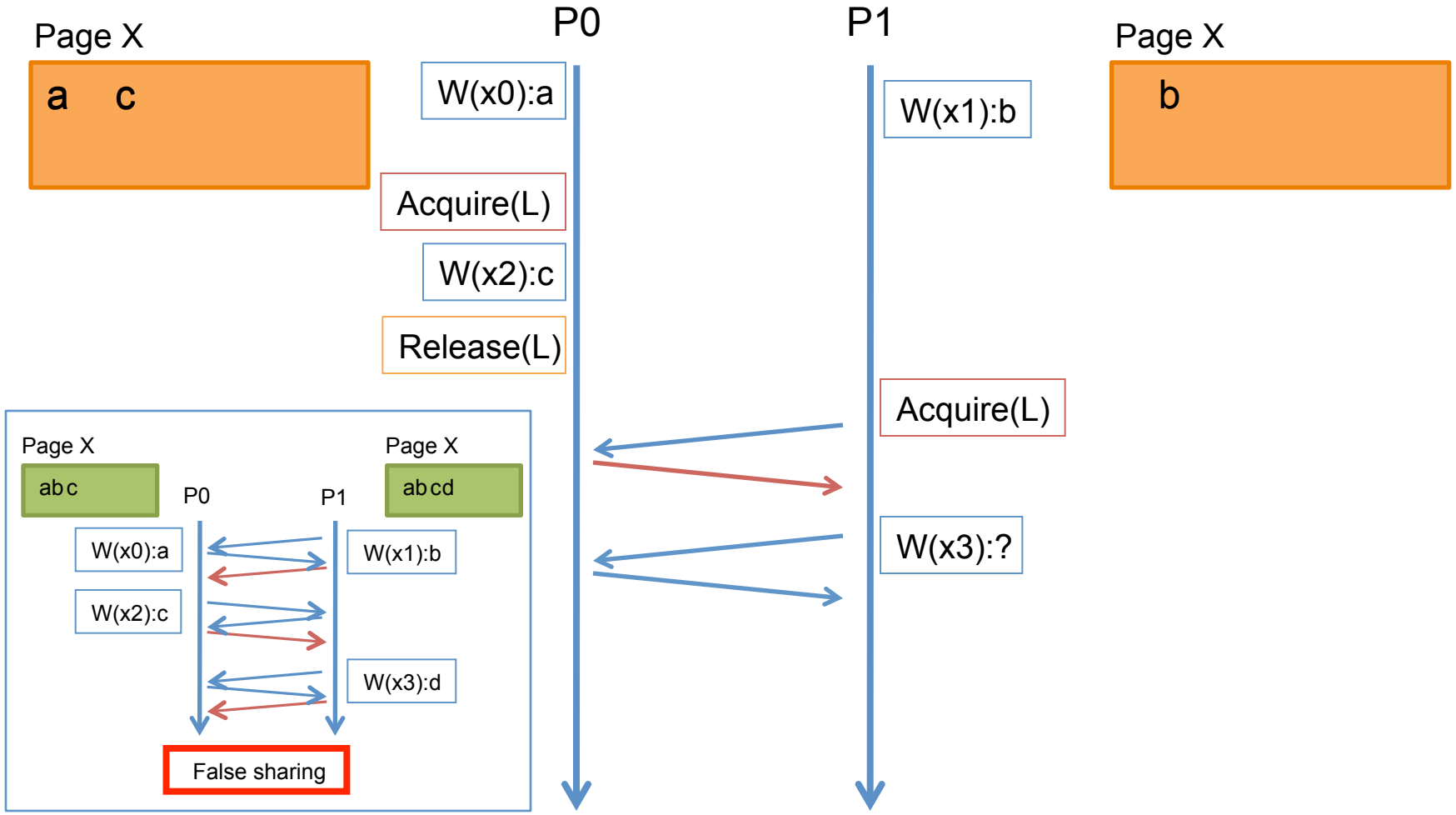


Consistency protocol (RC)

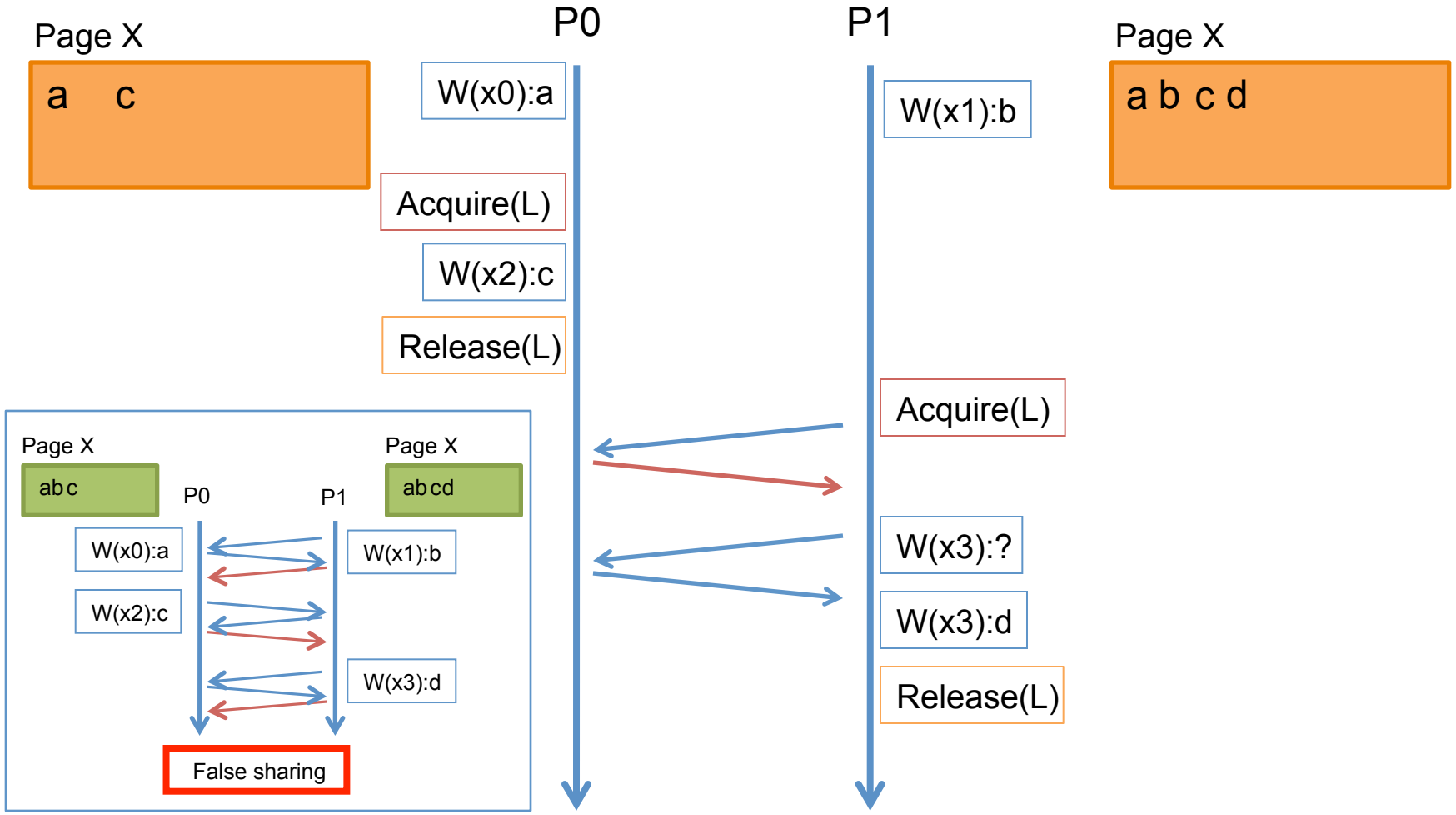
- ERC vs. LRC



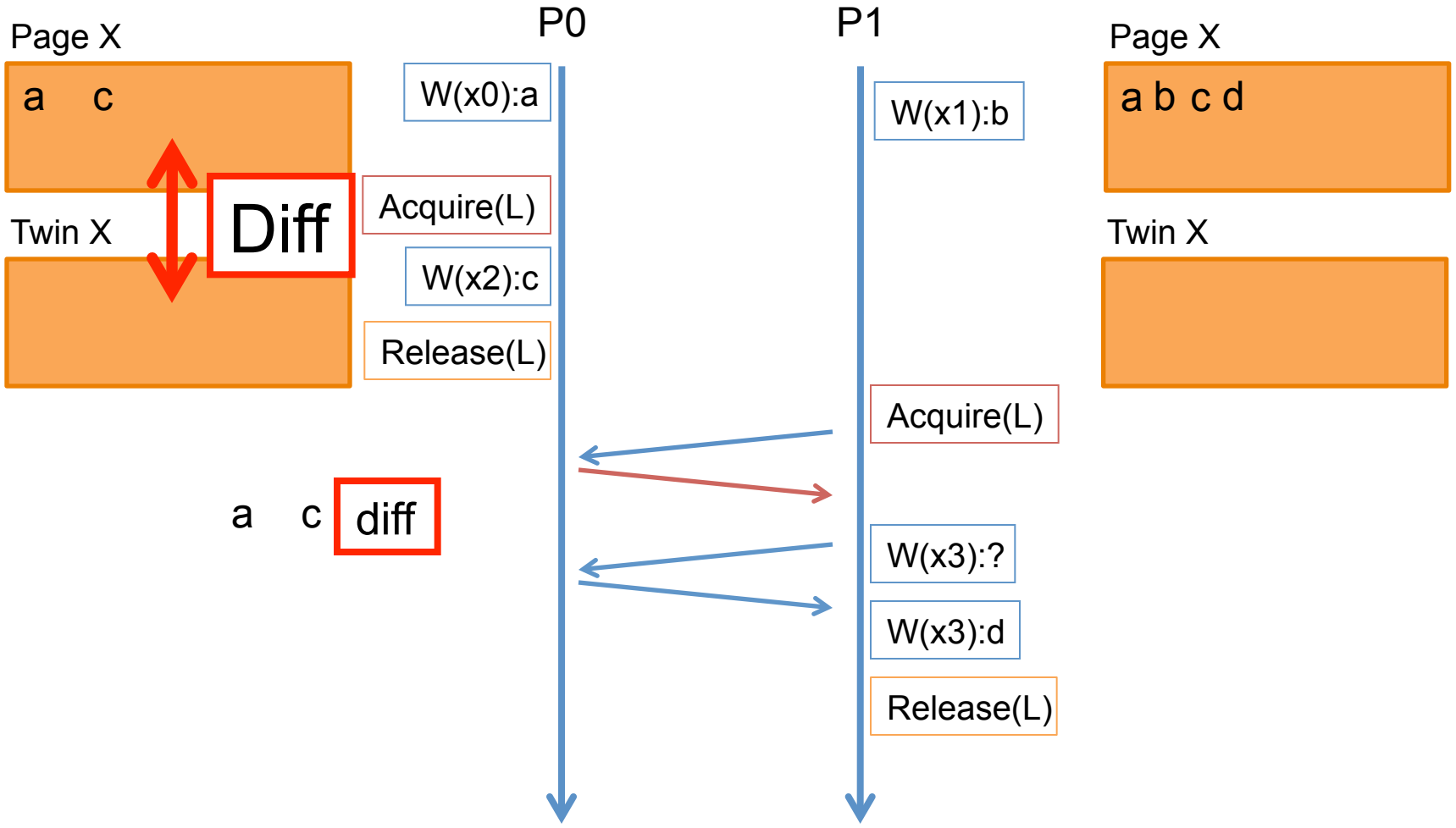
Multiple writer protocol



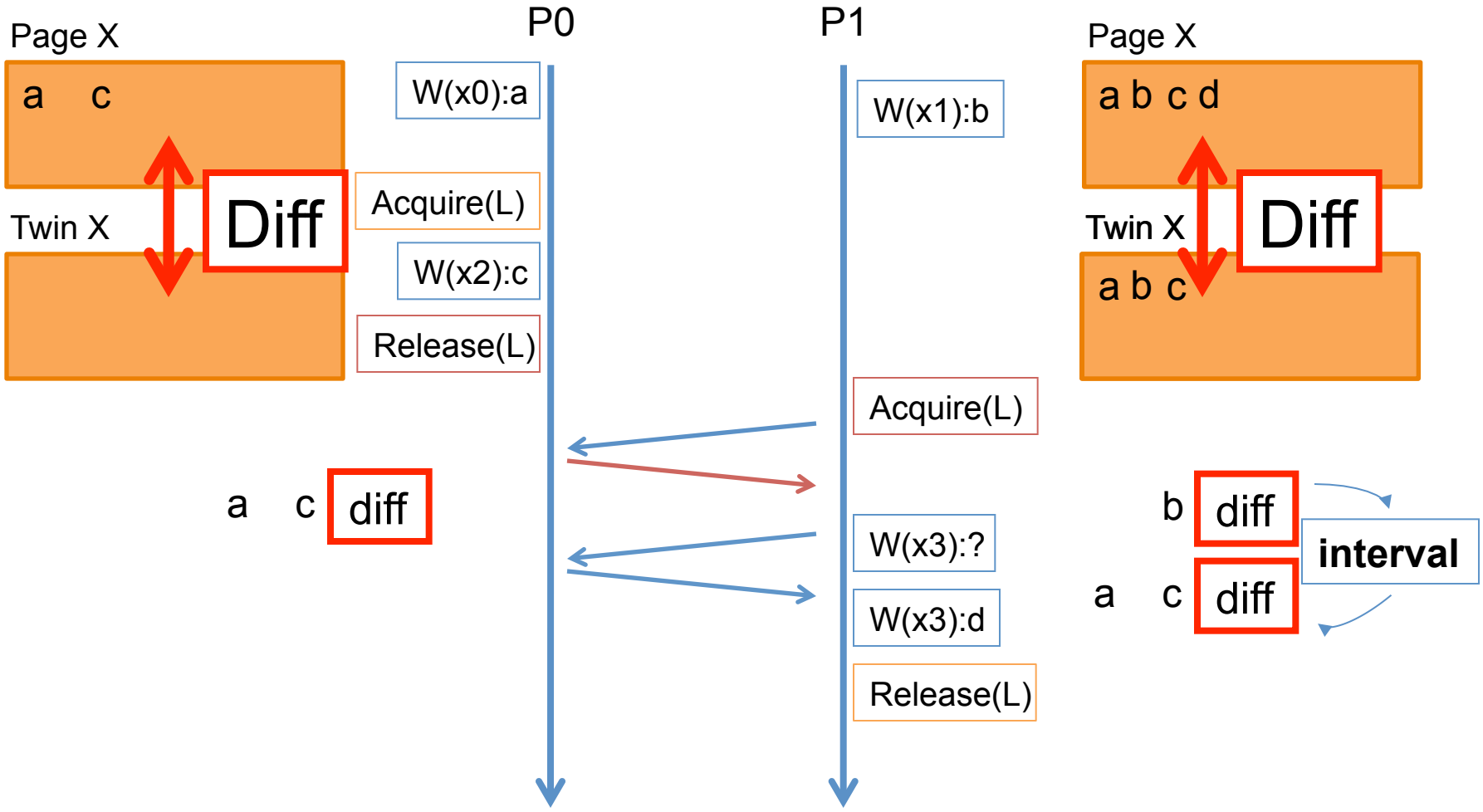
Multiple writer protocol



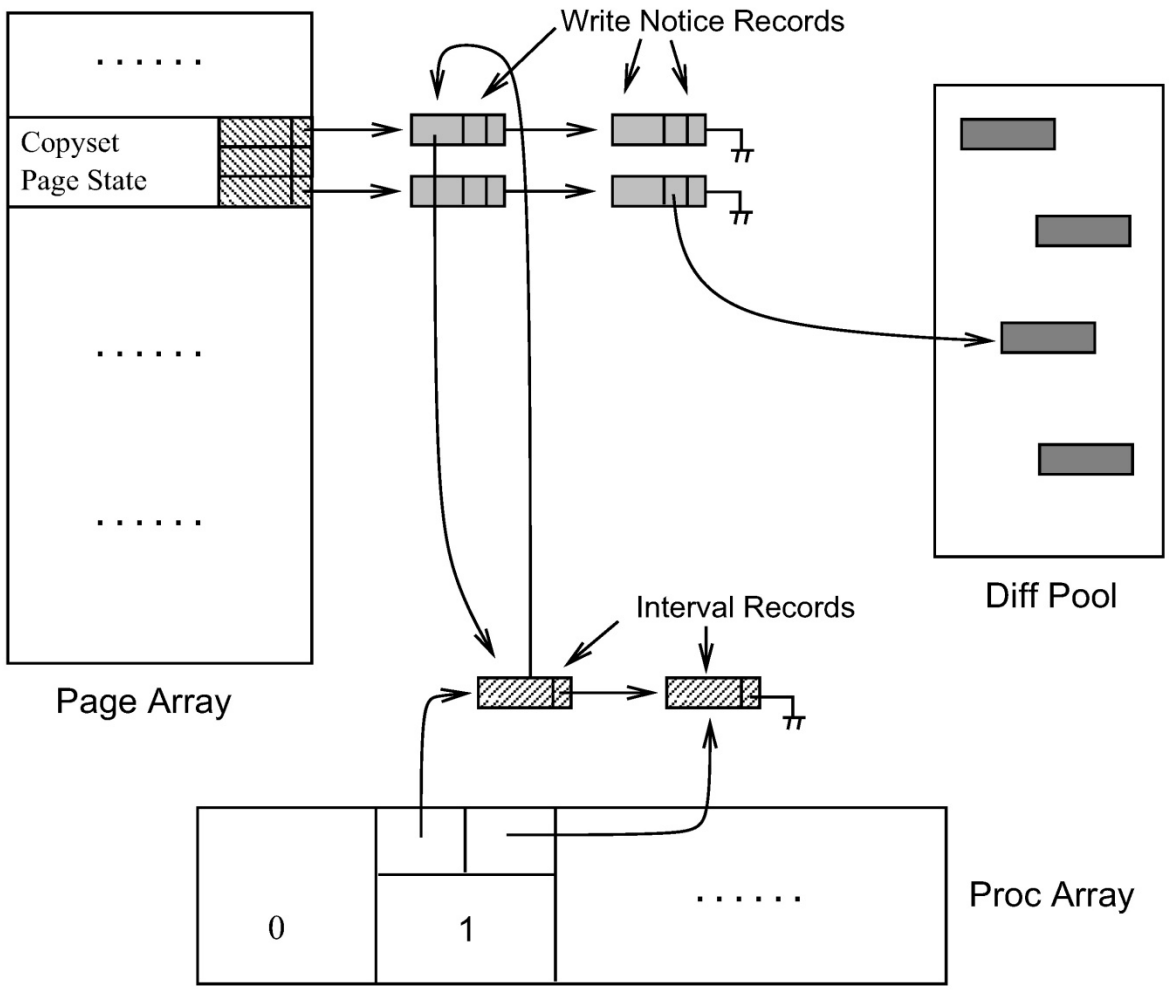
Twin and Diff



Twin and Diff



Implementation



Implementation

P0 side

P1 side

pages

. . .	
	P
. . .	
. . .	

Acq(L)

pages

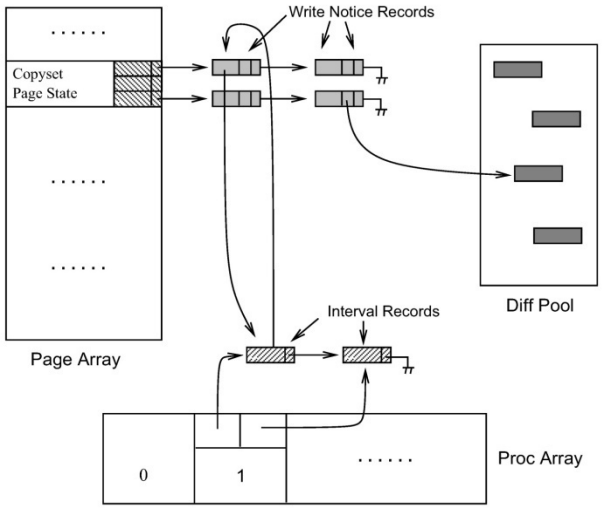
. . .	
	P
. . .	
. . .	

time stamp

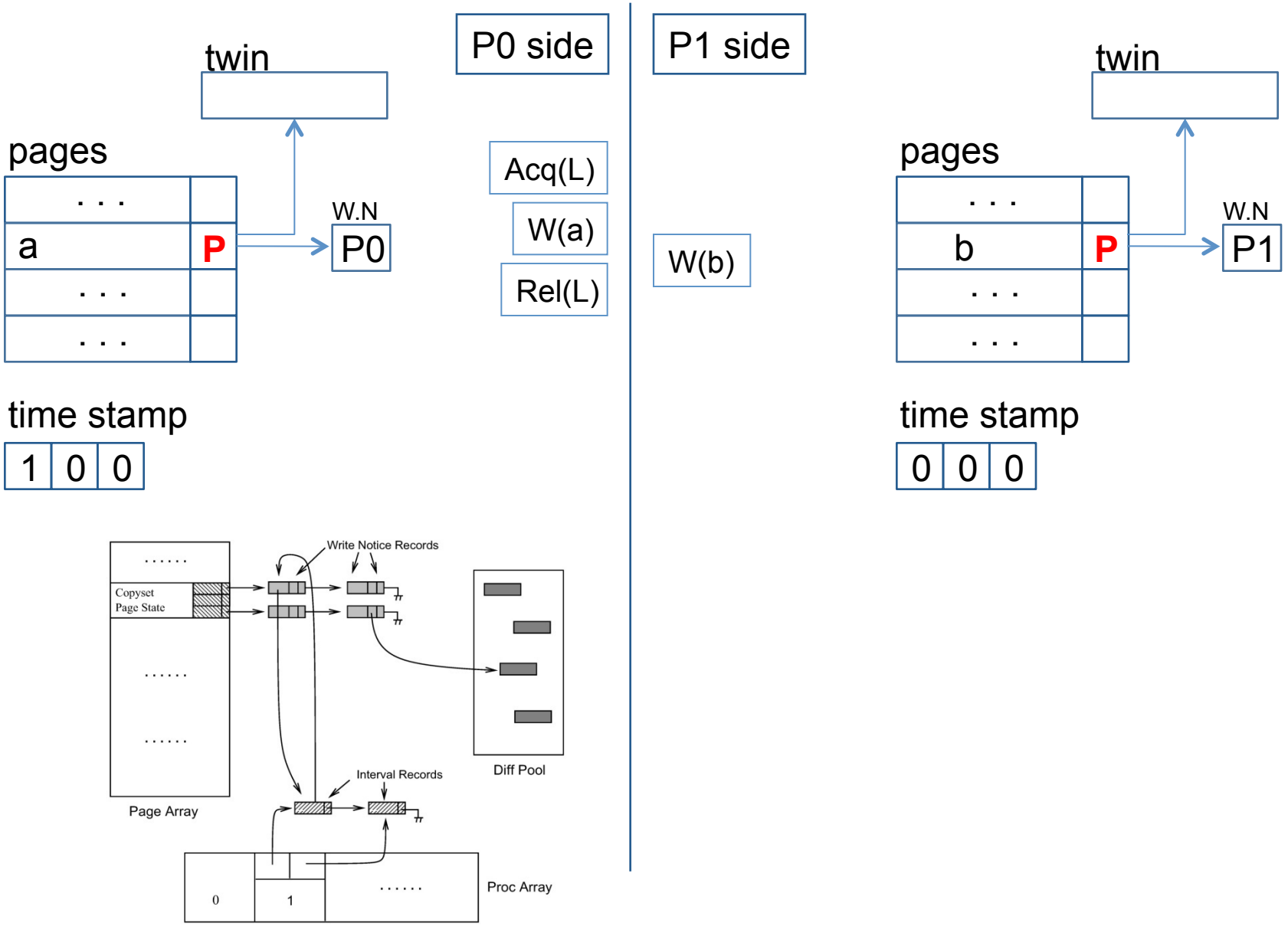
0	0	0

time stamp

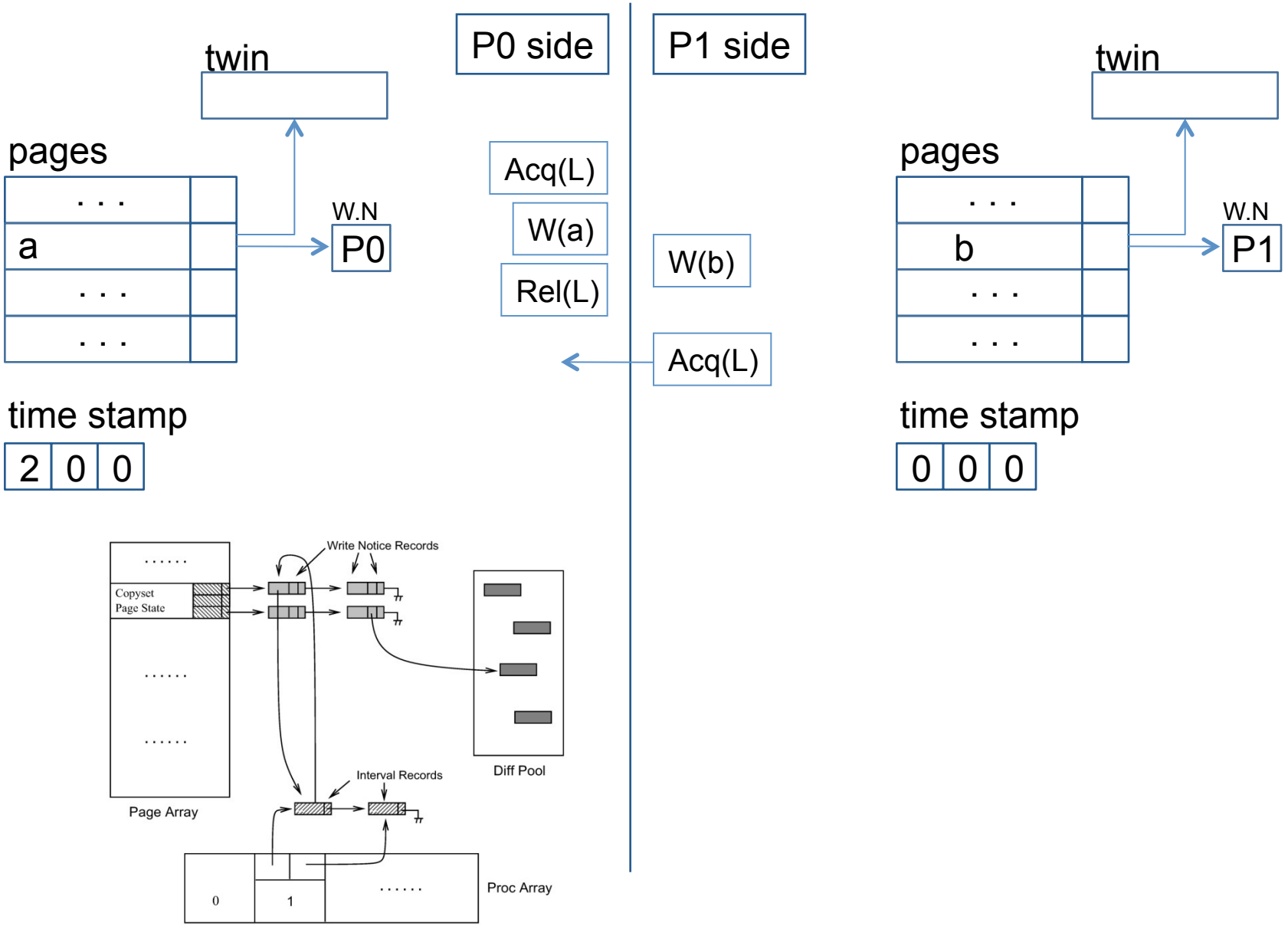
0	0	0



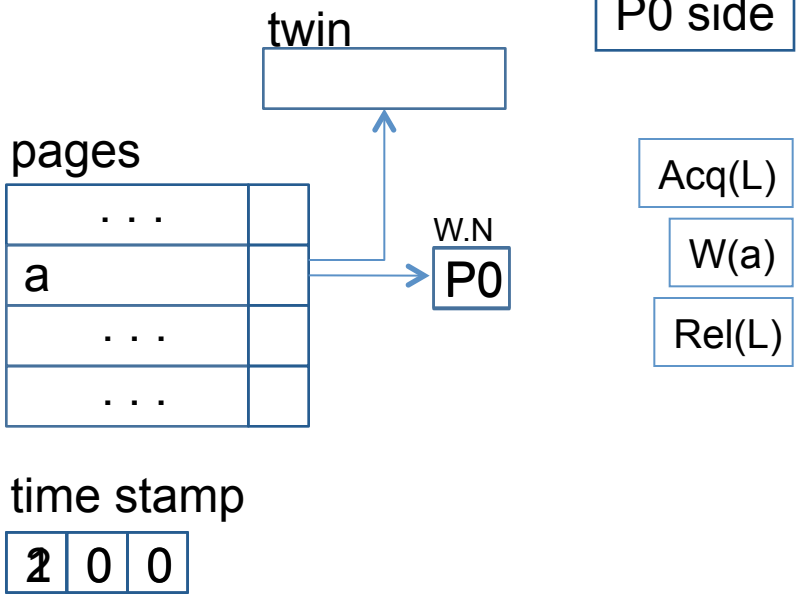
Implementation

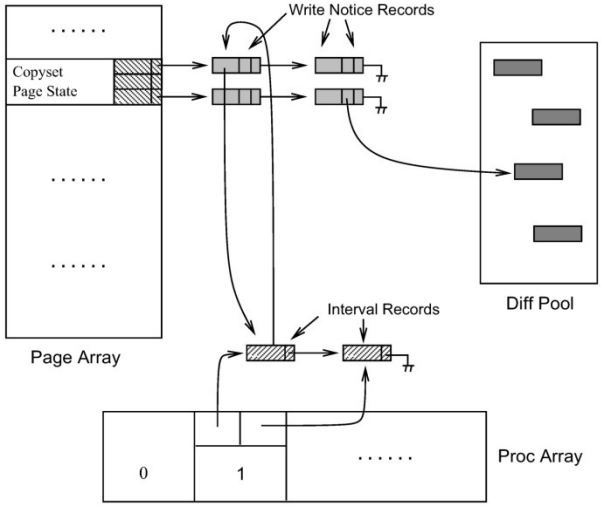


Implementation

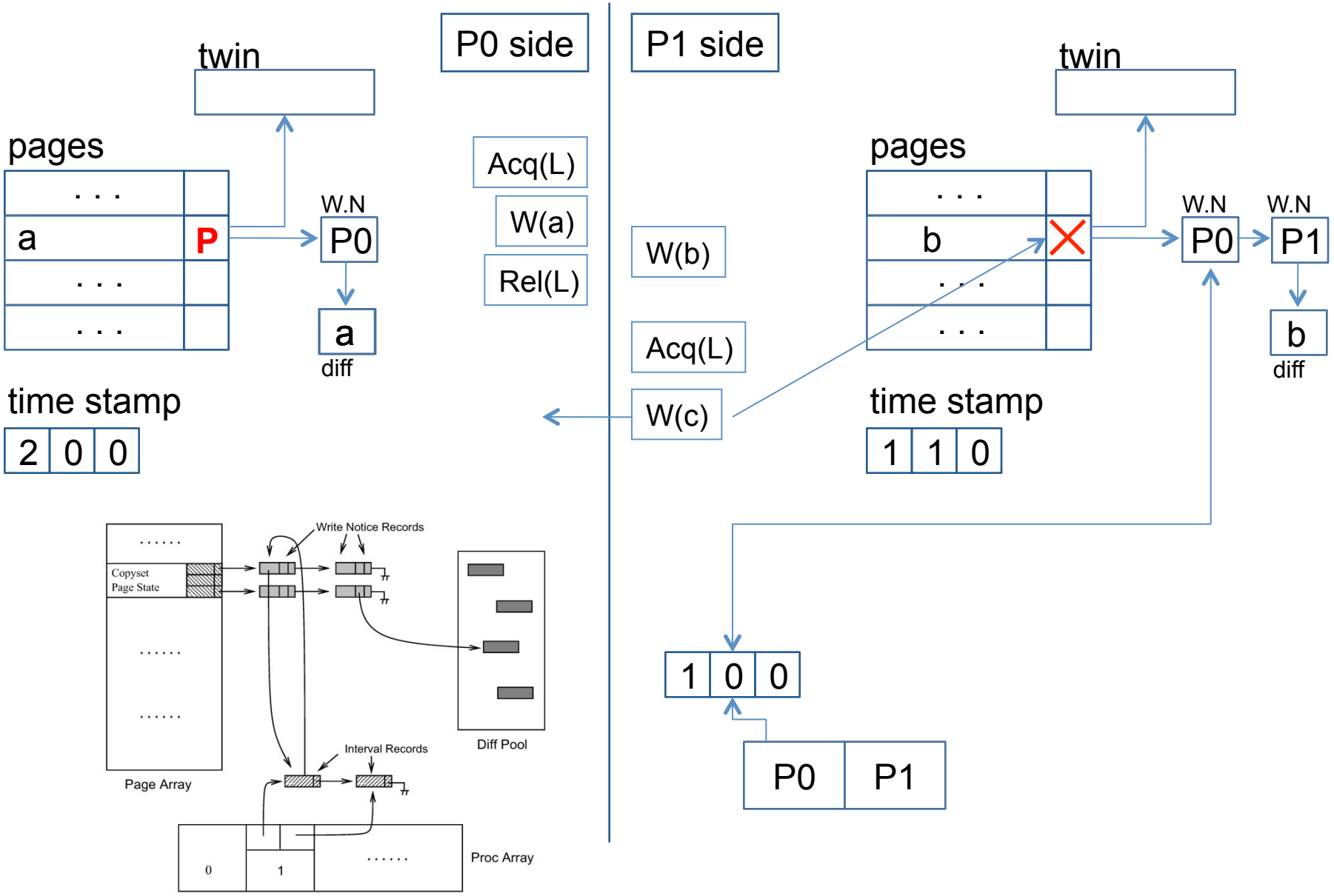


Implementation





Implementation



Implementation

P0 side

P1 side

pages

. . .	
a	P
. . .	
. . .	

W.N → P0 → a (diff)

Acq(L)

W(a)

Rel(L)

W(b)

Acq(L)

W(c)

Rel(L)

time stamp

2	0	0

twin

a b

pages

. . .	
a b c	P
. . .	
. . .	

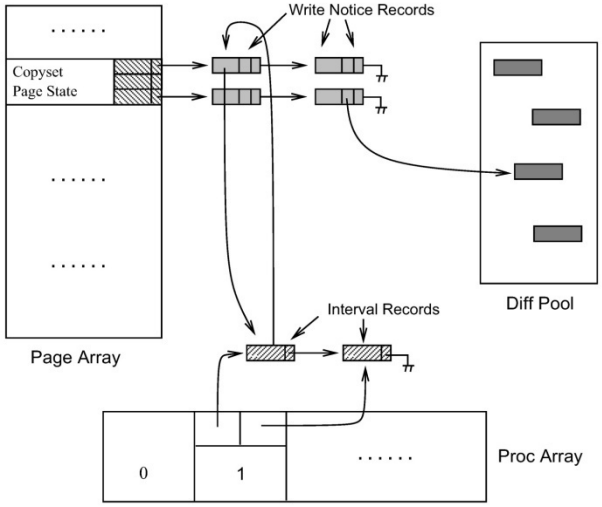
W.N → P1 → W.N → P0 → W.N → P1

P0 → a (diff)

P1 → b (diff)

time stamp

1	1	0



1	0	0

P0	P1

Implementation

P0 side

P1 side

pages

. . .	
a	P
. . .	
. . .	

W.N → P0 → a (diff)

Acq(L)

W(a)

Rel(L)

W(b)

Acq(L)

W(c)

Rel(L)

time stamp

2	0	0

twin

a b

pages

. . .	
a b c	
. . .	
. . .	

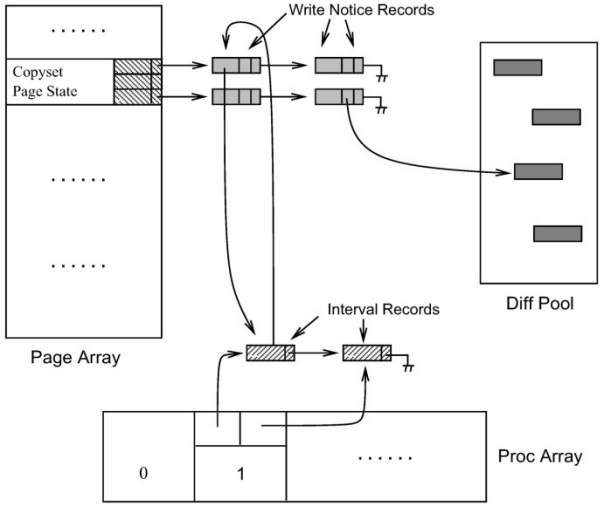
W.N → P1 → W.N → P0 → W.N → P1

P0: a (diff)

P1: b (diff)

time stamp

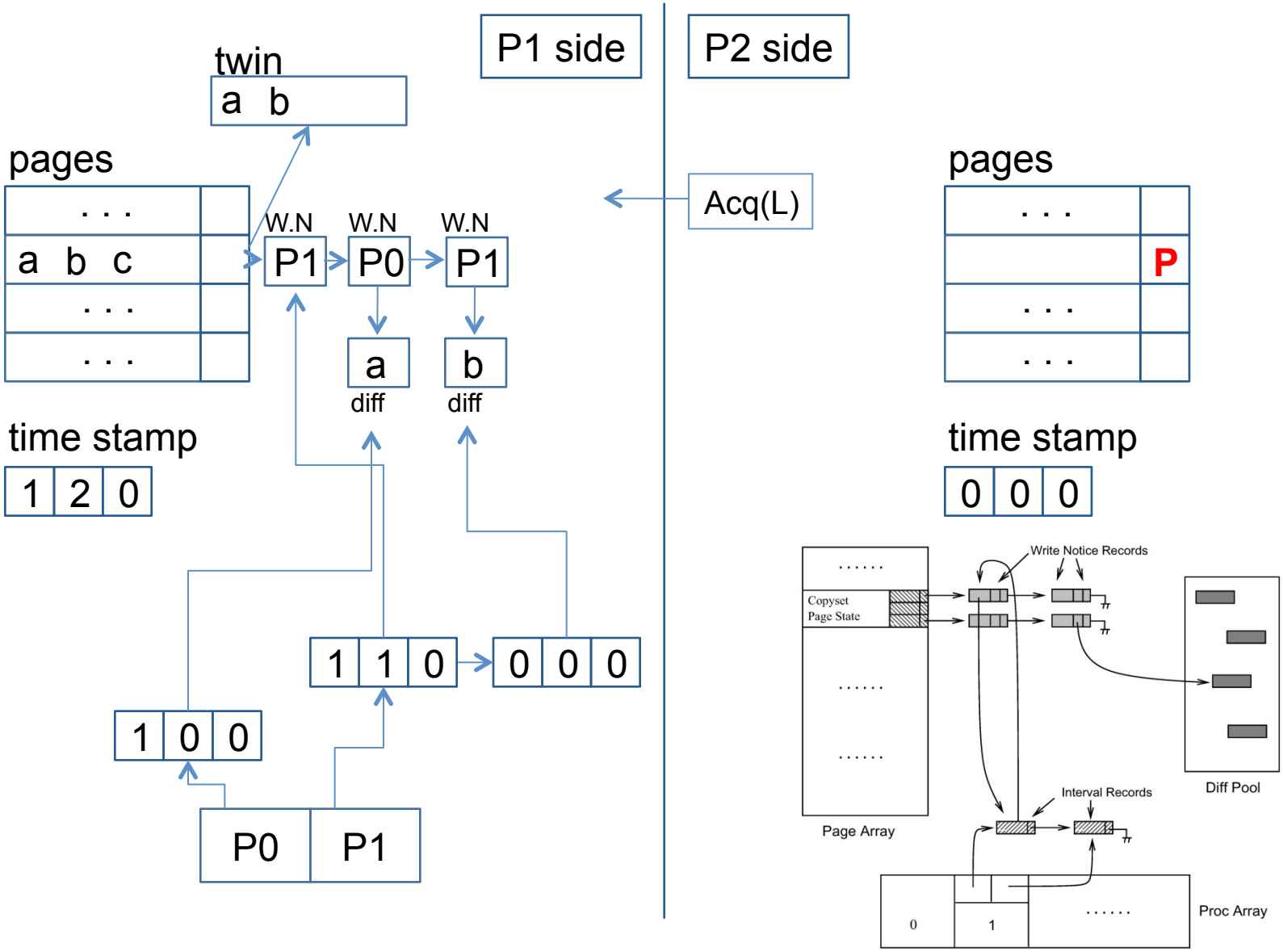
1	2	0



1	0	0

P0	P1

Implementation



Implementation

P1 side

P2 side

twin

a b

pages

. . .

a b c

. . .

. . .

W.N P1 → W.N P0 → W.N P1

a (diff)

b (diff)

time stamp

1 2 0

1 1 0 → 0 0 0

1 0 0

P0 P1

Acq(L)

pages

. . .

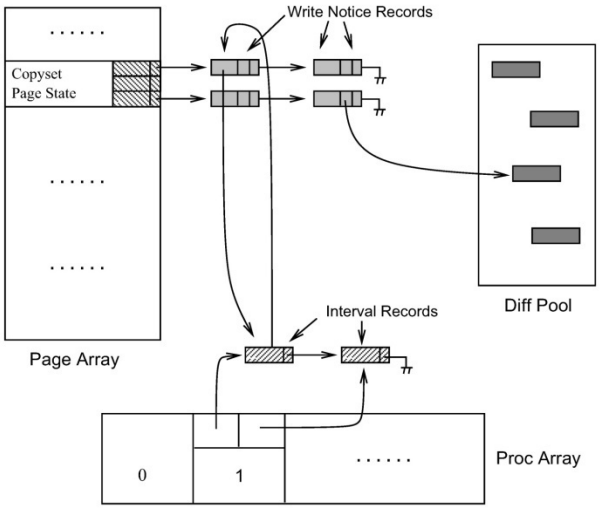
P → W.N P1 → W.N P0 → W.N P1

. . .

. . .

time stamp

1 1 1



Others

- Lock & barrier
  - Statically assigned manager
- Garbage collection
  - reclaim the space used by write notice records, interval records, and diffs
  - Triggered when the free space drops below a threshold

Evaluation

- Experimental Environment
  - 8 DECstation-5000/240
  - connected to a 100-Mbps ATM LAN and a 10-Mbps Ethernet

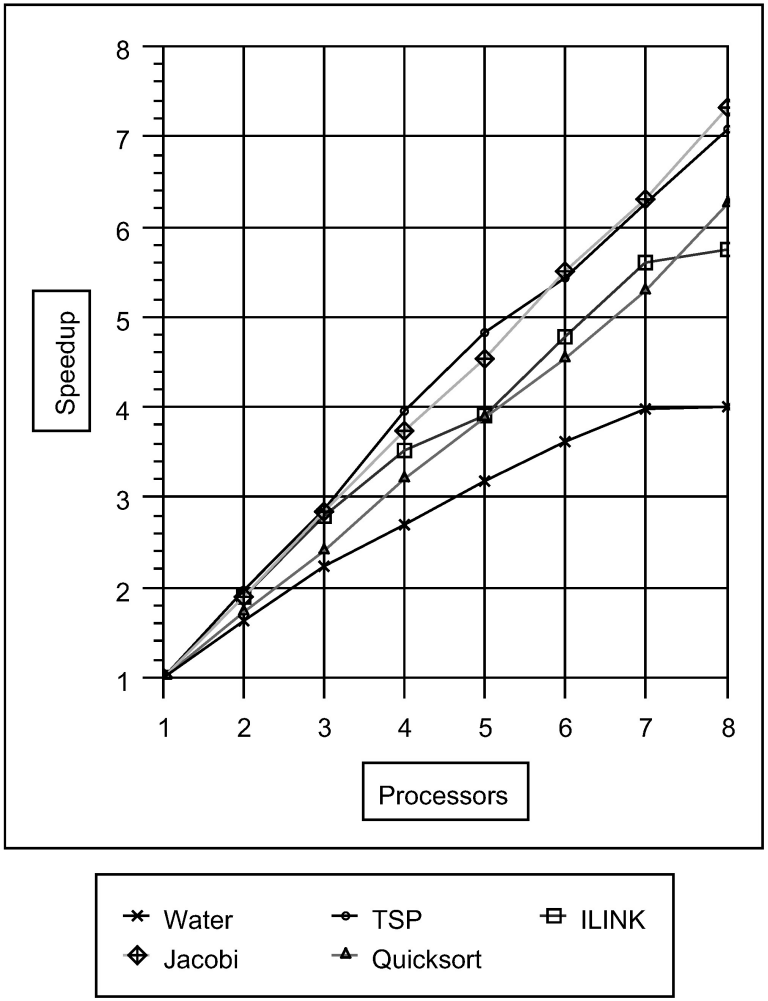
- Applications
  - Water – molecular dynamics simulation
  - Jacobi – Successive Over-Relaxation
  - TSP – branch & bound algorithm to solve the traveling salesman problem
  - Quicksort – using bubblesort to sort subarray of less than 1K element
  - ILINK – genetic linkage analysis

Evaluation

Execution statistics

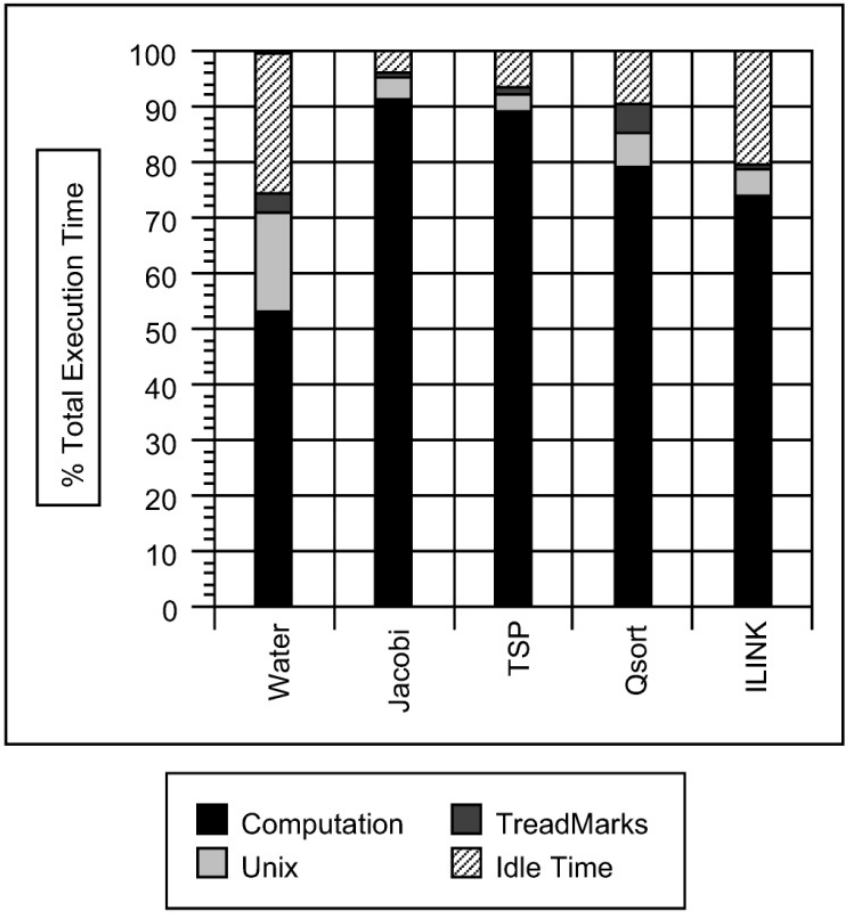
	Water	Jacobi	TSP	Quicksort	ILINK
Input	343 mols 5 steps	2000x1000 floats	19-city tour	256000 integers	CLP
Time (secs)	15.0	32.0	43.8	13.1	1113
Barriers/sec	2.5	6.3	0	0.4	0.4
Locks/sec	582.4	0	16.1	53.9	0
Msgs/sec	2238	334	404	703	456
Kbytes/sec	798	415	121	788	164

Speedup



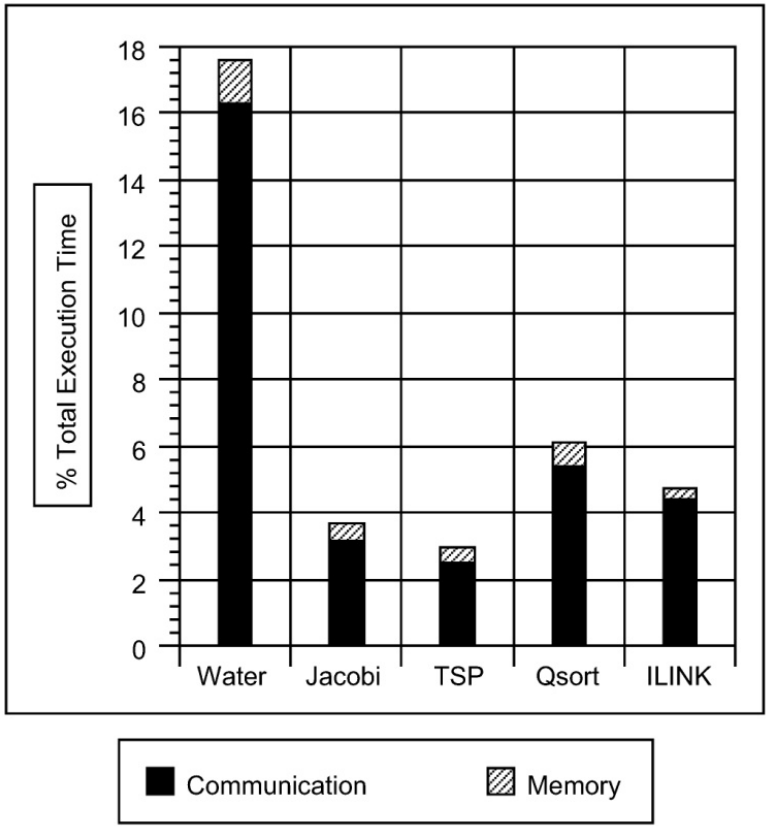
Evaluation

Execution time breakdown

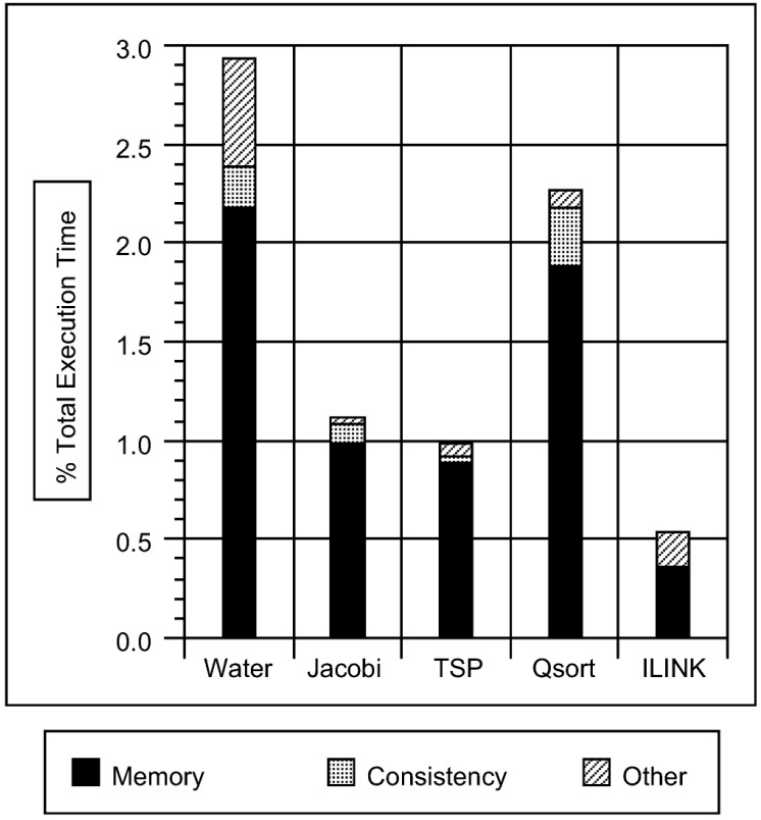


Evaluation

Unix overhead breakdown

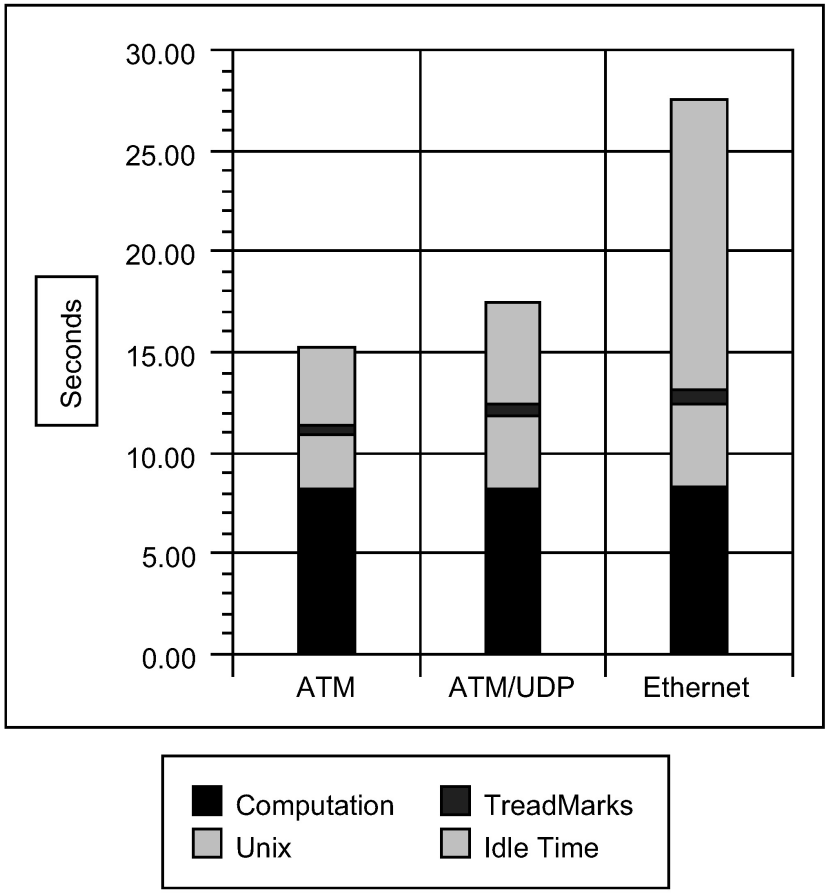


TreadMarks overhead breakdown



Evaluation

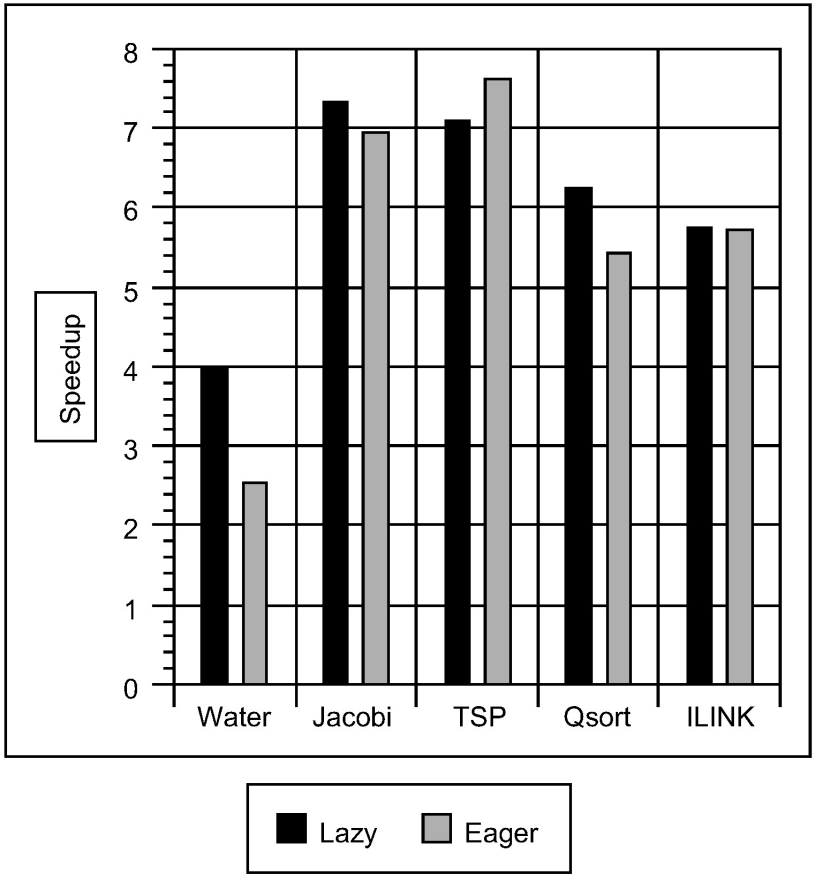
Execution time breakdown for Water



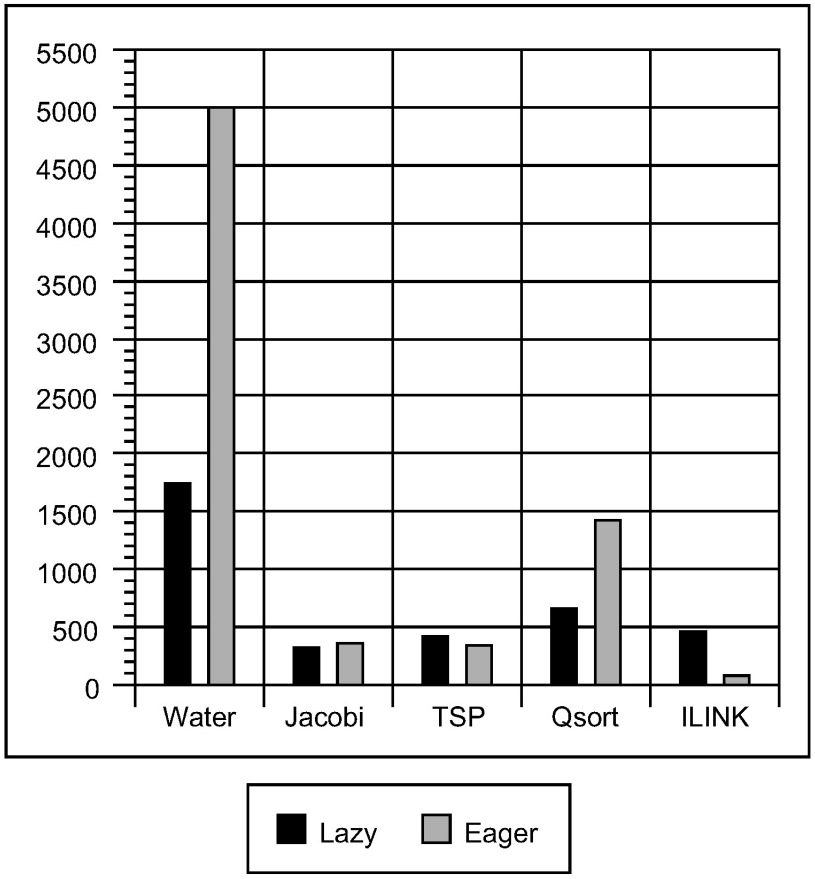
Evaluation

ERC vs. LRC

Speedup



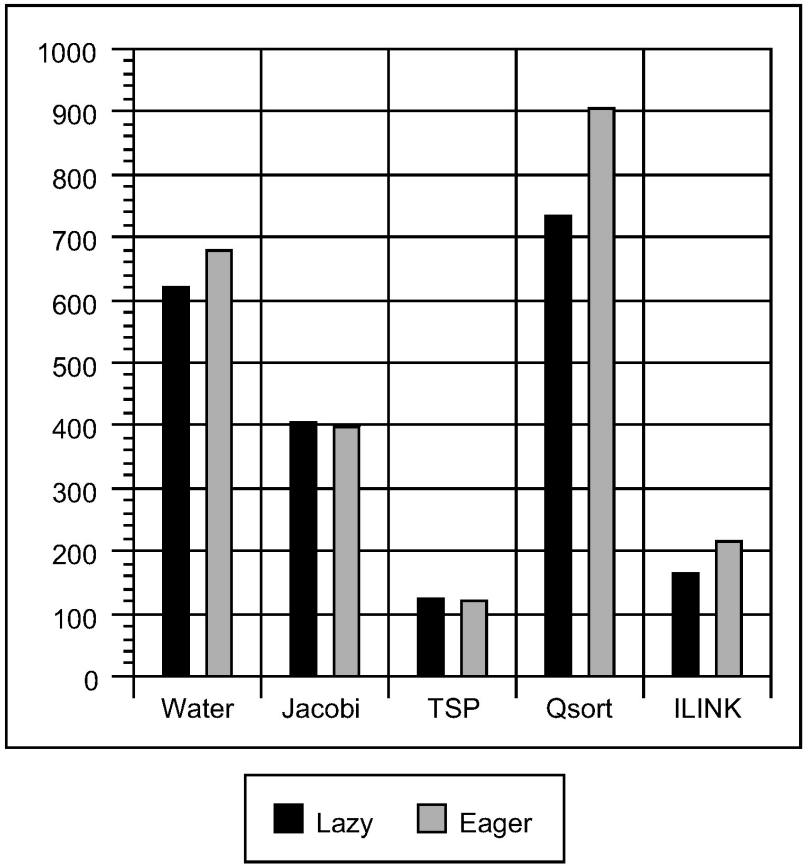
Message rate



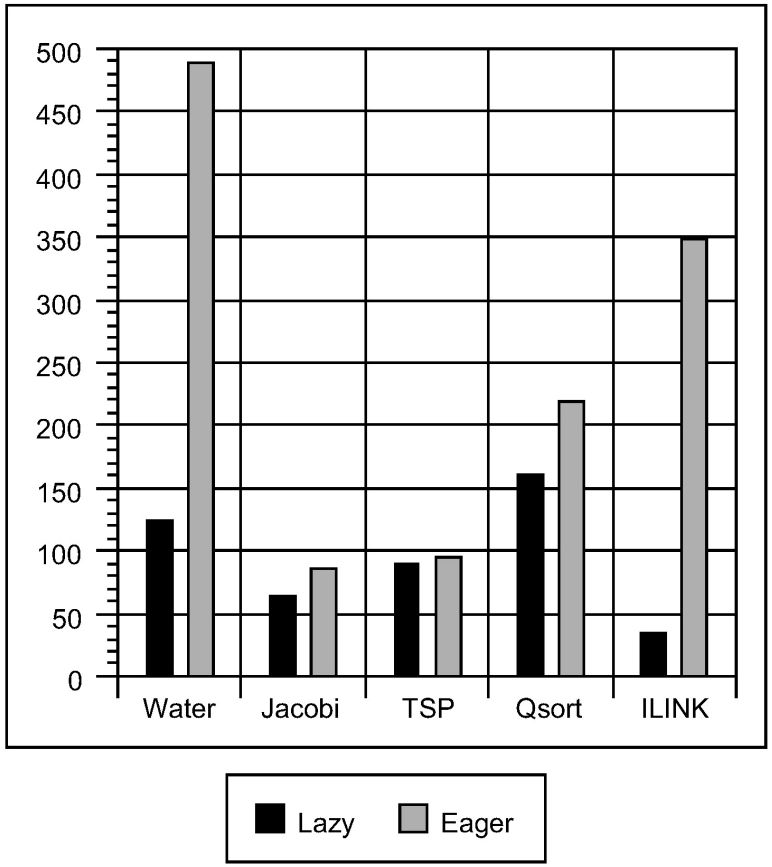
Evaluation

ERC vs. LRC

Data rate



Diff creation rate



Next Classes

- Will be given by Rong Chen
- Topics cover:
  - Eventual consistency
  - Fault tolerance
  - Graph processing
  - Key-value stores
  - …

Happy Mid-Autumn's Day

Thank you !

Any questions?