Today

Distributed Subsystems (Lesson 7)

✓ * Global memory system

⋯ * Distributed Shared Memory

⇒ * Distributed file Systems

Monday

Failures + recovery (Lesson 8)

* please watch LRVM lecture *

LRC With Multi-Writer Coherence Protocol

P1

Lock(L);

x ←
y ←
z ←
} X, Y, Z pages Modified in C.S ⇒ X_d, Y_d, Z_d } diffs

Unlock(L);

LRC With Multi-Writer Coherence Protocol

P1

Lock(L);

x ←
y ←
z ←
} X, Y, Z pages Modified in C.S ⇒ X_d, Y_d, Z_d } diffs

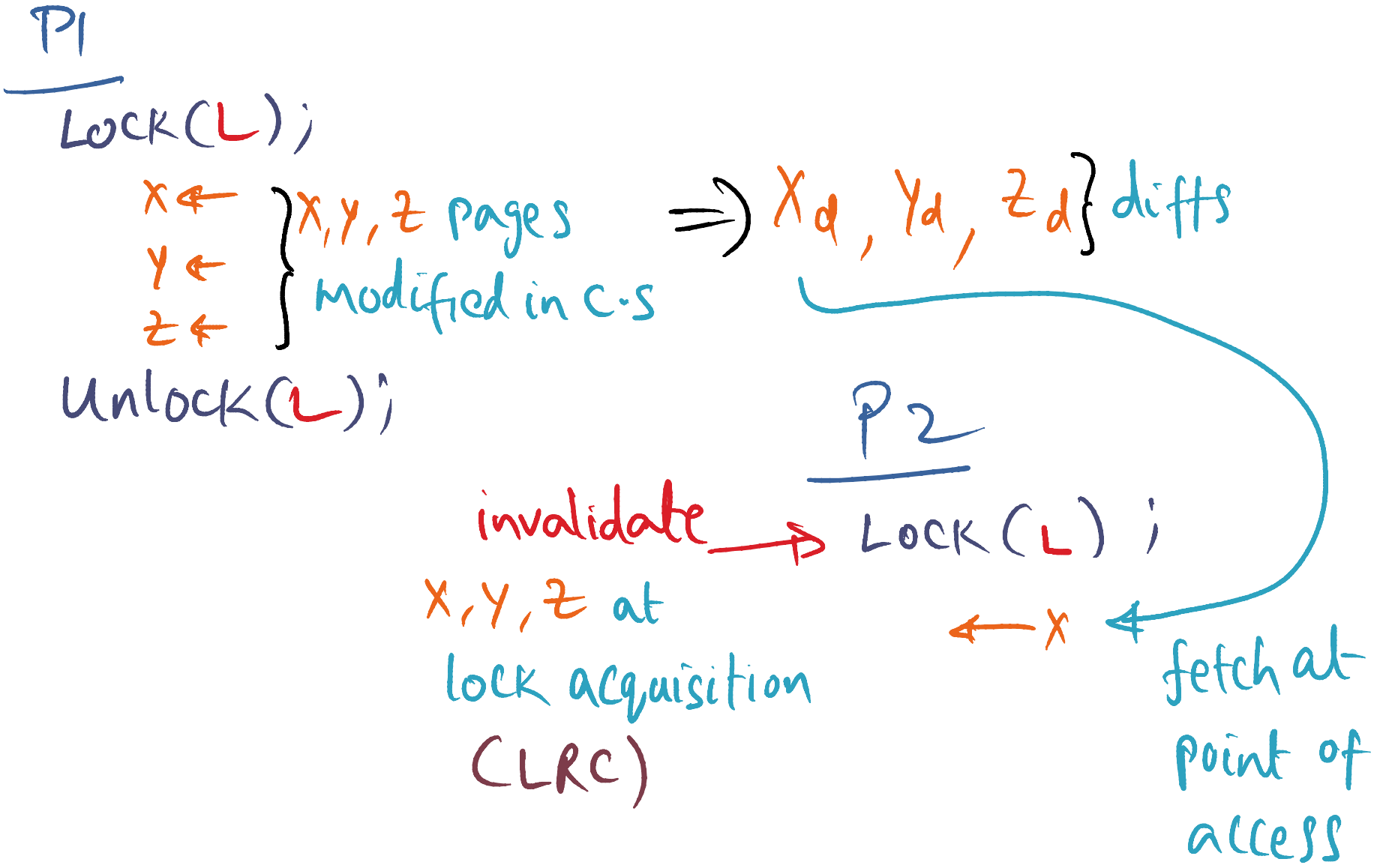
Unlock(L);

P2

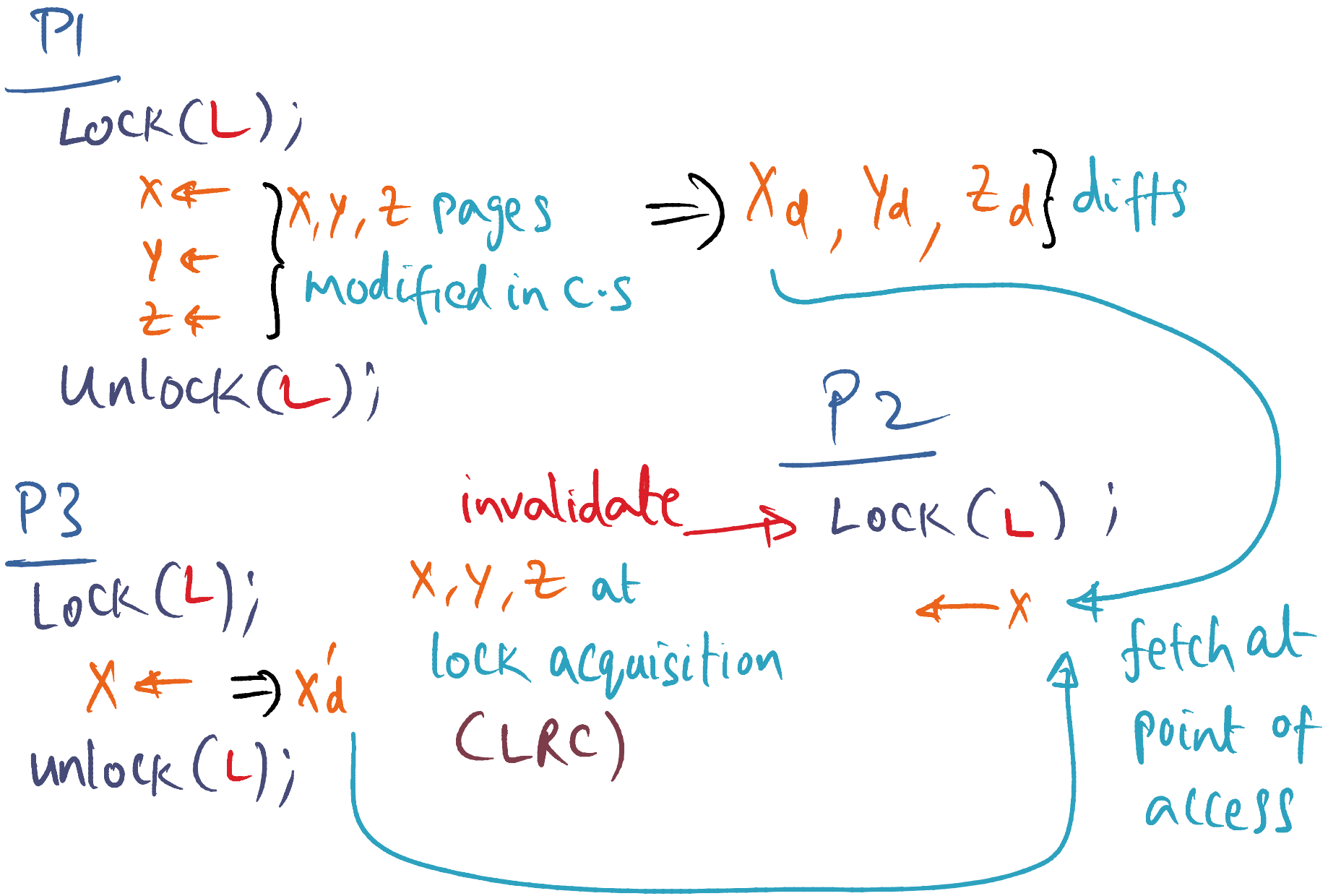
Lock(L);

invalidate → X, Y, Z at lock acquisition (LRC)

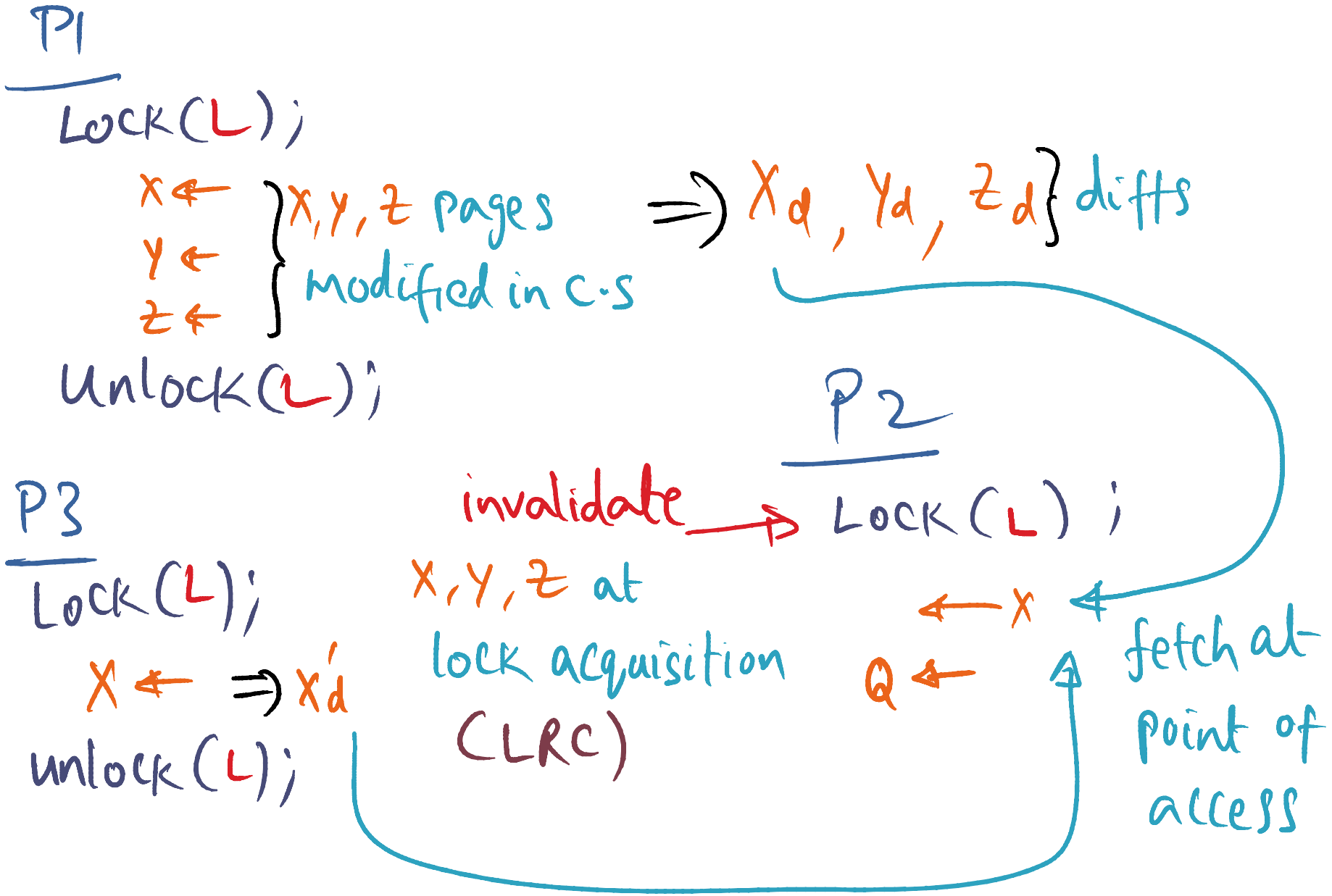
LRC With Multi-Writer Coherence Protocol

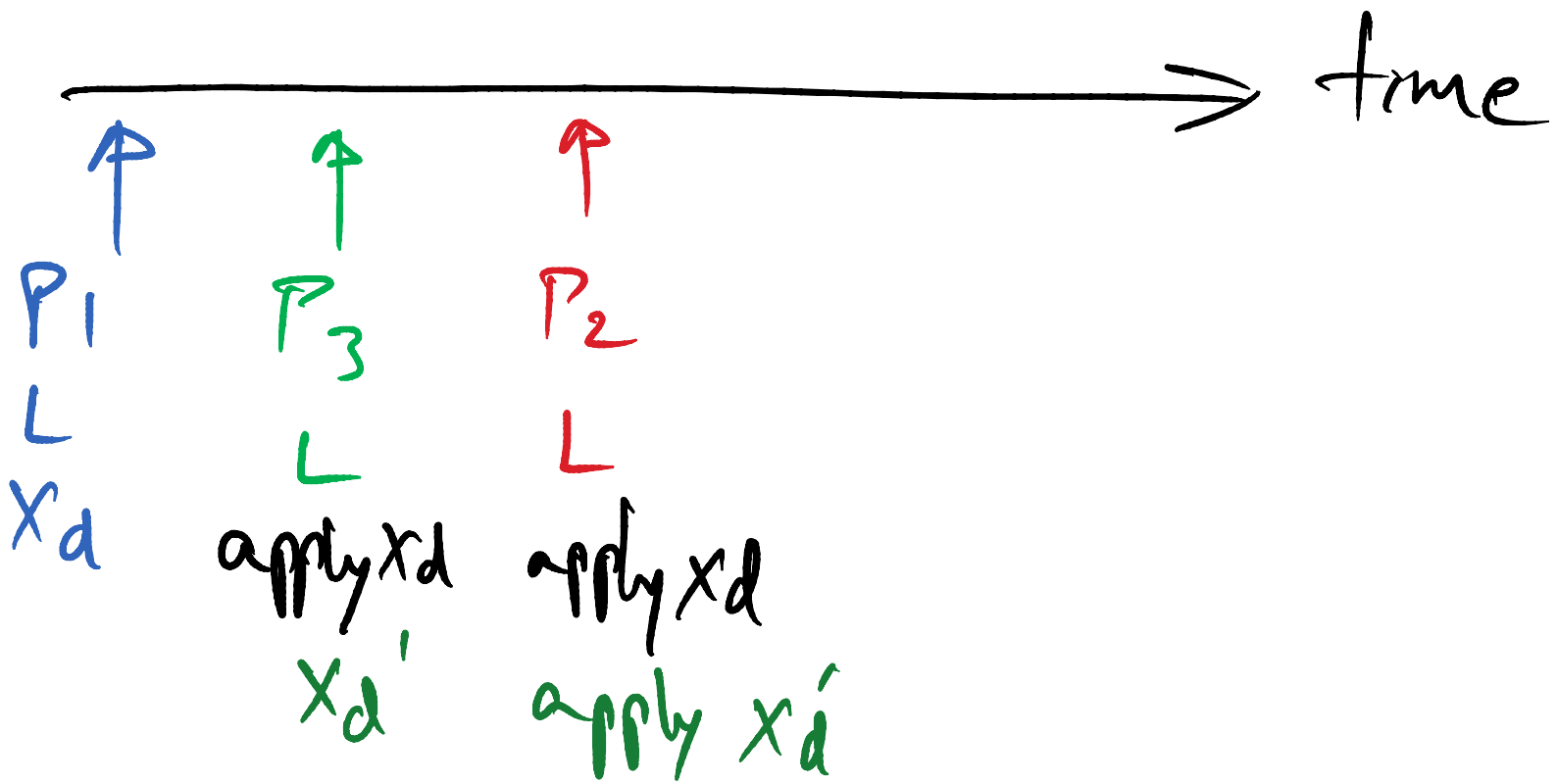


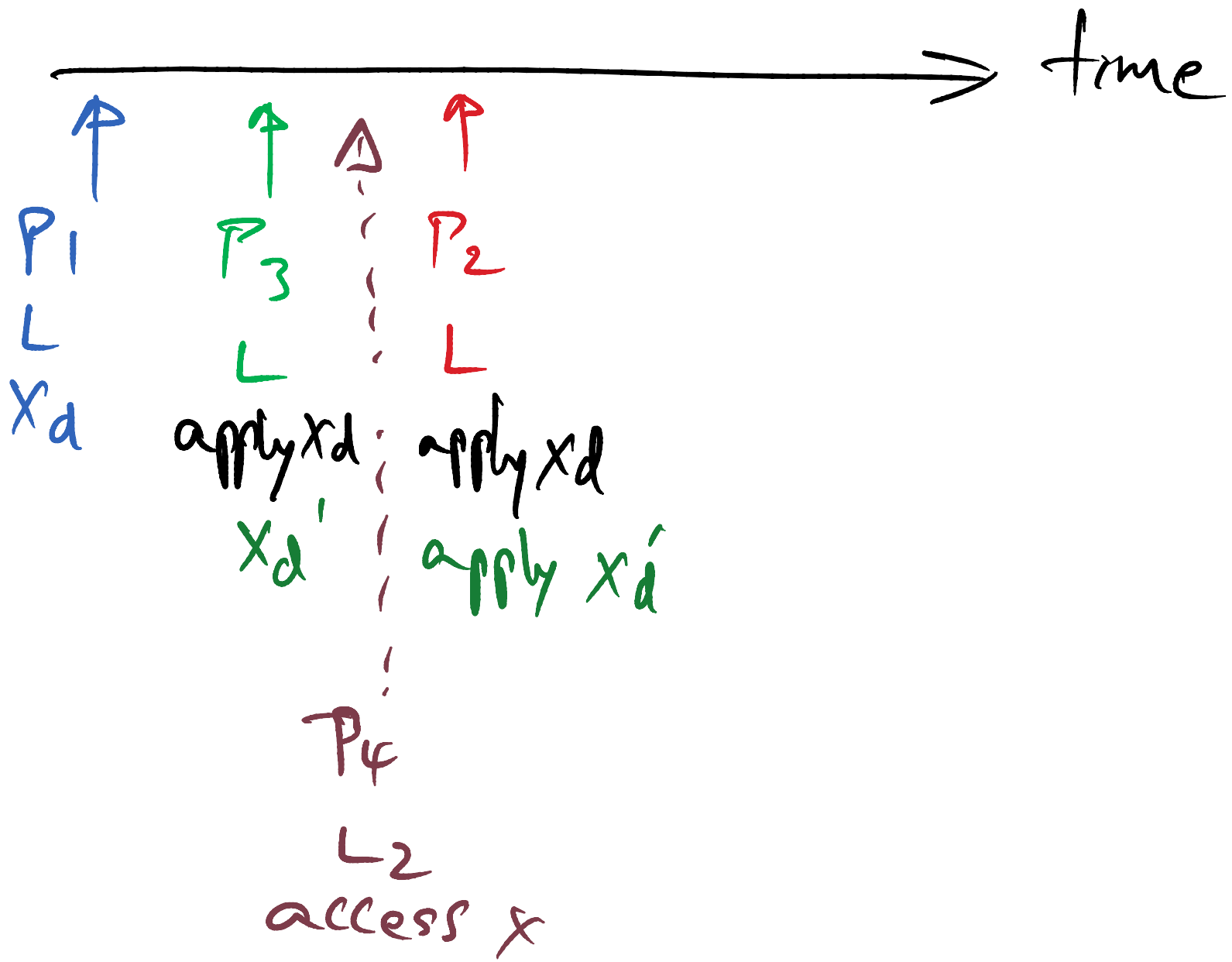
LRC With Multi-Writer Coherence Protocol

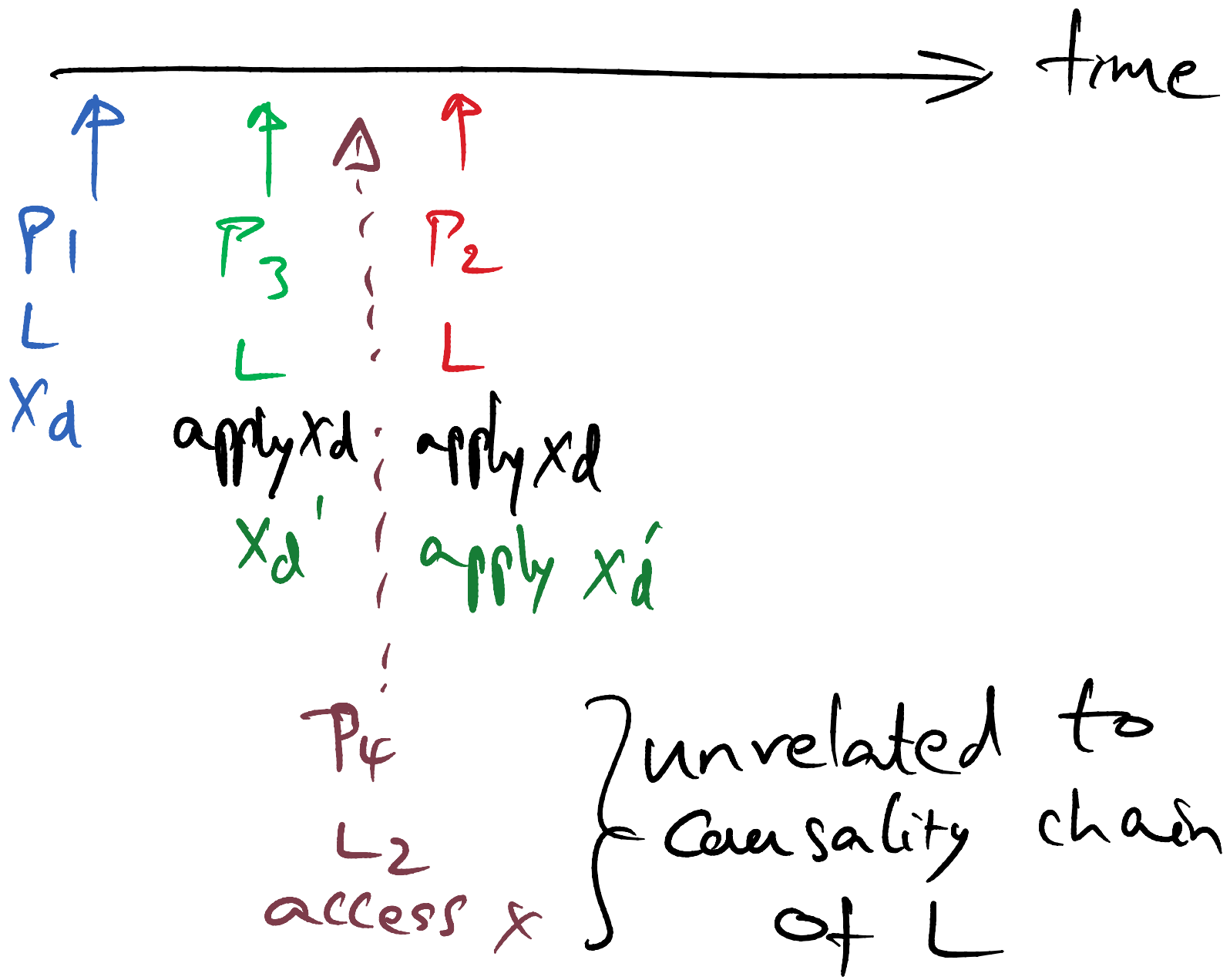


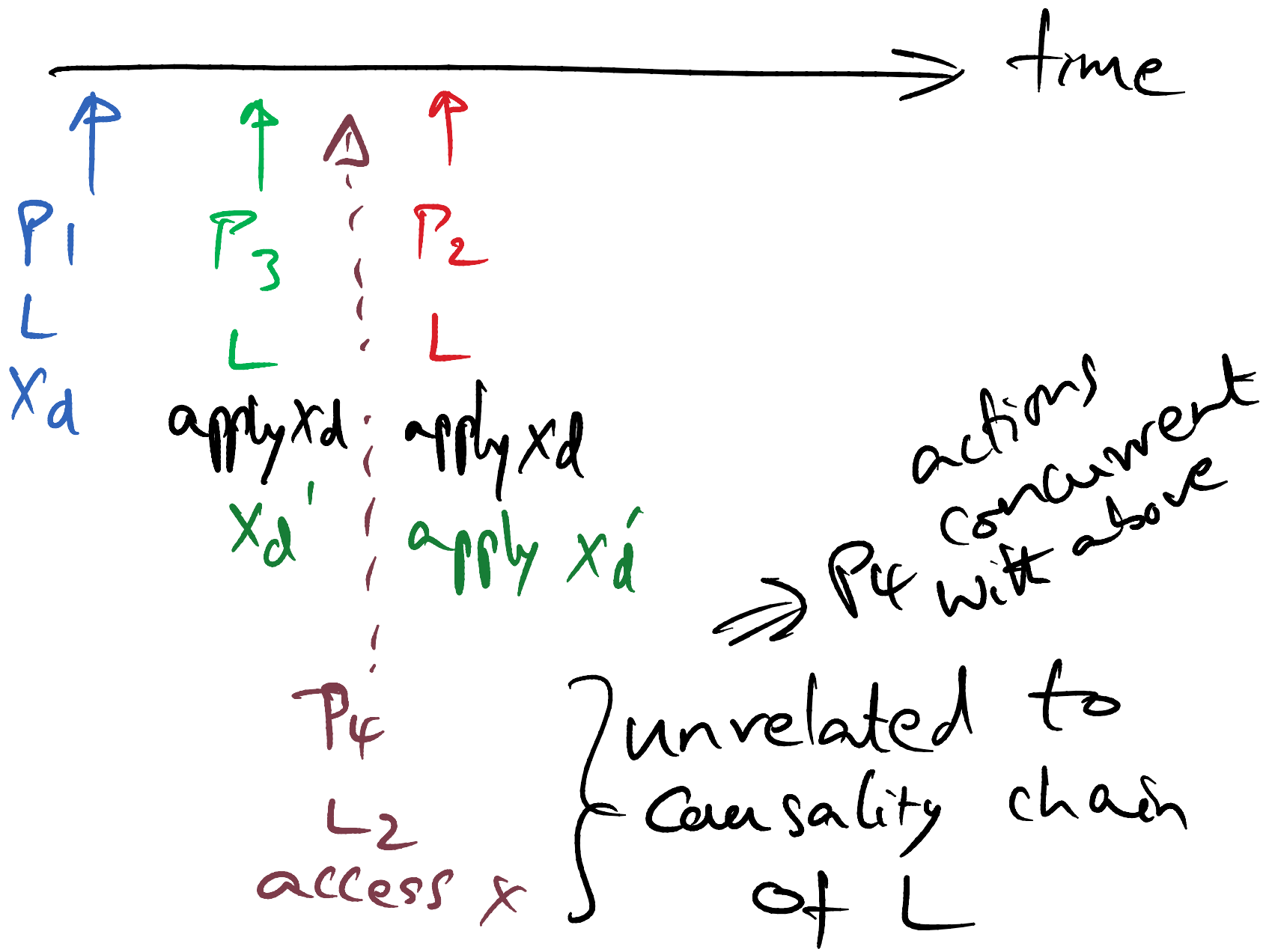
LRC With Multi-Writer Coherence Protocol

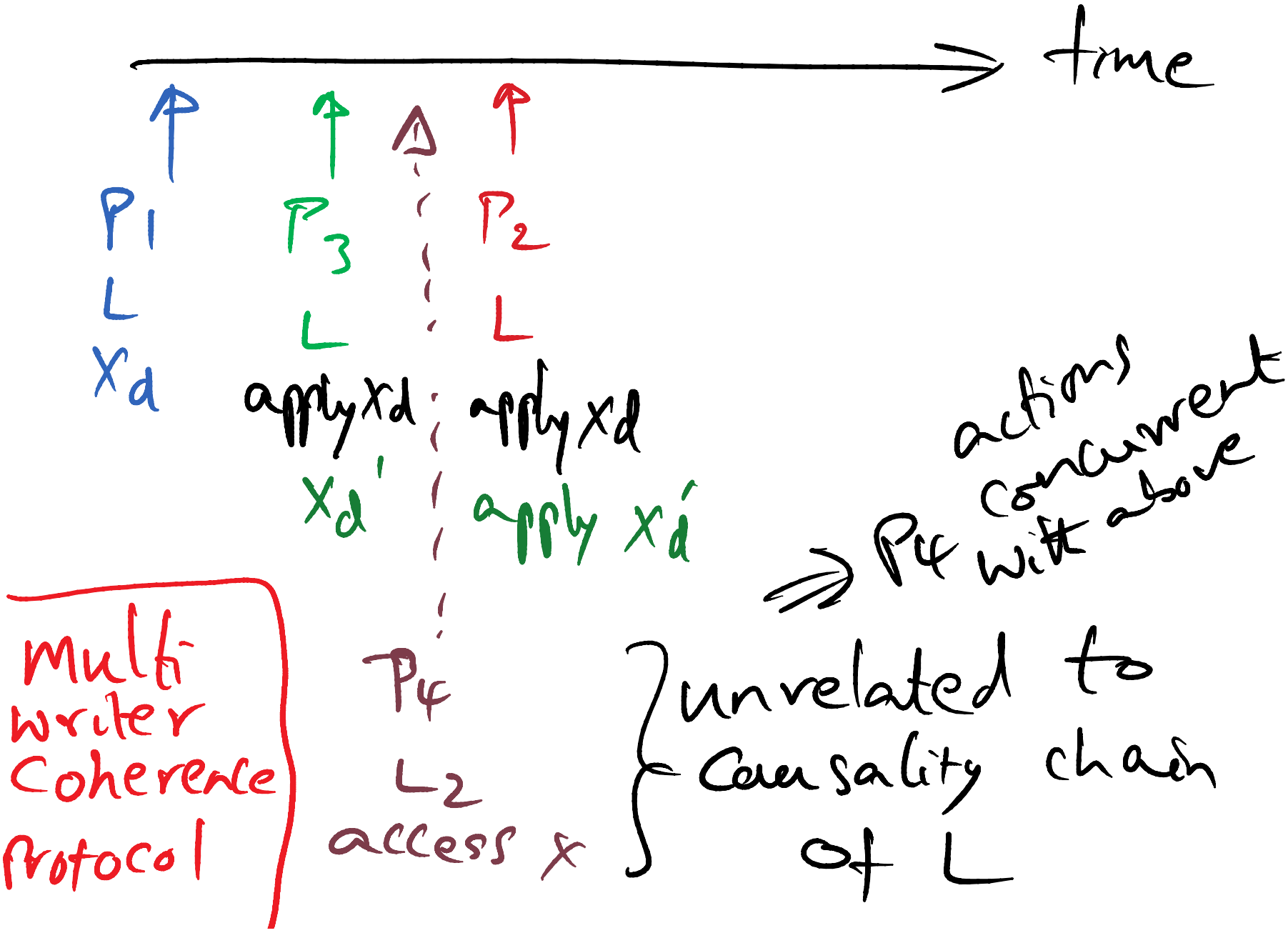



time
P1
L
Xd
P3
L
apply xd
Xd'
P2
L
apply xd
apply xd'

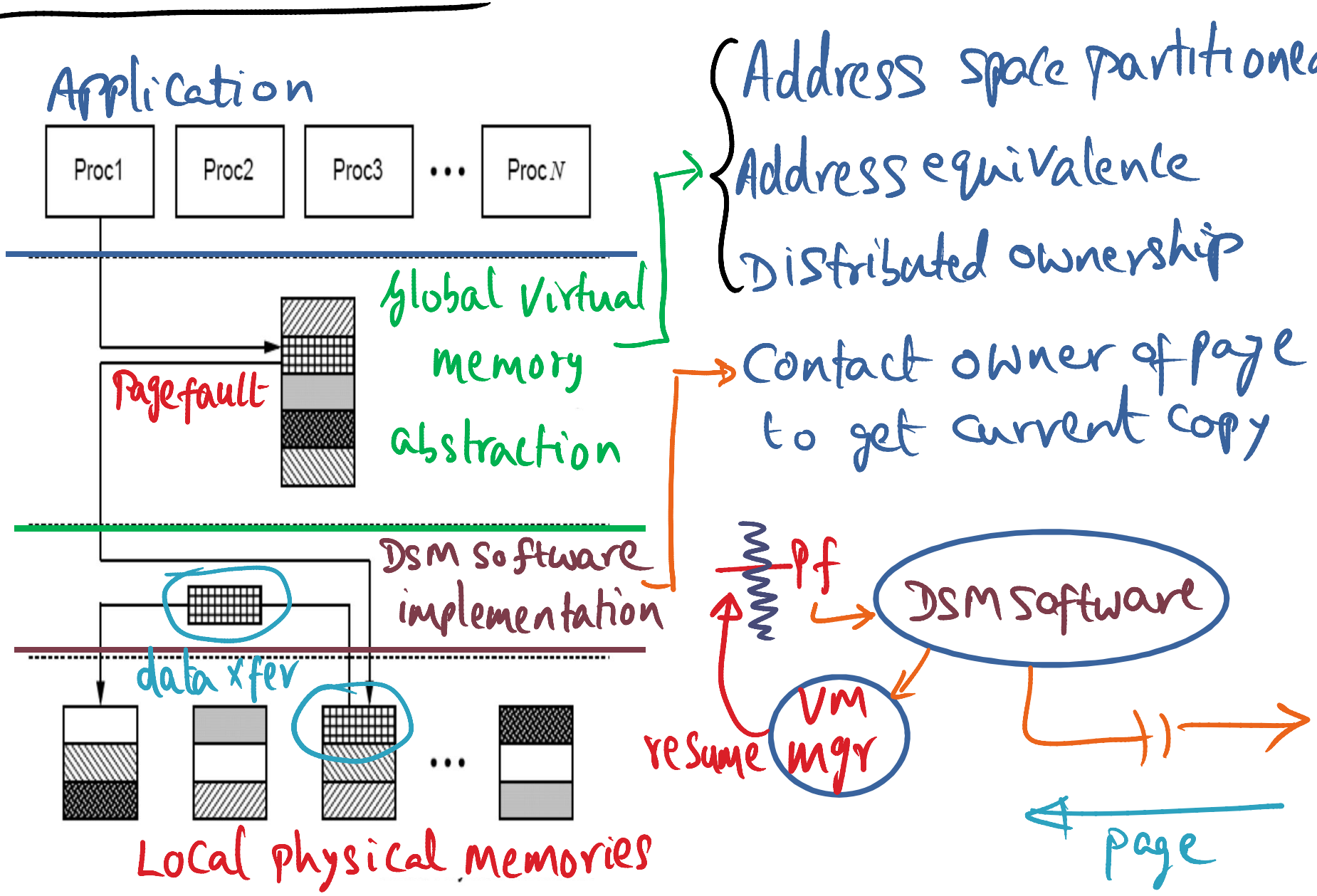

time
P1
L
Xd
P3
L
apply xd
Xd'
P2
L
apply xd
apply x'd
P4
L2
access x


time
P1
L
Xd
P3
L
apply xd
Xd'
P2
L
apply xd
apply xd'
P4
L2
access x
unrelated to
causality chain
of L


time
P1
L
Xd
P3
L
P2
L
apply xd
apply xd
Xd'
apply xd'
P4
L2
access x
actions concurrent with above
unrelated to causality chain of L

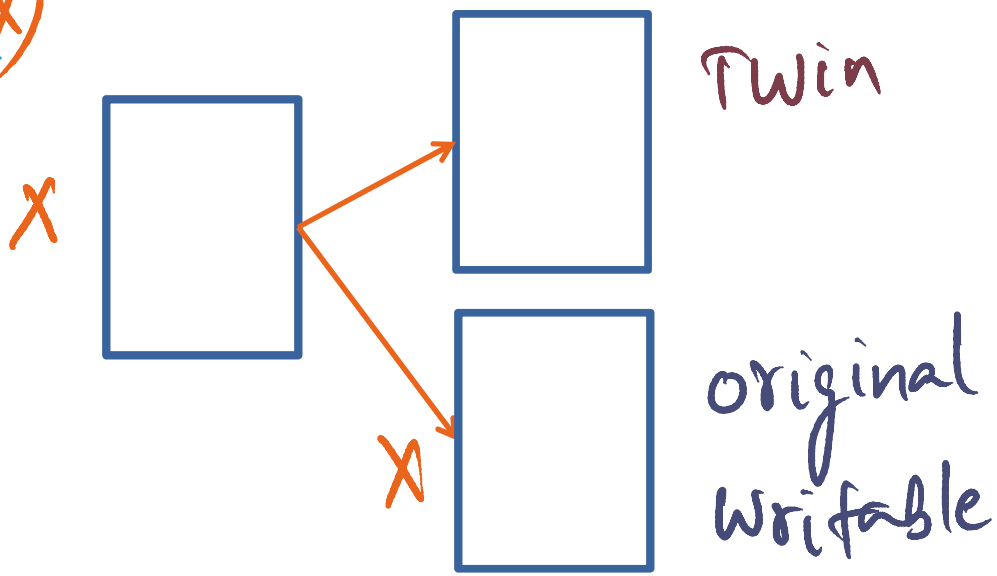

time
P1
P3
P2
L
L
L
Xd
apply xd
apply xd
Xd'
apply xd'
Pk
L2
access x
actions concurrent with above
unrelated to causality chain of L
Multi writer Coherence Protocol

Software DSM



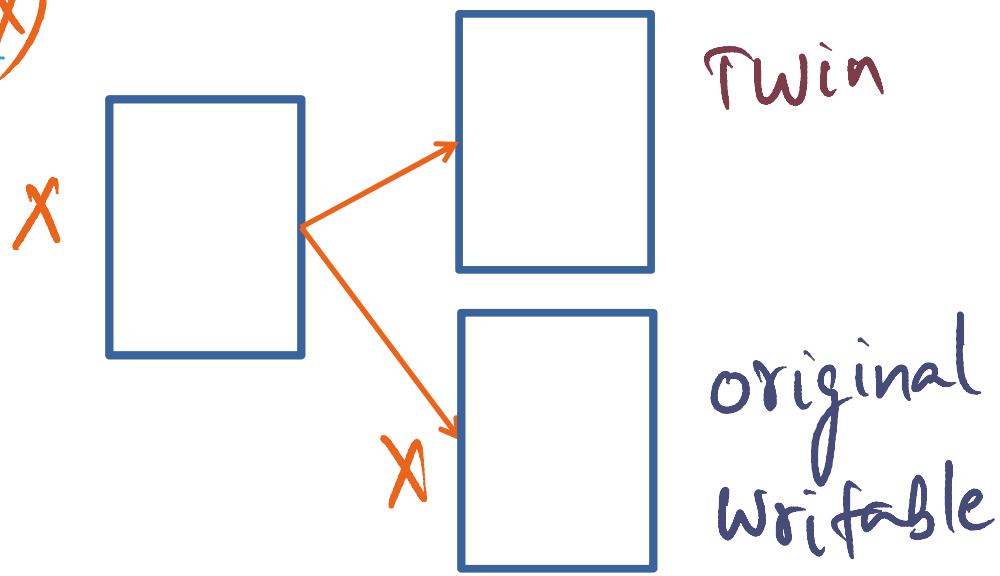
Implementation

write(x)

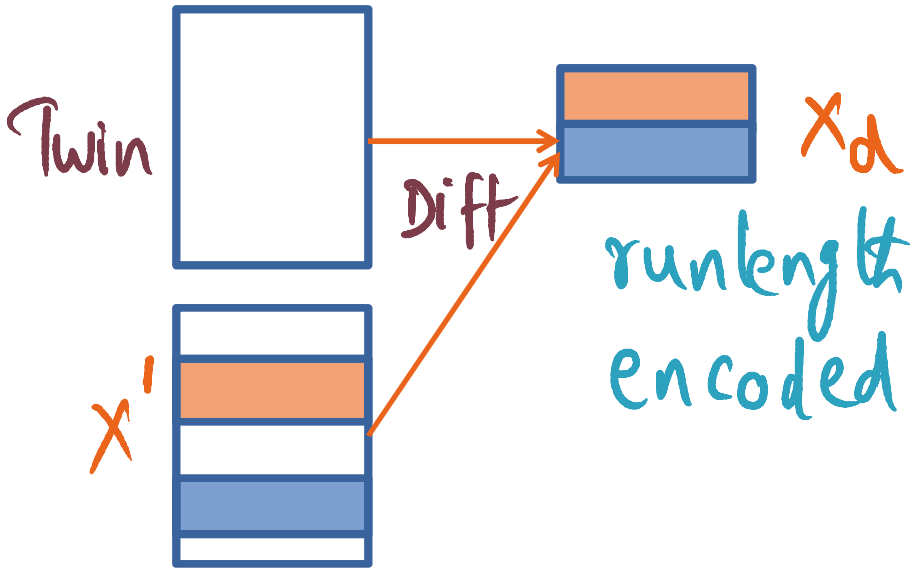


Implementation

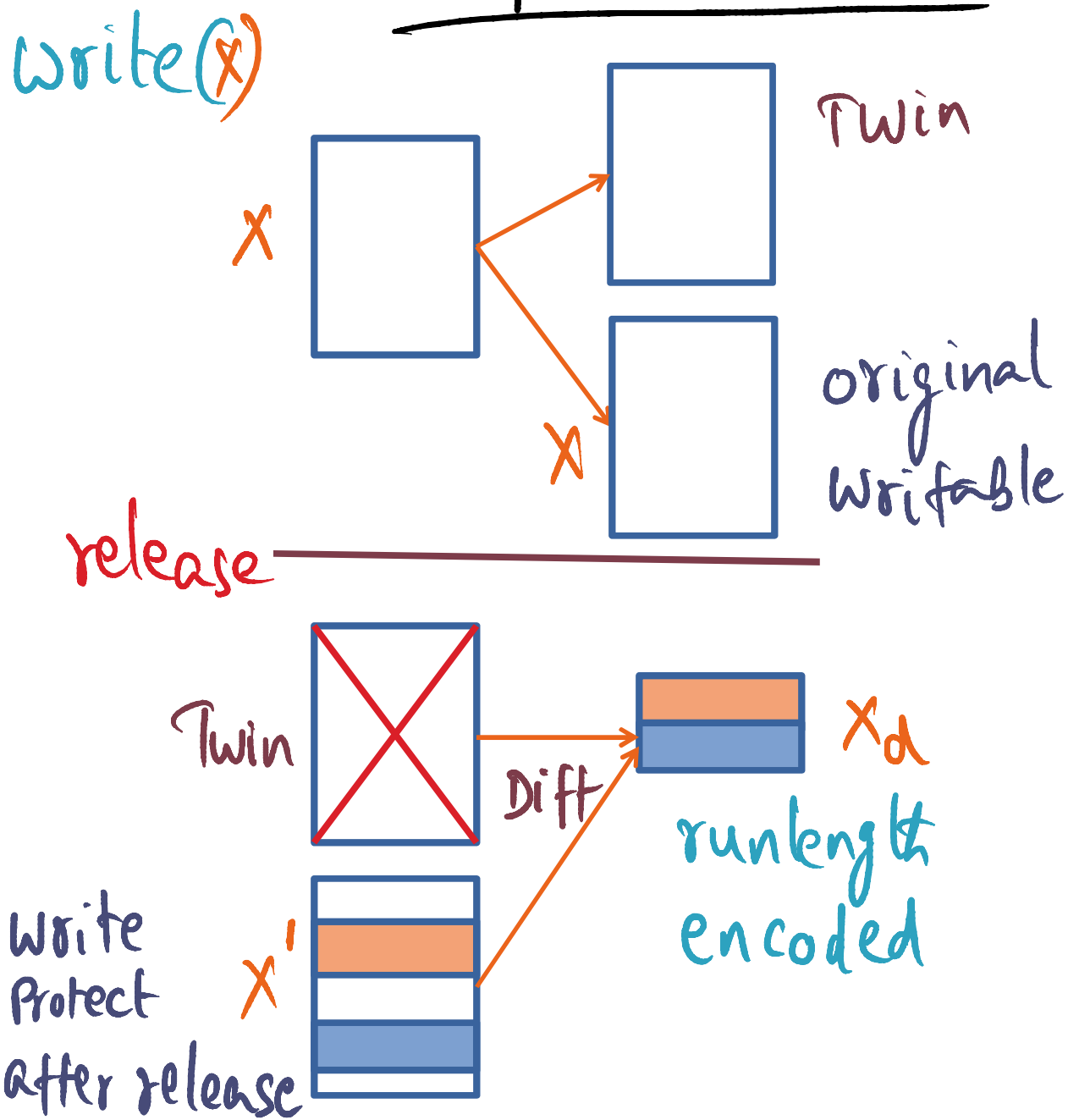
write(x)



release

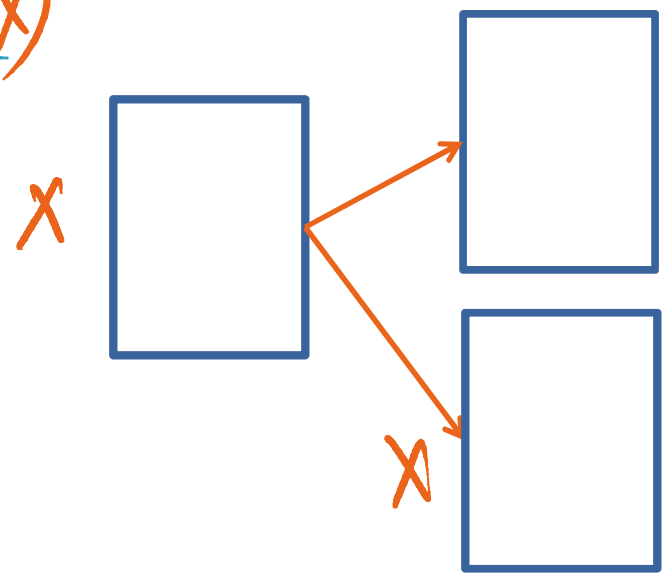


Implementation



Implementation

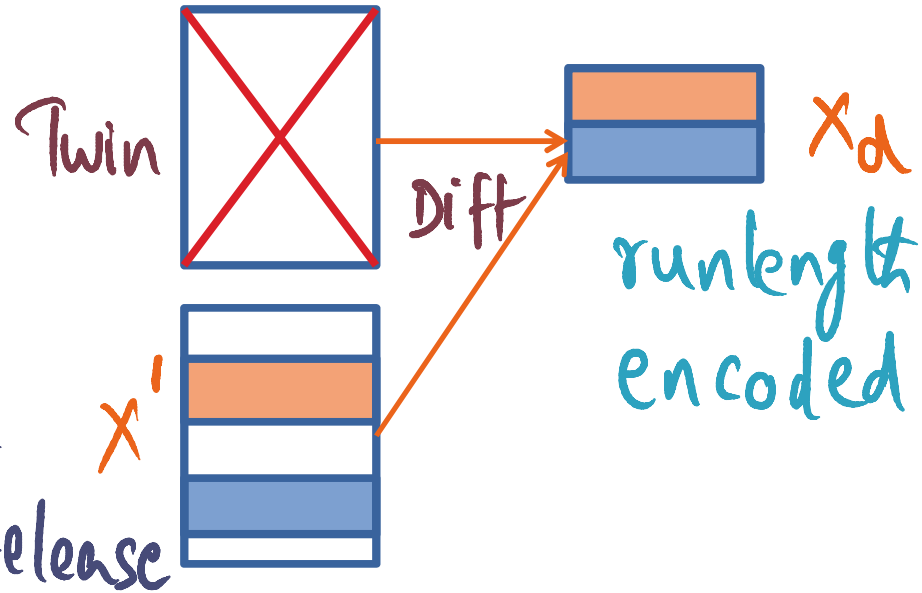
write(x)



Twin

original Writable

release



runlength encoded

Writes to same Portion of a Page under different locks

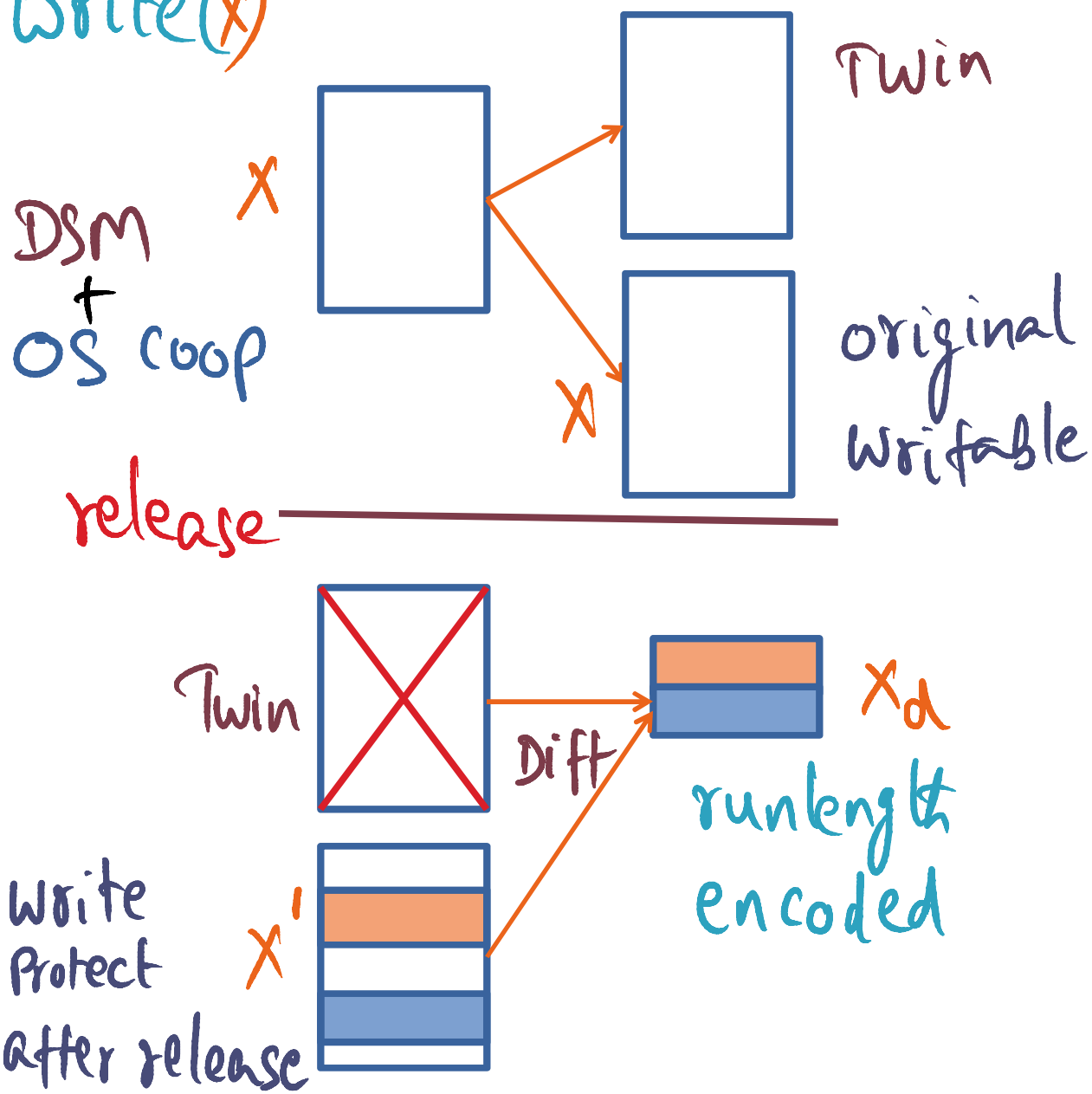
⇒ Data race

⇓

Application Problem

Implementation

write(x)



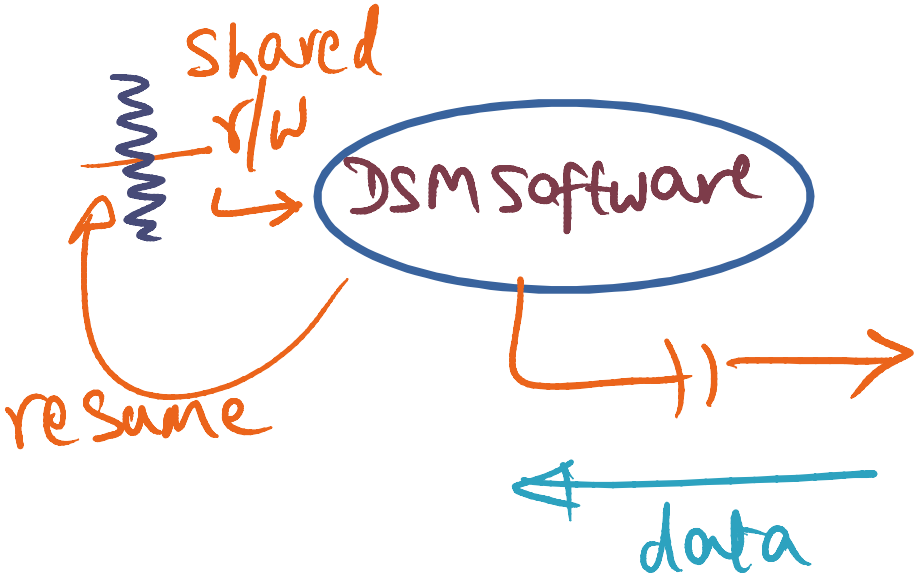
Writes to same Portion of a Page under different locks

⇒ Data race

⇓

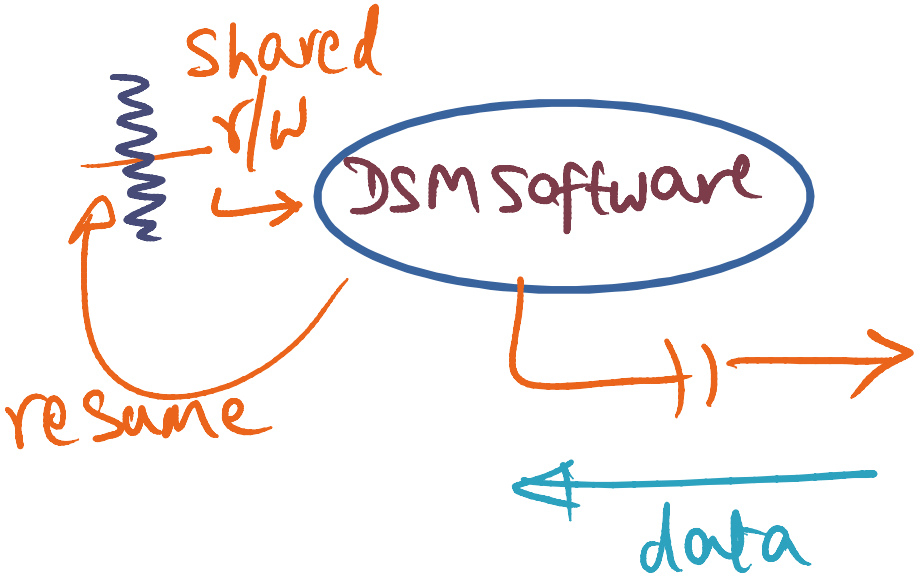
Application Problem

Non-page-based DSM

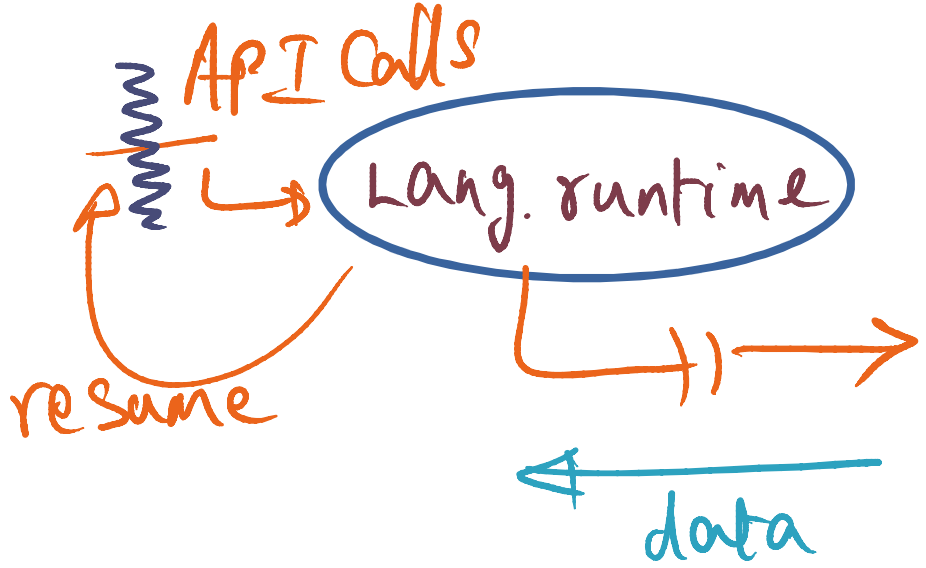


Library-based
- annotate shared variables
- Coherence actions inserted at point of access

Non-page-based DSM



Library-based
- annotate shared variables
- Coherence actions inserted at point of access



Structured DSM
- API for structs
- Coherence actions on API calls

Scalability

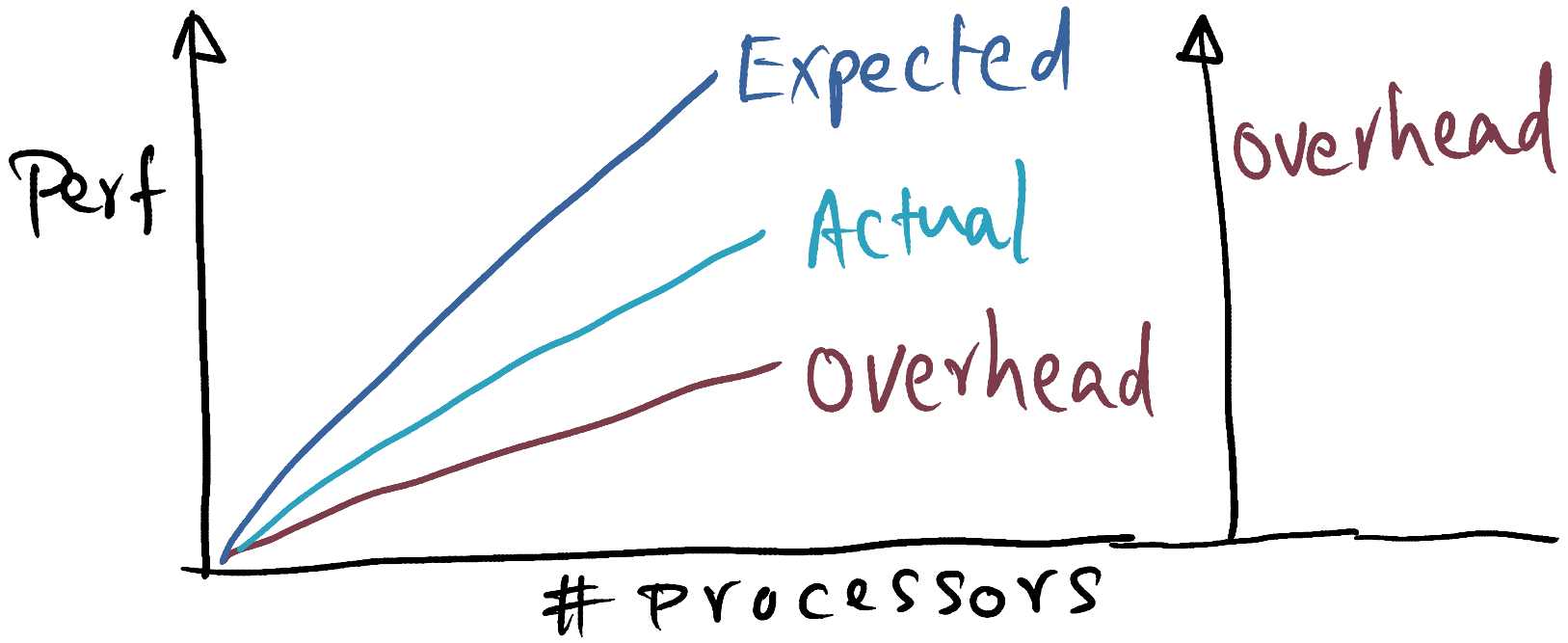
Expectation with more processors?

Pro:

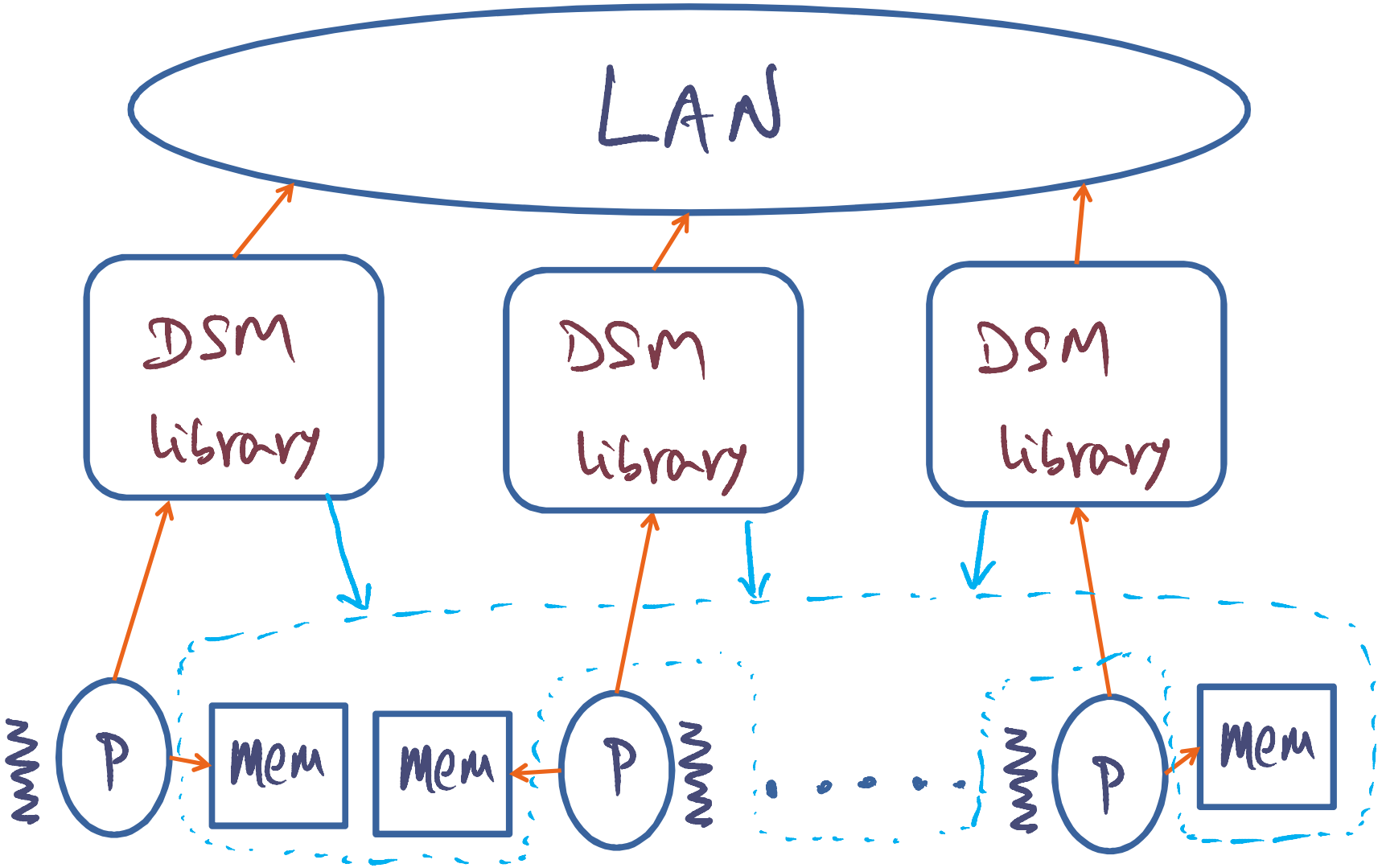
Exploit parallelism

Con:

Increased overhead



DSM and Speedup



- Speedup is not automatic
  - If sharing is too fine grain no hope of speedup on DSM systems
  - Illusion of shared memory via software, not really physically shared memory => you have to be very careful how much memory you actually share between the threads that are running on different nodes of the cluster

Recall Thacker's quote regarding sharing memory!

- Basic principle
  - Comp:comm ratio has to be high for any hope of speedup
  - What does this mean for shared memory codes?
    - Dynamic data structures can lead to a lot of "implicit" communication across the LAN

Some classes of scientific applications with coarse-grained sharing may be able to achieve speedup with increasing number of processors if carefully constructed.

Key takeaways

- DSM as originally envisioned, i.e., as a "threads package" for a cluster is dead.
- Structured DSM, namely, providing higher level data abstractions for sharing among threads executing on different nodes of the cluster is attractive to reduce the programming pain for the developers of distributed applications on a cluster.

We will discuss one such system called "Persistent Temporal Streams" in a later lesson.