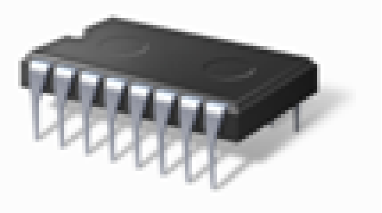
Better I/O Through Byte-Addressable, Persistent Memory

Jeremy Condit, Ed Nightingale, Chris Frost,
Engin Ipek, Ben Lee, Doug Burger, Derrick Coetzee

Microsoft®
Research

A New World of Storage

DRAM



+ Fast

+ Byte-addressable

- Volatile

Disk / Flash

+ Non-volatile

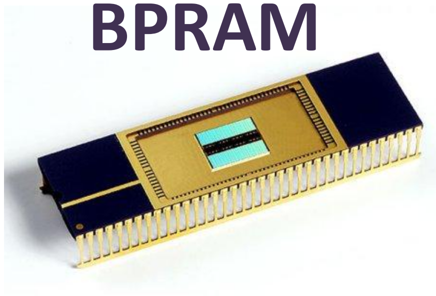
- Slow

- Block-addressable

A New World of Storage

Byte-addressable, Persistent RAM

BPRAM

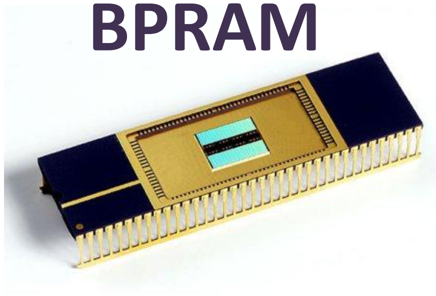


+ Fast
+ Byte-addressable
+ Non-volatile

A New World of Storage

Byte-addressable, Persistent RAM

BPRAM

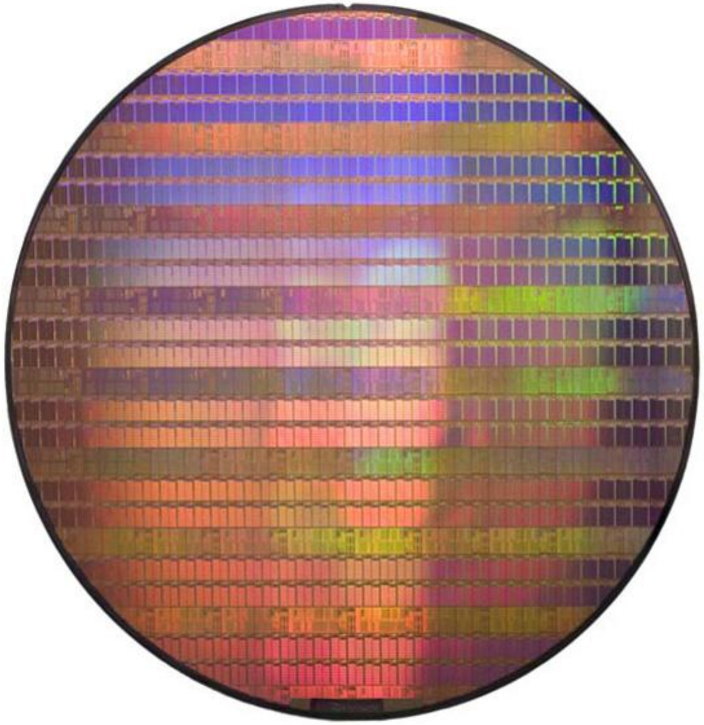


+ Fast
+ Byte-addressable
+ Non-volatile

How do we build fast, reliable systems with BPRAM?

Phase Change Memory

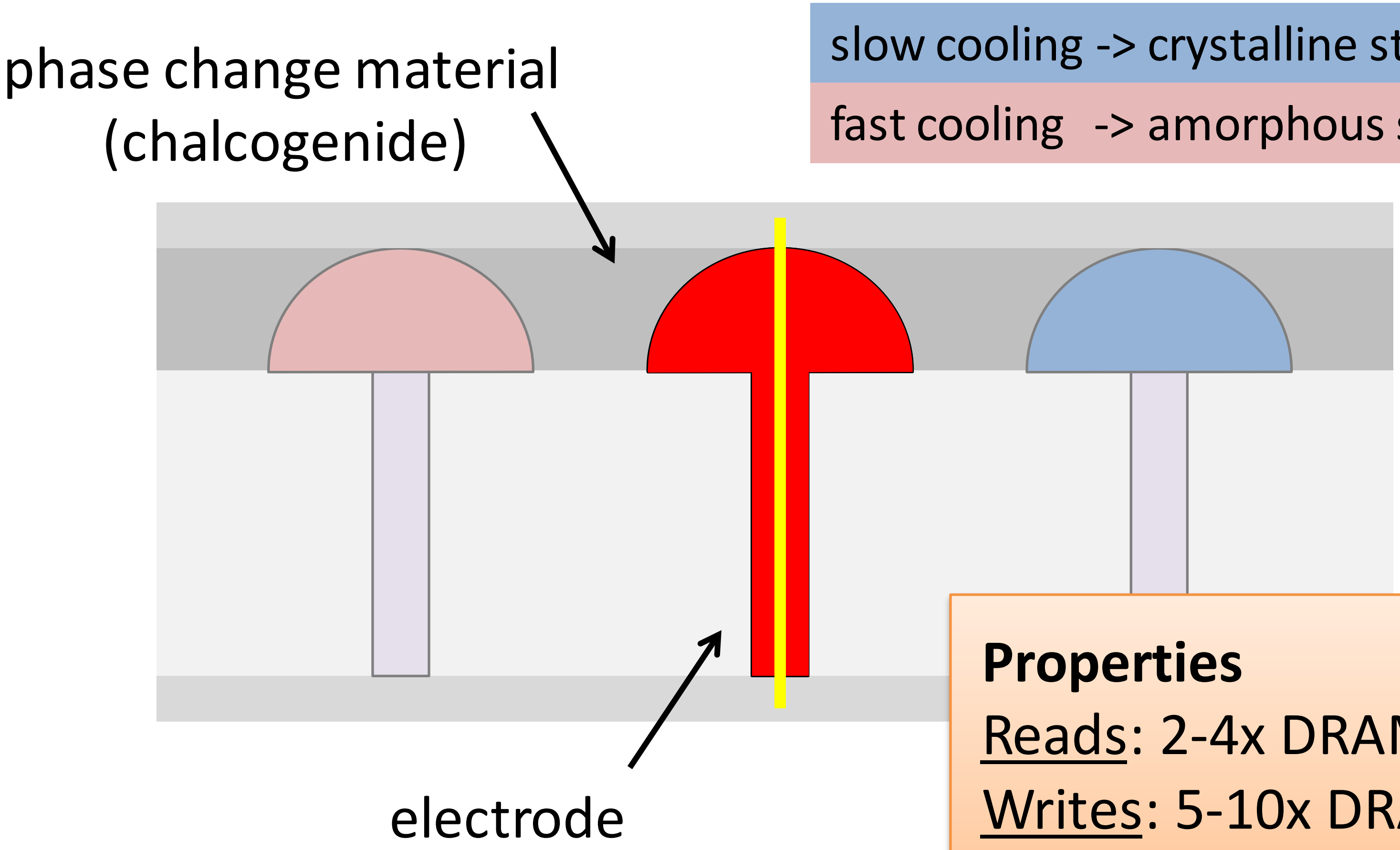
- Most promising form of BPRAM
- “Melting memory chips in mass production” – *Nature*, 9/25/09



Phase Change Memory

slow cooling -> crystalline state (1)

fast cooling -> amorphous state (0)



Properties

Reads: 2-4x DRAM

Writes: 5-10x DRAM

Endurance: 10^8+

A New World of Storage

Byte-addressable, Persistent RAM

BPRAM

+ Fast
+ Byte-addressable
+ Non-volatile

How do we build fast, reliable systems with BPRAM?

This talk: **BPFS**, a file system for BPRAM

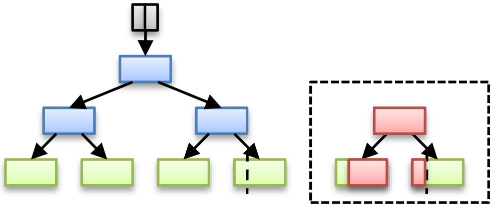
Result: Improved **performance** and **reliability**

Goal

New guarantees for applications

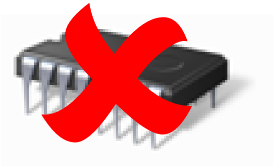
- File system operations will commit **atomically** and **in program order**
- Your data is **durable** as soon as the **cache is flushed**

New mechanism:
short-circuit shadow paging

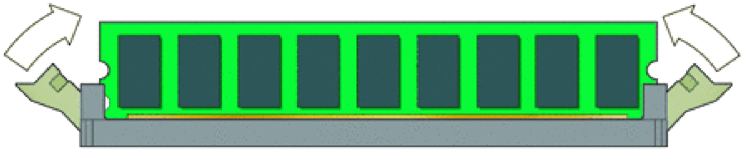


Design Principles

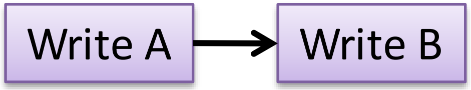
1. Eliminate the **DRAM buffer cache**; use the **L1/L2 cache** instead



2. Put BPRAM on the **memory bus**



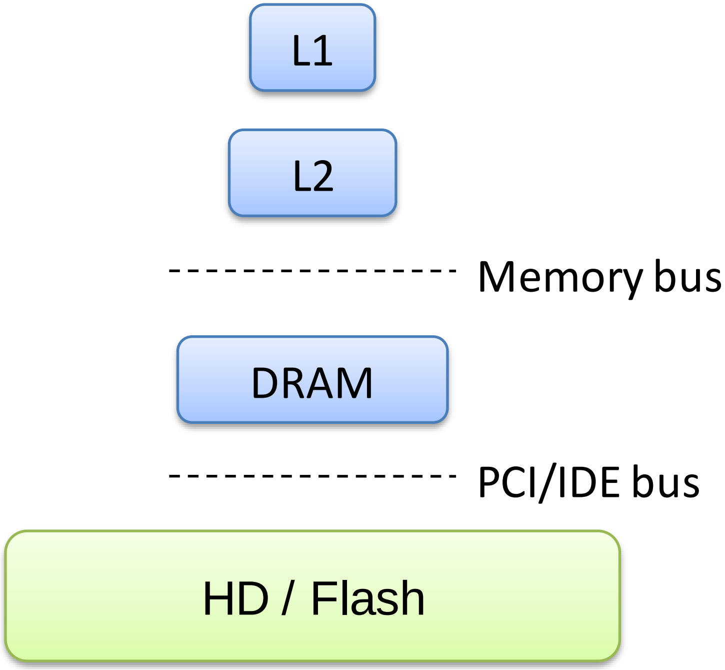
3. Provide **atomicity** and **ordering** in hardware



Outline

- Intro
- **File System**
- Hardware Support
- Evaluation
- Conclusion

BPRAM in the PC



L1

L2

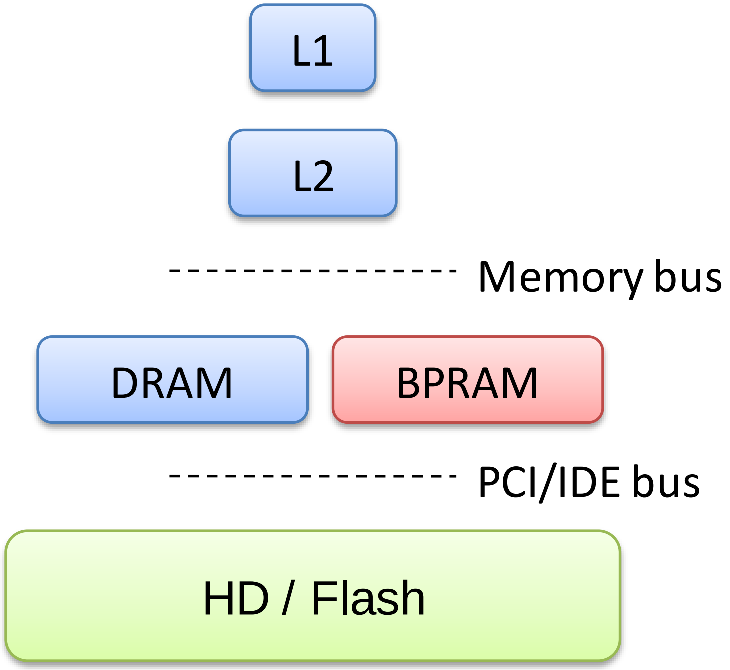
Memory bus

DRAM

PCI/IDE bus

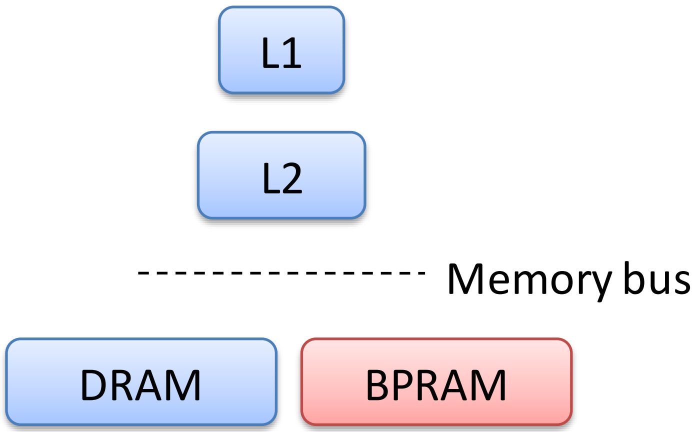
HD / Flash

BPRAM in the PC



- BPRAM and DRAM are **addressable** by the CPU
- Physical address space is **partitioned**
- BPRAM data may be **cached** in L1/L2

BPRAM in the PC

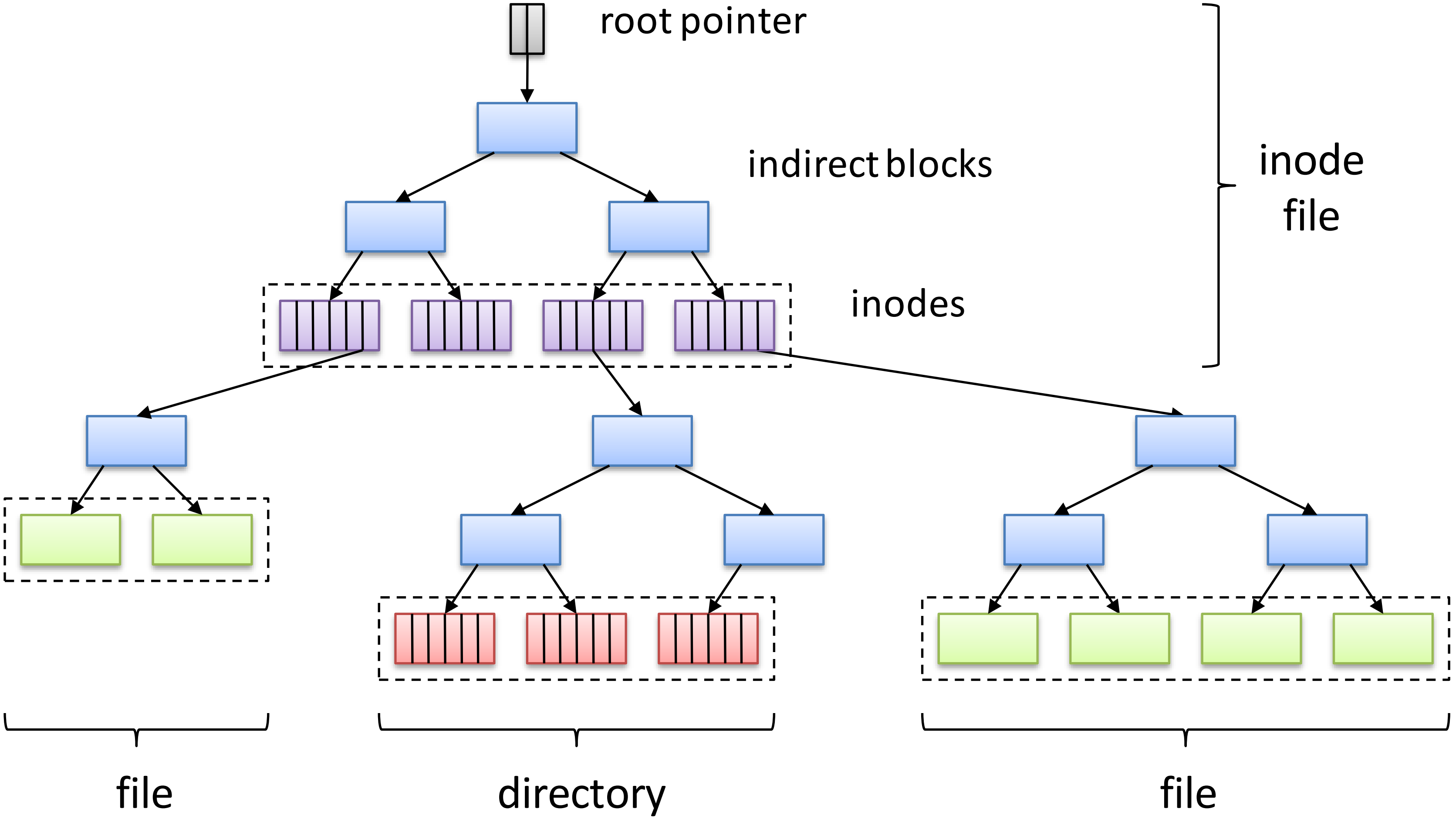


- BPRAM and DRAM are **addressable** by the CPU
- Physical address space is **partitioned**
- BPRAM data may be **cached** in L1/L2

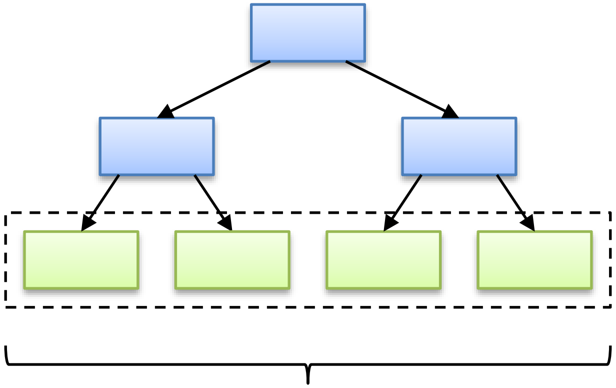
BPFS: A BPRAM File System

- Guarantees that all file operations execute atomically and in program order
- Despite guarantees, significant performance improvements over NTFS on the same media
- Short-circuit shadow paging often allows atomic, in-place updates

BPFS: A BPRAM File System



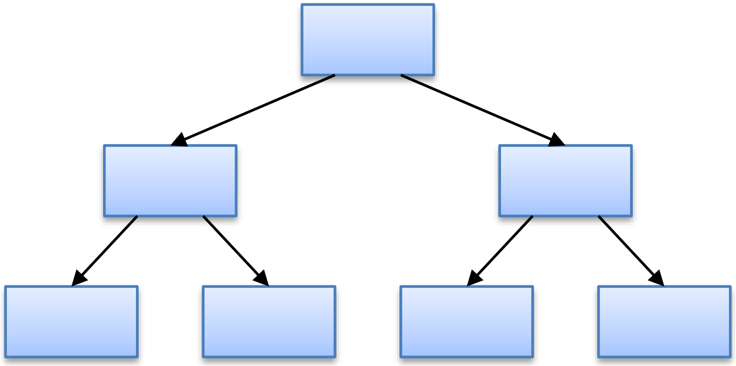
BPFS: A BPRAM File System



file

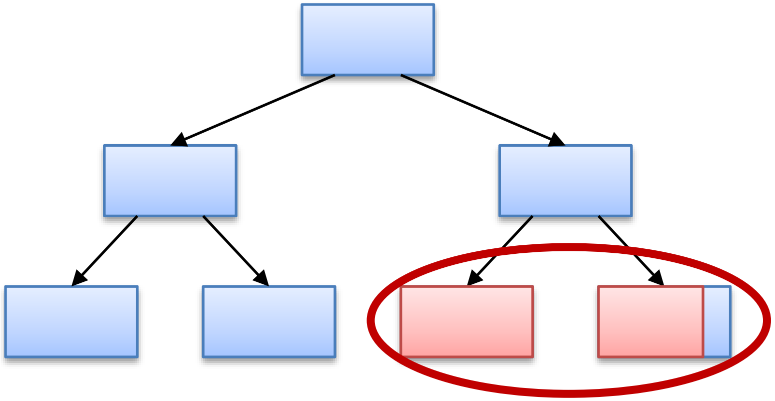
Enforcing FS Consistency Guarantees

- What happens if we crash during an update?



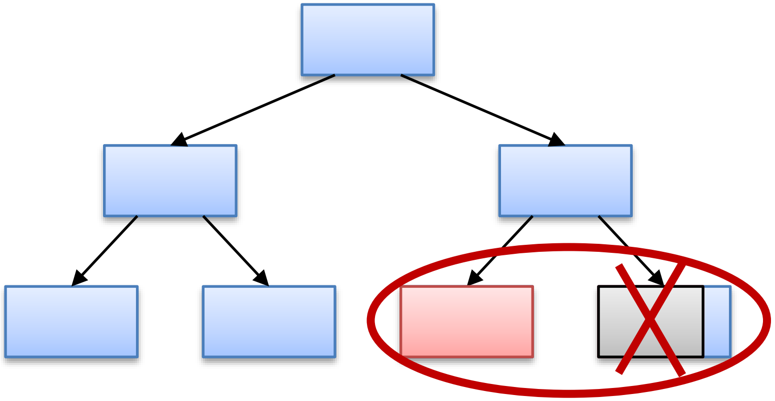
Enforcing FS Consistency Guarantees

- What happens if we crash during an update?



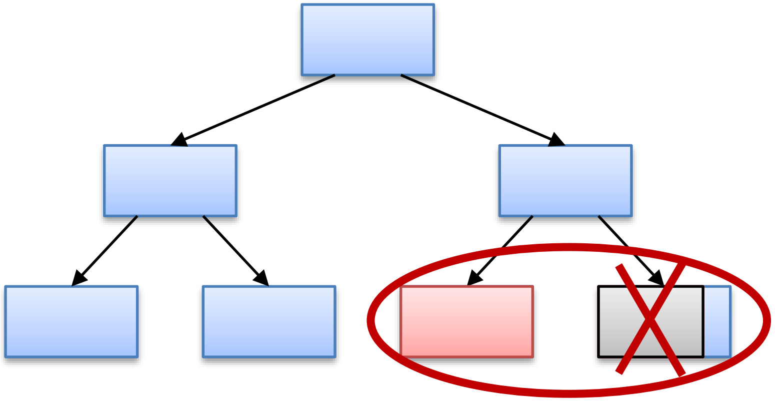
Enforcing FS Consistency Guarantees

- What happens if we crash during an update?



Enforcing FS Consistency Guarantees

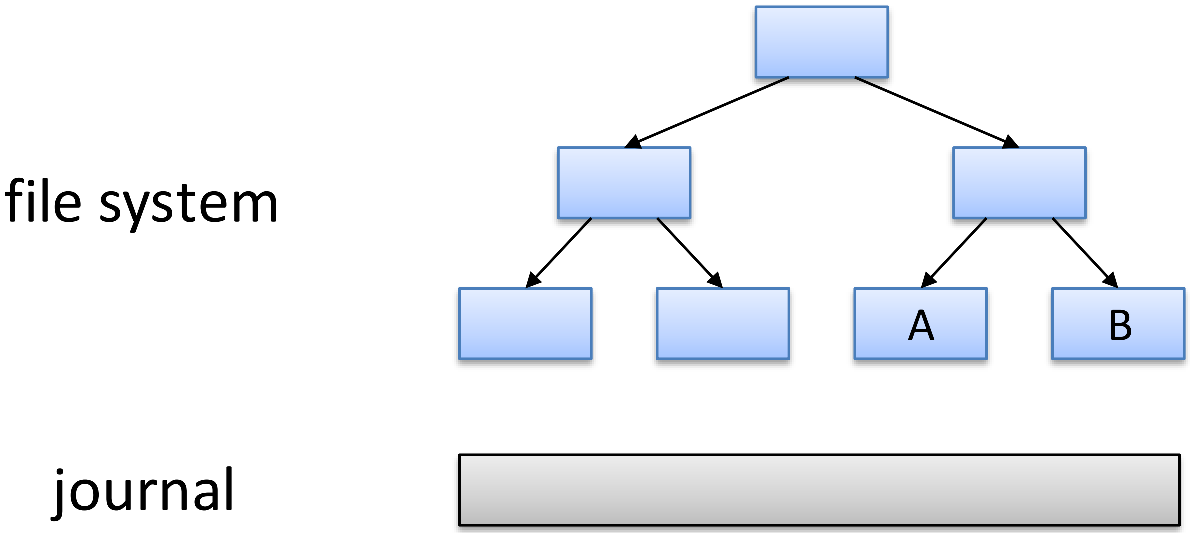
- What happens if we crash during an update?



  - Disk: Use journaling or shadow paging
  - BPRAM: Use short-circuit shadow paging

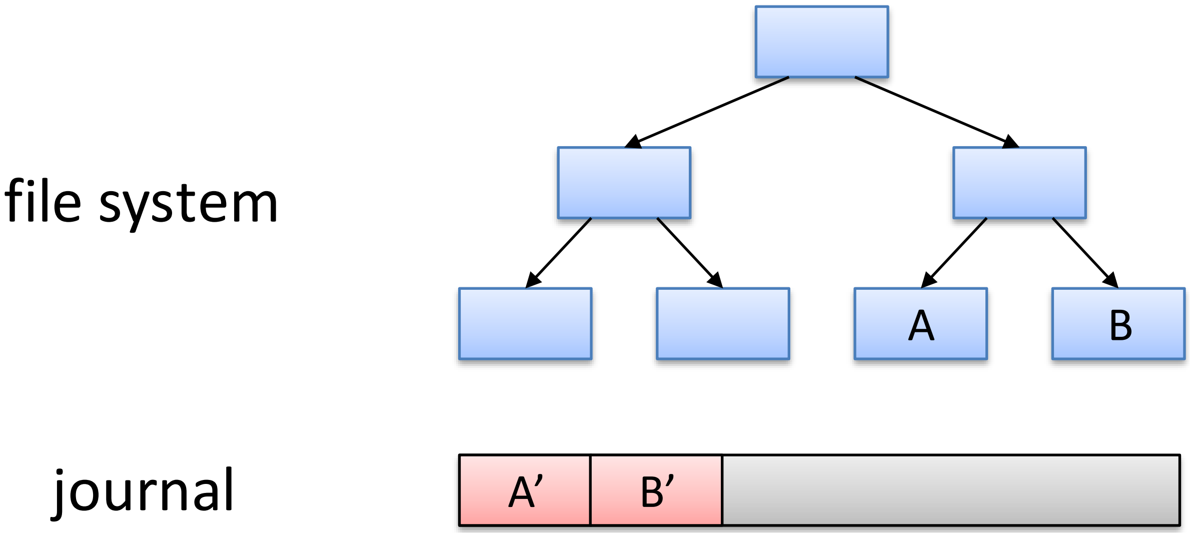
Review 1: Journaling

- Write to journal, then write to file system



Review 1: Journaling

- Write to journal, then write to file system



file system

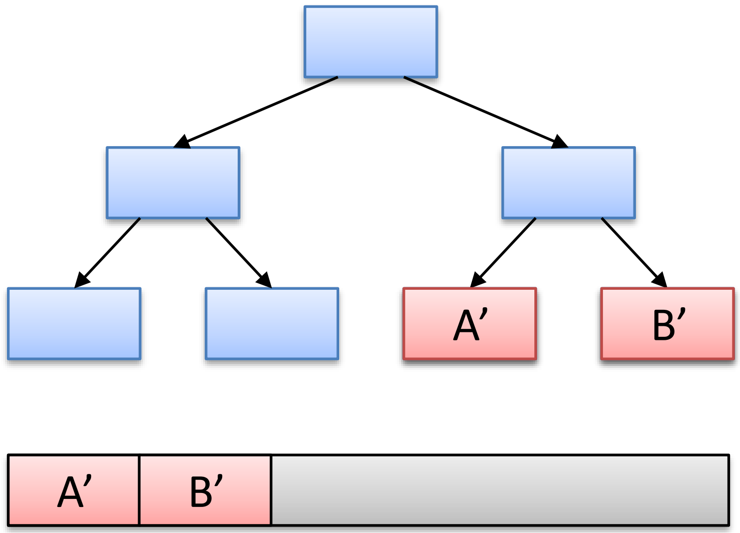
journal

Review 1: Journaling

- Write to journal, then write to file system

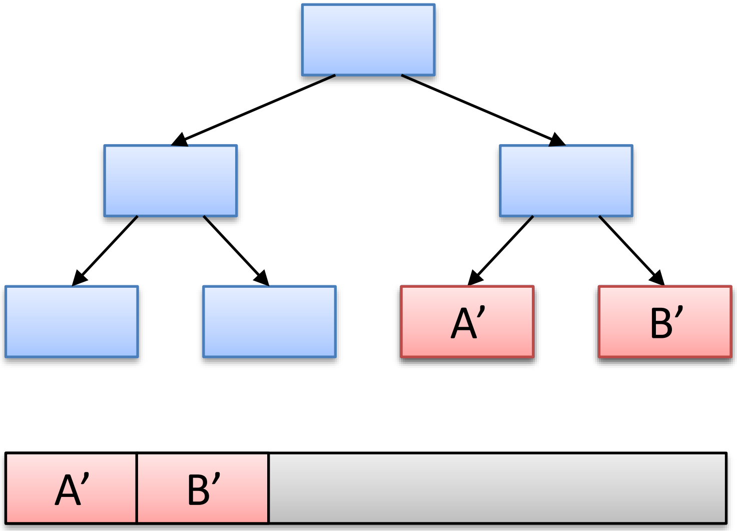
file system

journal



Review 1: Journaling

- Write to journal, then write to file system



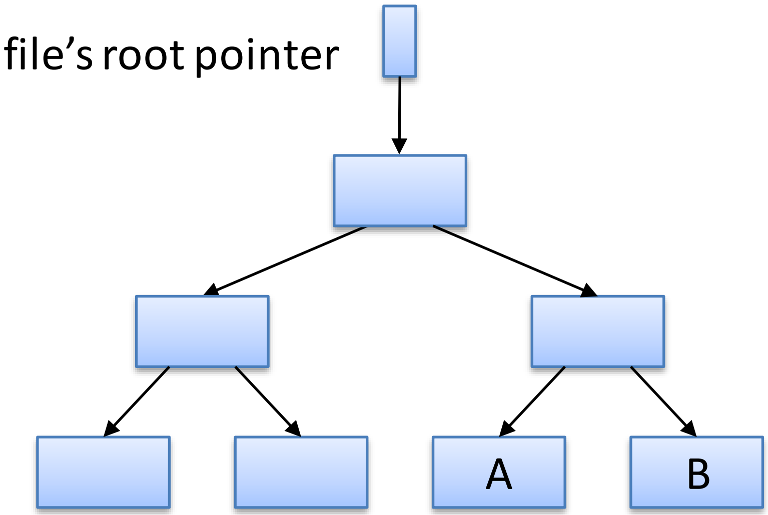
file system

journal

- Reliable, but all data is written twice

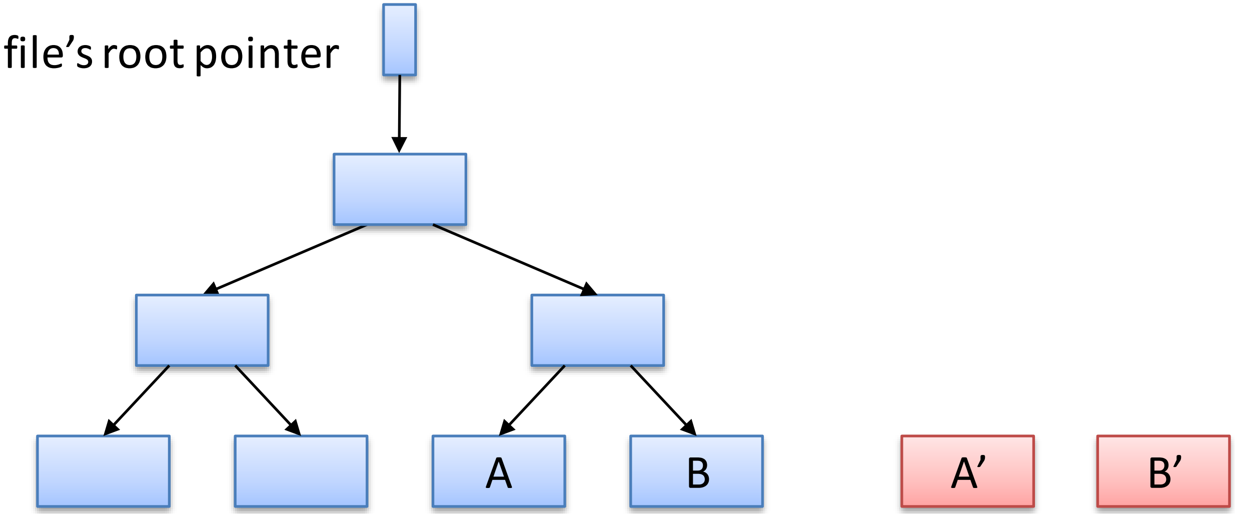
Review 2: Shadow Paging

- Use copy-on-write up to root of file system



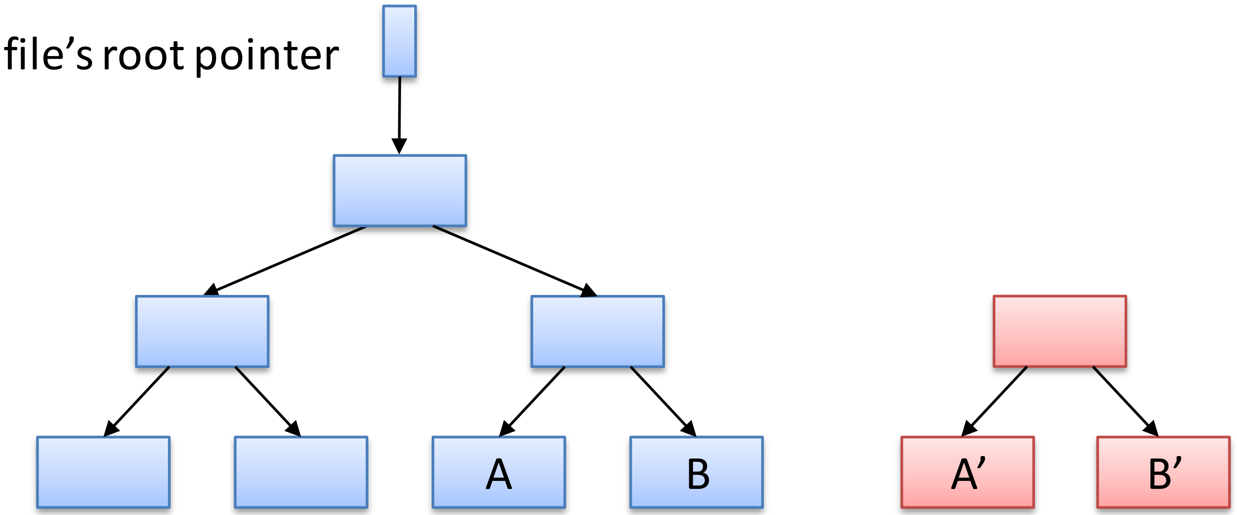
Review 2: Shadow Paging

- Use copy-on-write up to root of file system



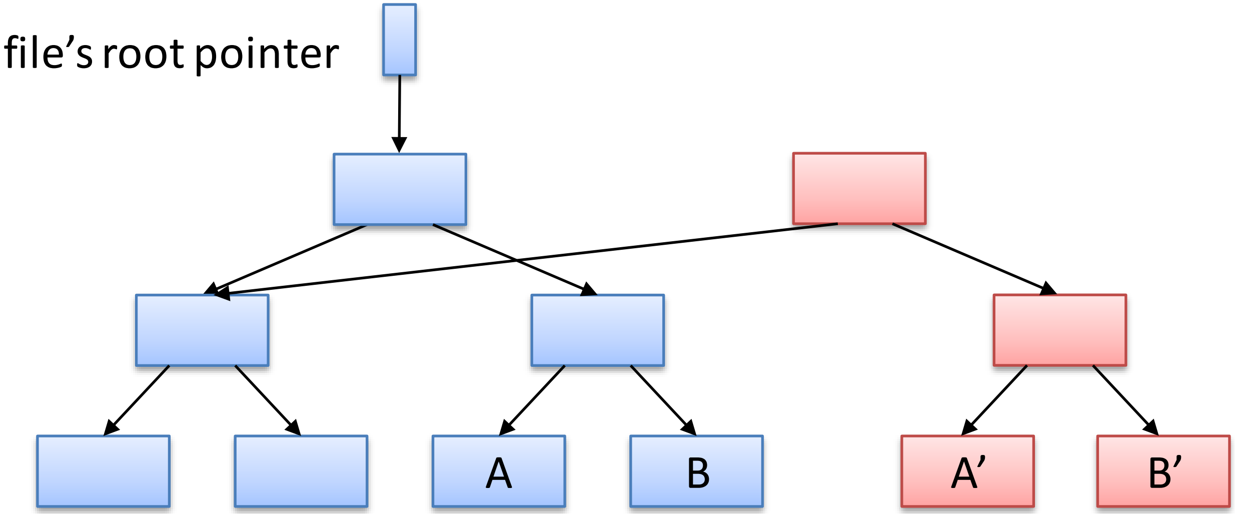
Review 2: Shadow Paging

- Use copy-on-write up to root of file system



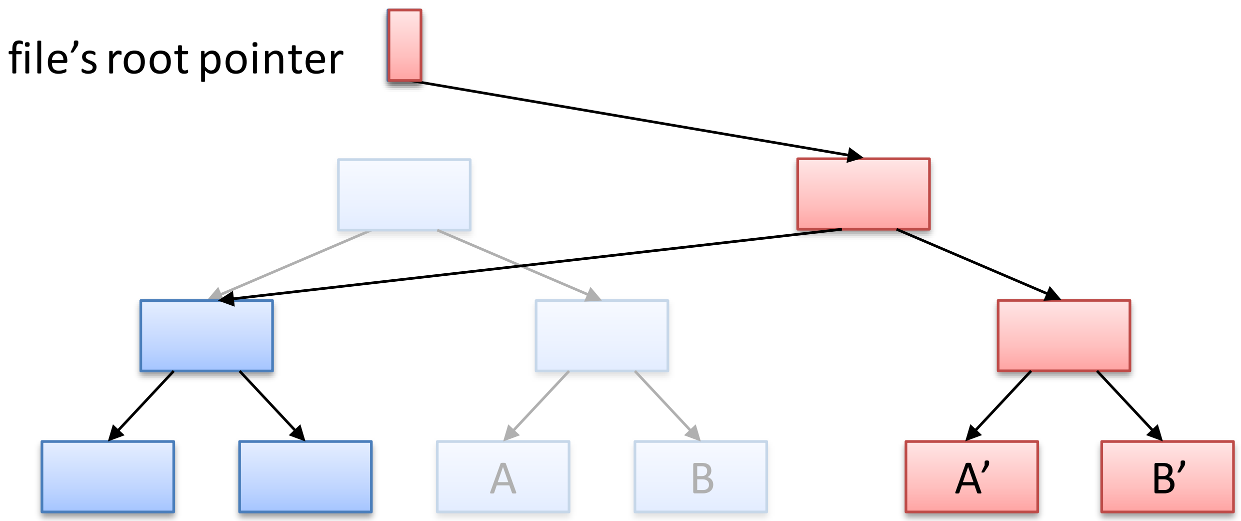
Review 2: Shadow Paging

- Use copy-on-write up to root of file system



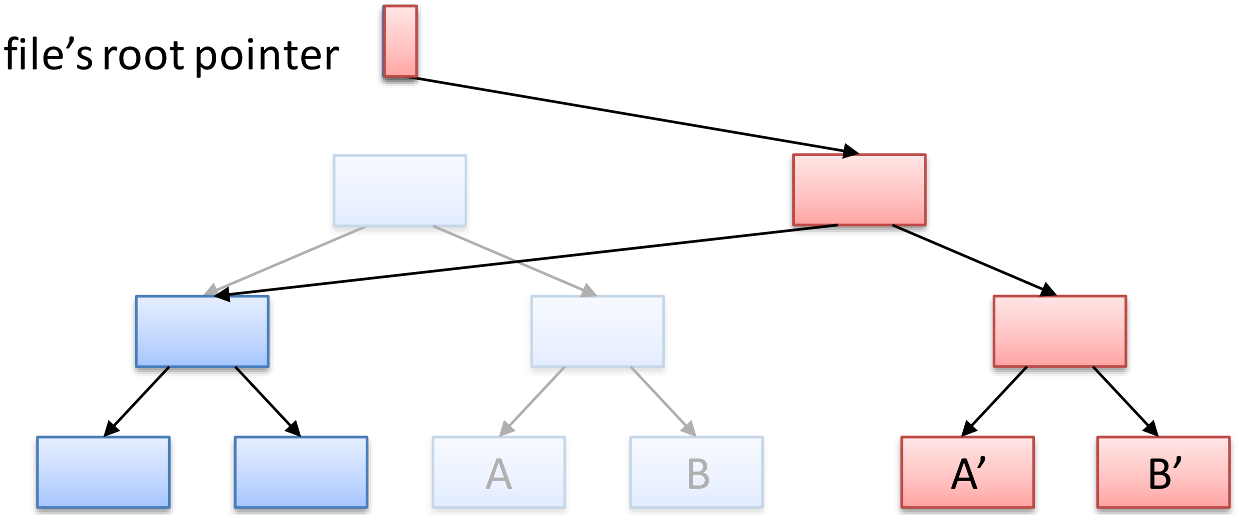
Review 2: Shadow Paging

- Use copy-on-write up to root of file system



Review 2: Shadow Paging

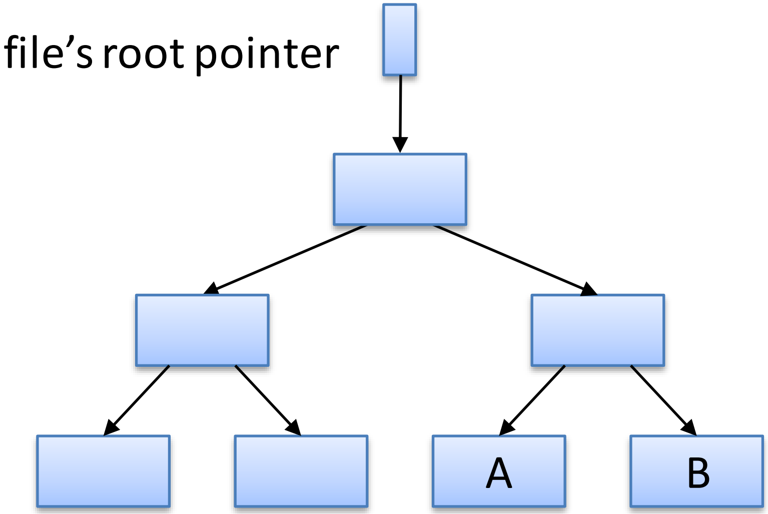
- Use copy-on-write up to root of file system



- Any change requires bubbling to the FS root
- Small writes require large copying overhead

Short-Circuit Shadow Paging

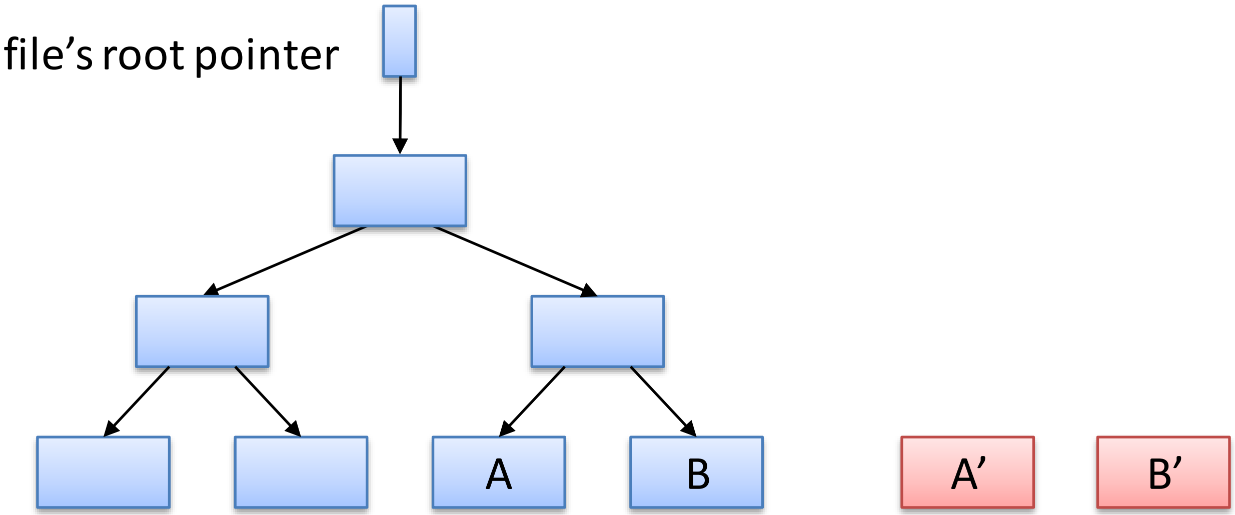
- Inspired by shadow paging
  - Optimization: In-place update when possible



- Uses byte-addressability and atomic 64b writes

Short-Circuit Shadow Paging

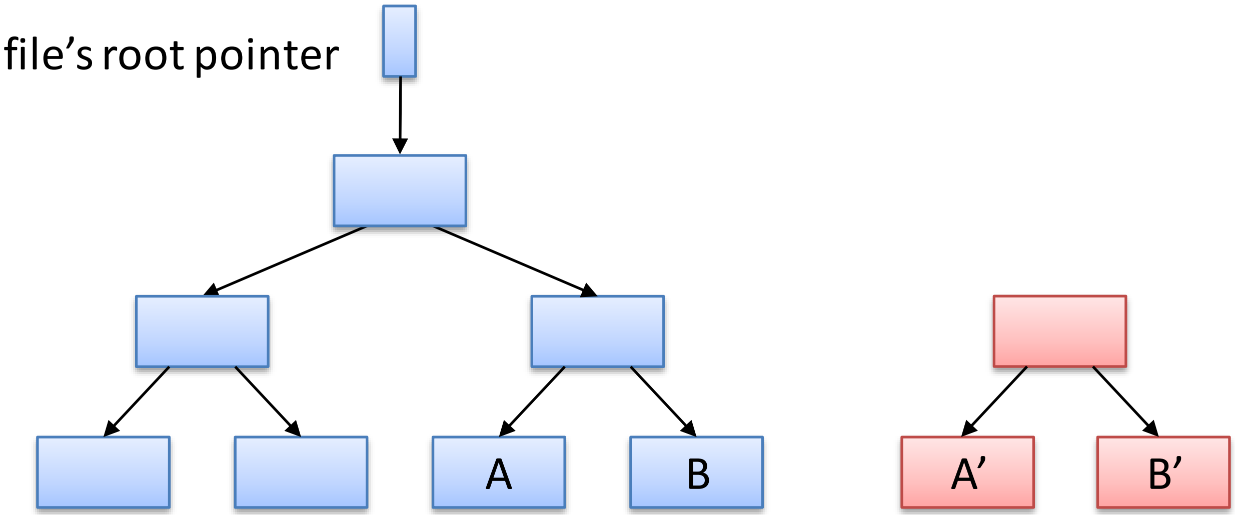
- Inspired by shadow paging
  - Optimization: In-place update when possible



- Uses byte-addressability and atomic 64b writes

Short-Circuit Shadow Paging

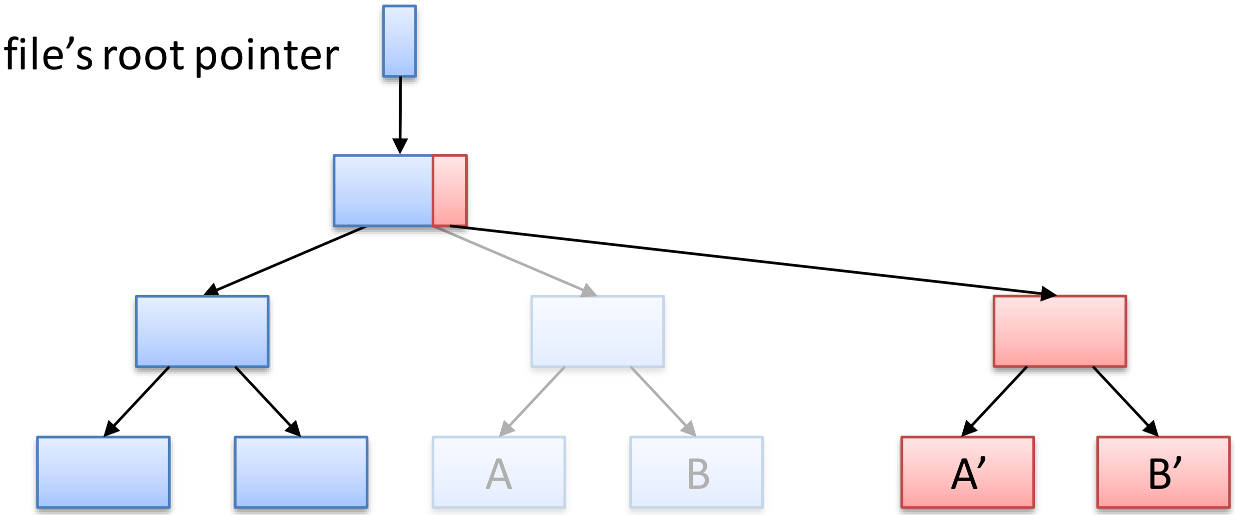
- Inspired by shadow paging
  - Optimization: In-place update when possible



- Uses byte-addressability and atomic 64b writes

Short-Circuit Shadow Paging

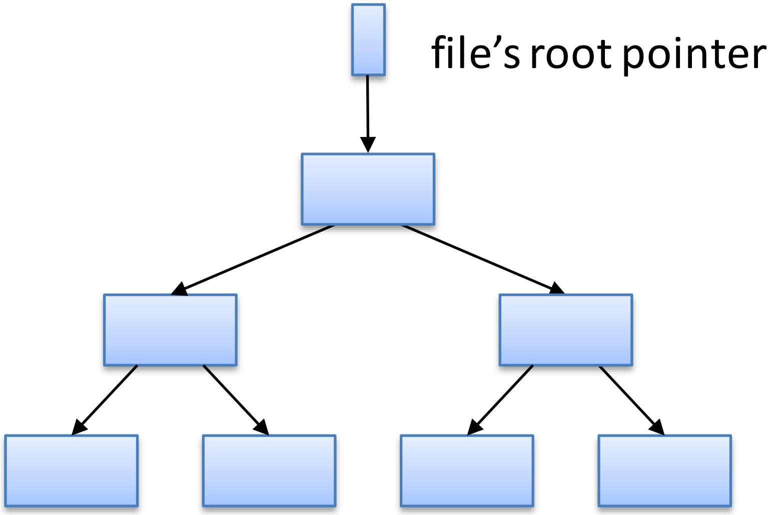
- Inspired by shadow paging
  - Optimization: In-place update when possible



- Uses byte-addressability and atomic 64b writes

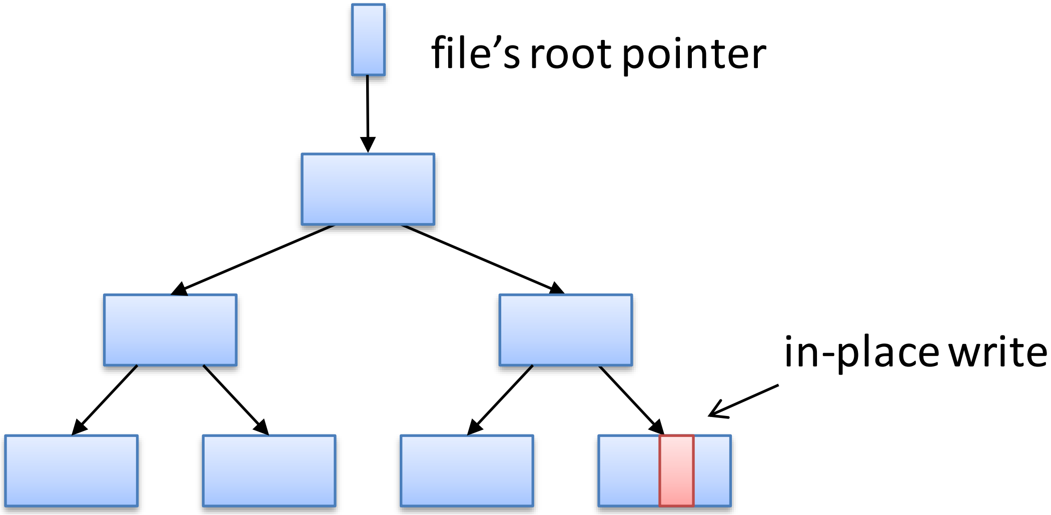
Opt. 1: In-Place Writes

- Aligned 64-bit writes are performed in place
  - Data and metadata



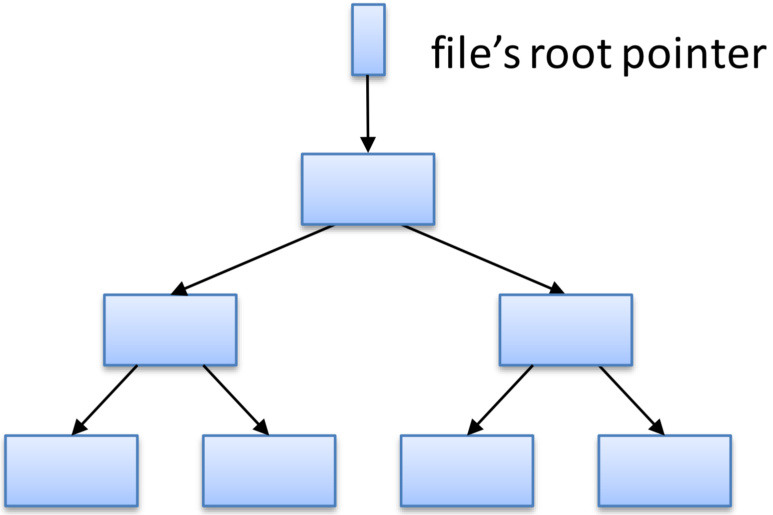
Opt. 1: In-Place Writes

- Aligned 64-bit writes are performed in place
  - Data and metadata



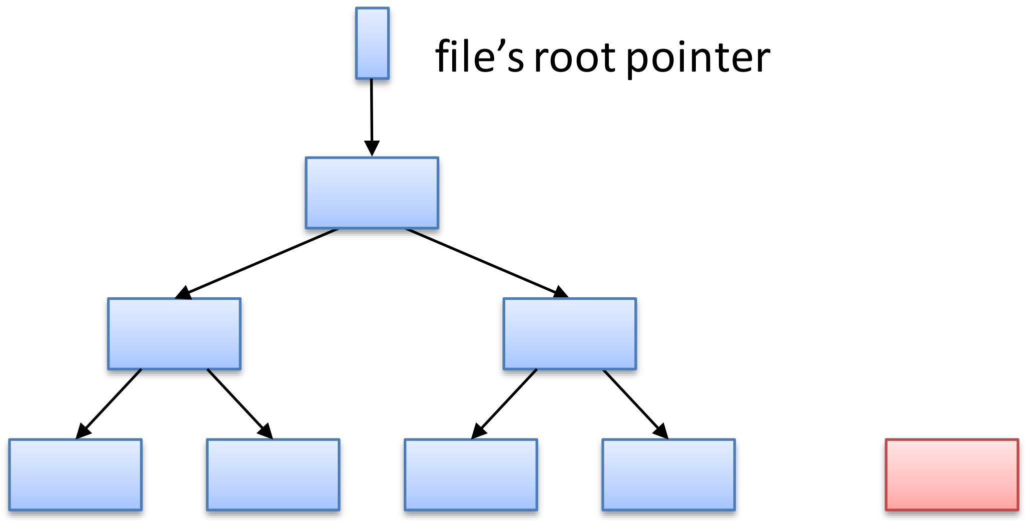
Opt. 1: In-Place Writes

- Aligned 64-bit writes are performed in place
  - Data and metadata



Opt. 1: In-Place Writes

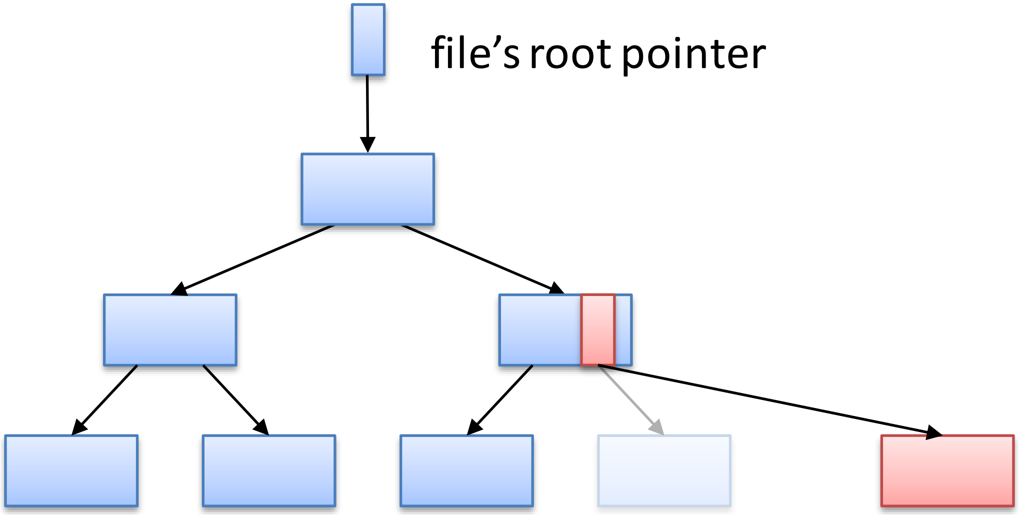
- Aligned 64-bit writes are performed in place
  - Data and metadata



file's root pointer

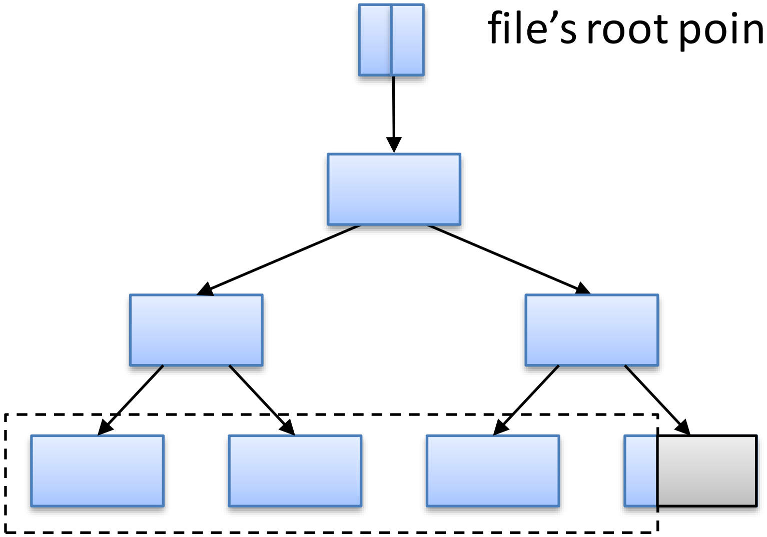
Opt. 1: In-Place Writes

- Aligned 64-bit writes are performed in place
  - Data and metadata



Opt. 2: Exploit Data-Metadata Invariants

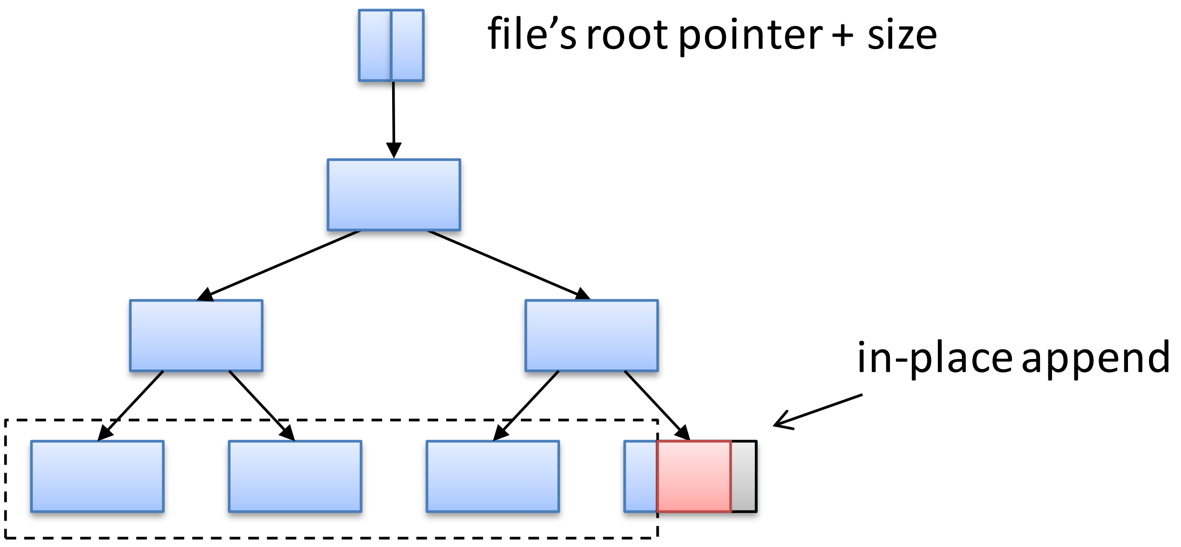
- Appends committed by updating file size



file's root pointer + size

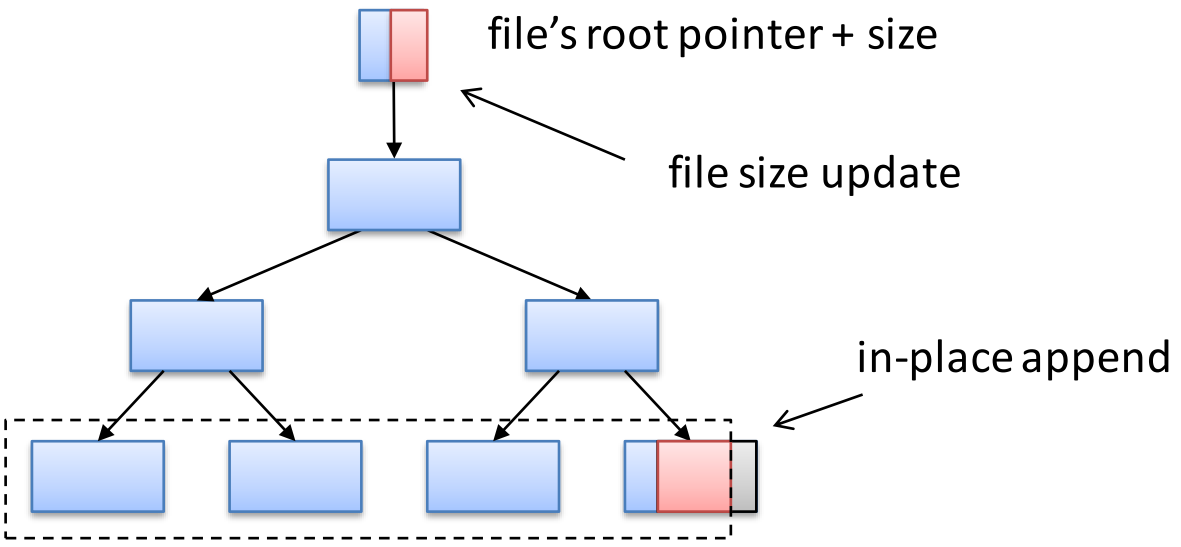
Opt. 2: Exploit Data-Metadata Invariants

- Appends committed by updating file size

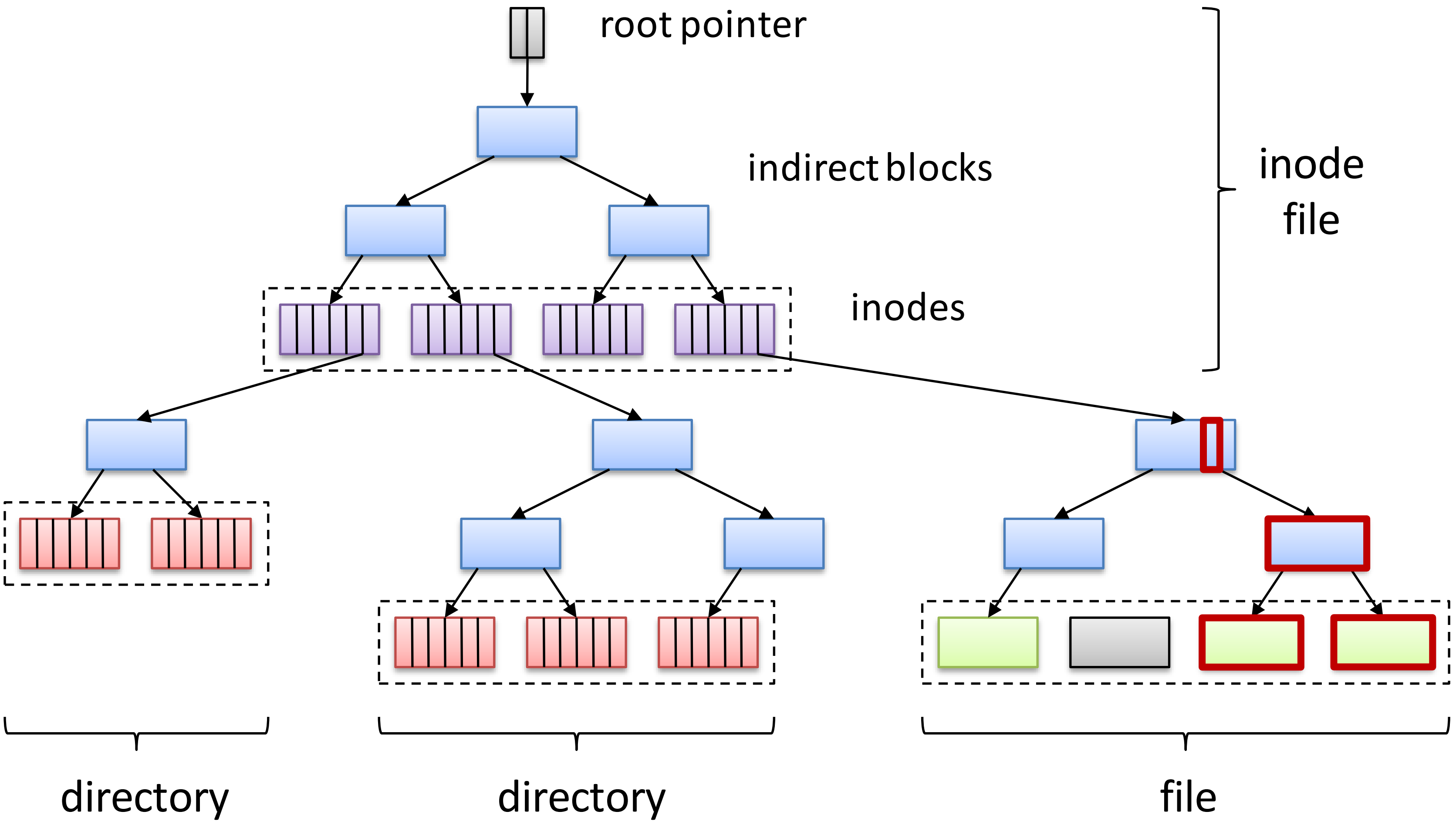


Opt. 2: Exploit Data-Metadata Invariants

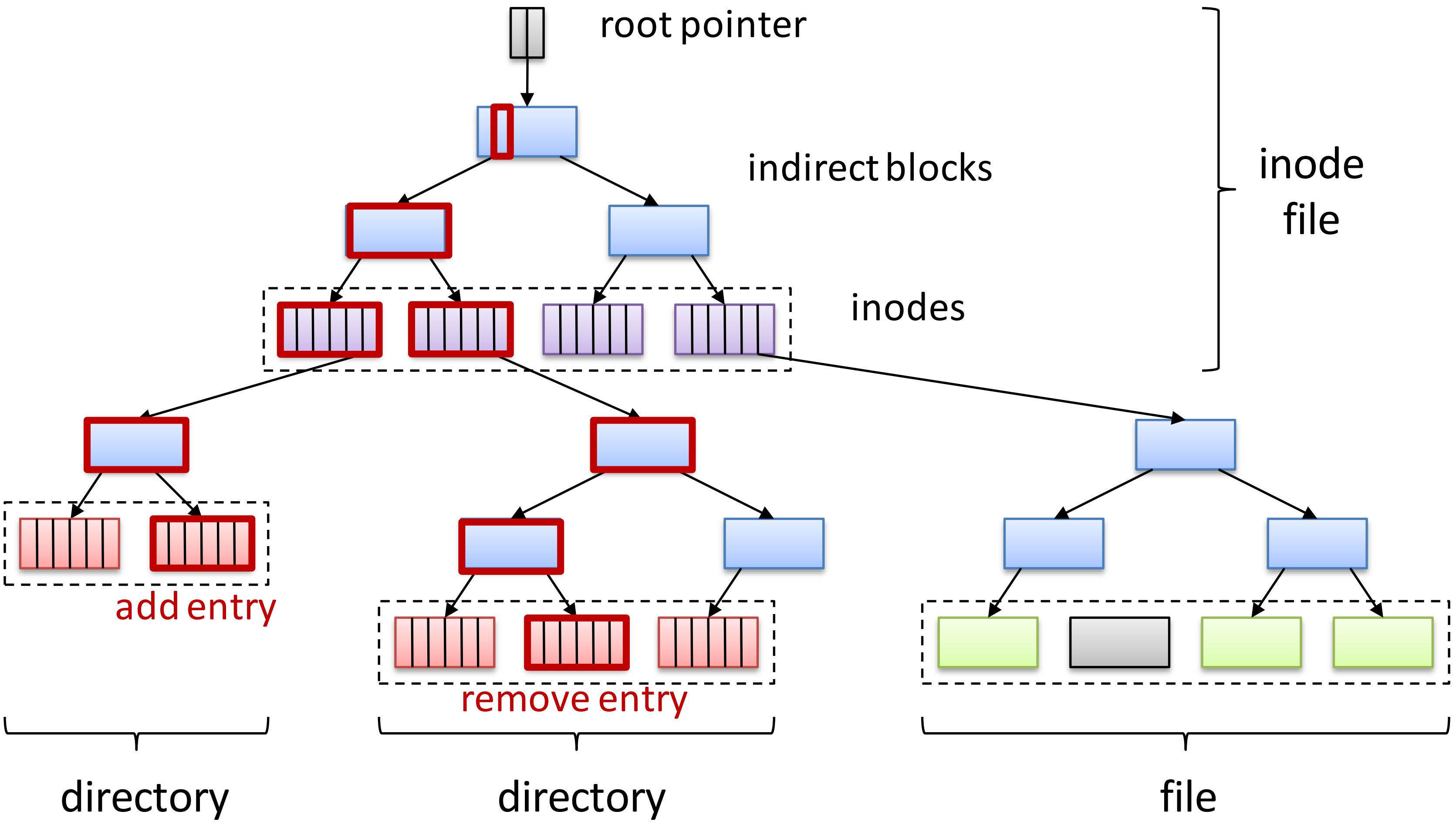
- Appends committed by updating file size



BPFS Example

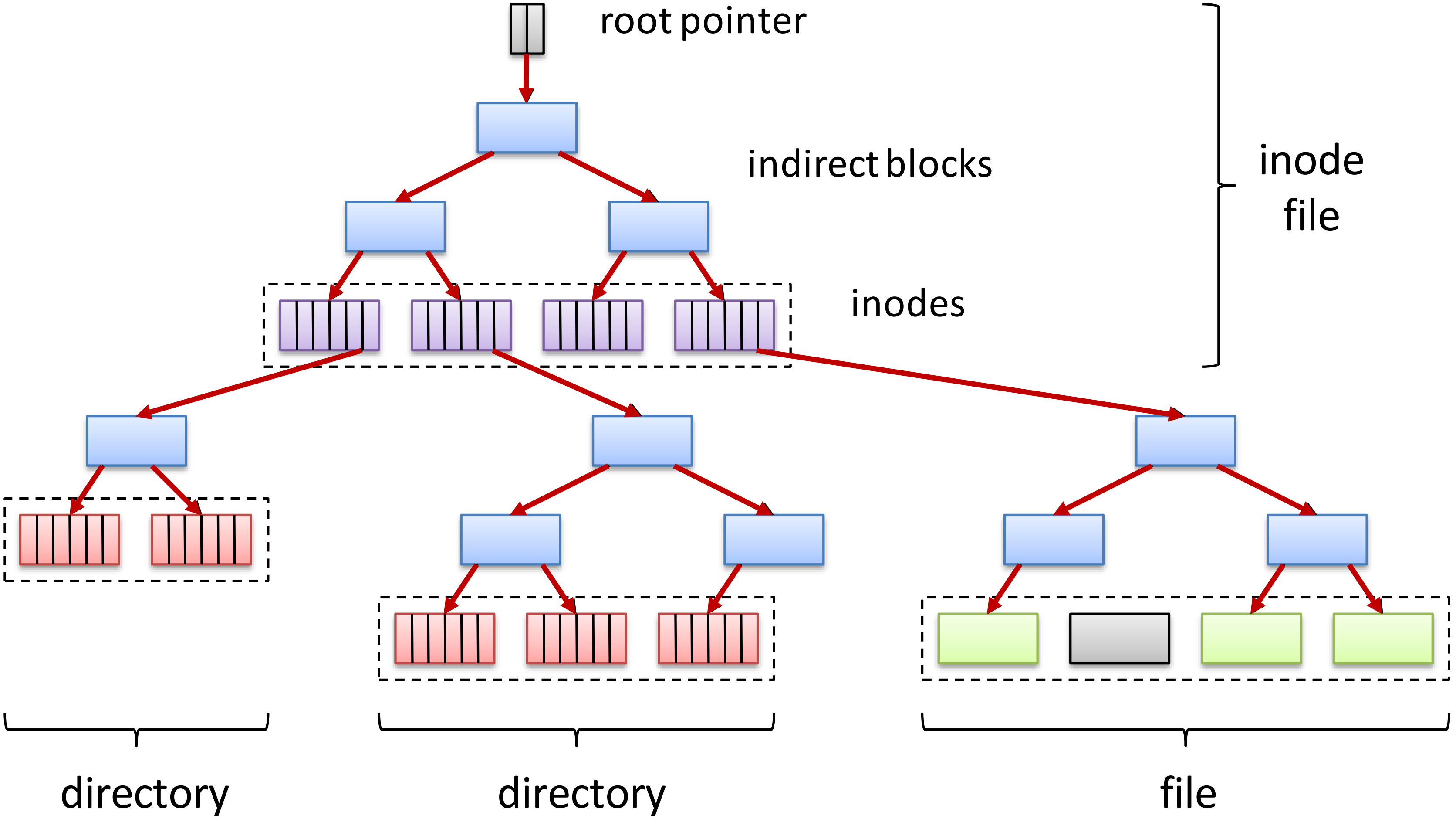


BPFS Example



- Cross-directory rename bubbles to common ancestor

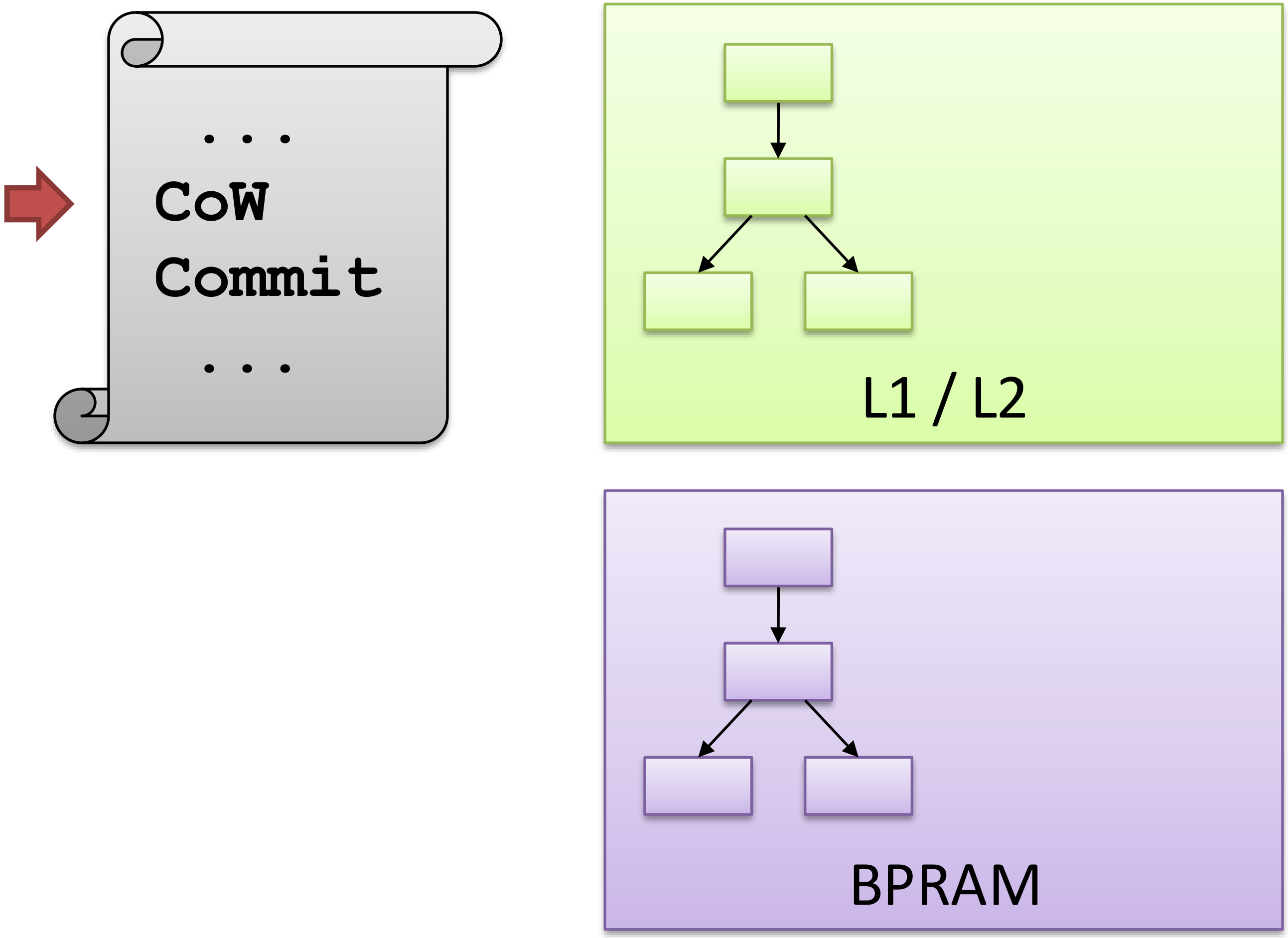
BPFS Example



Outline

- Intro
- File System
- **Hardware Support**
- Evaluation
- Conclusion

Problem 1: Ordering

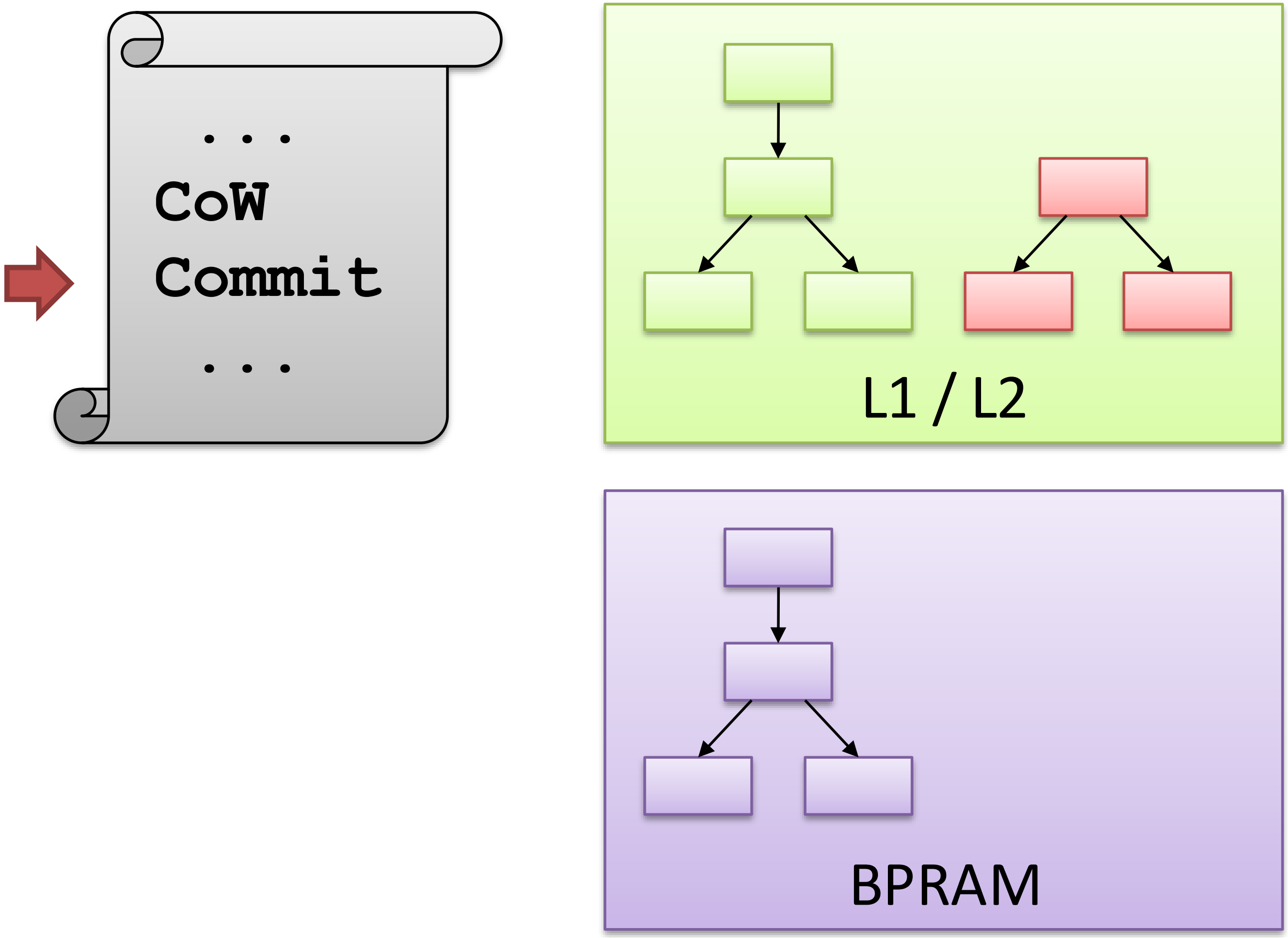


. . .
CoW
Commit
. . .

L1 / L2

BPRAM

Problem 1: Ordering



. . .

CoW

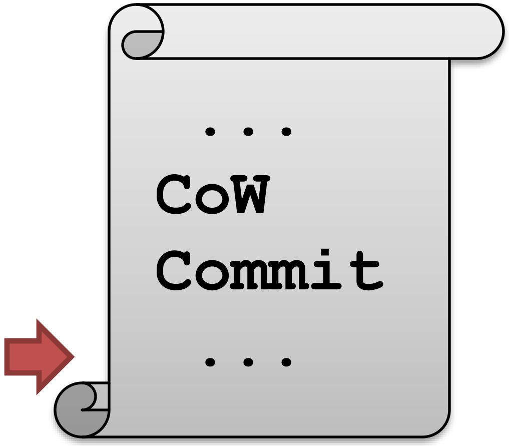
Commit

. . .

L1 / L2

BPRAM

Problem 1: Ordering

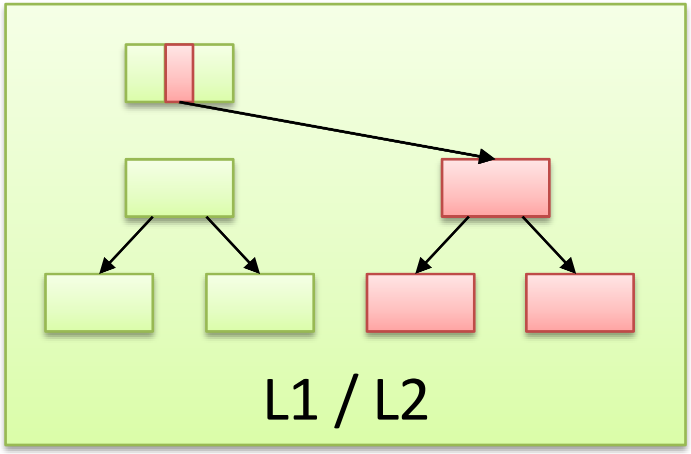


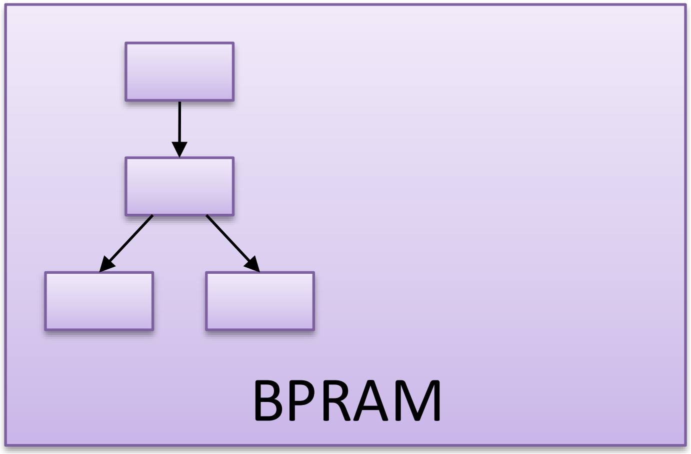
. . .

CoW

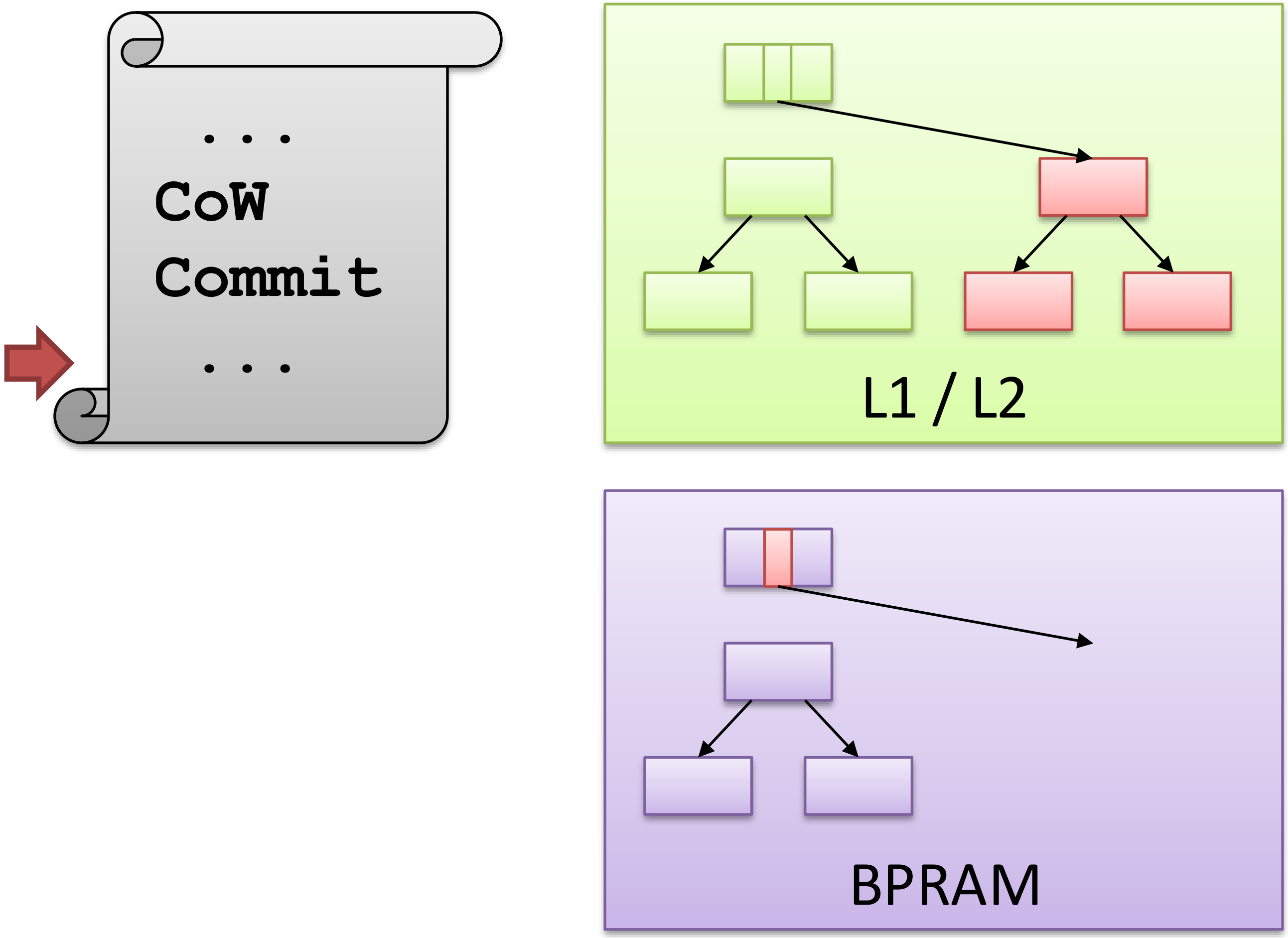
Commit

. . .





Problem 1: Ordering



. . .

CoW

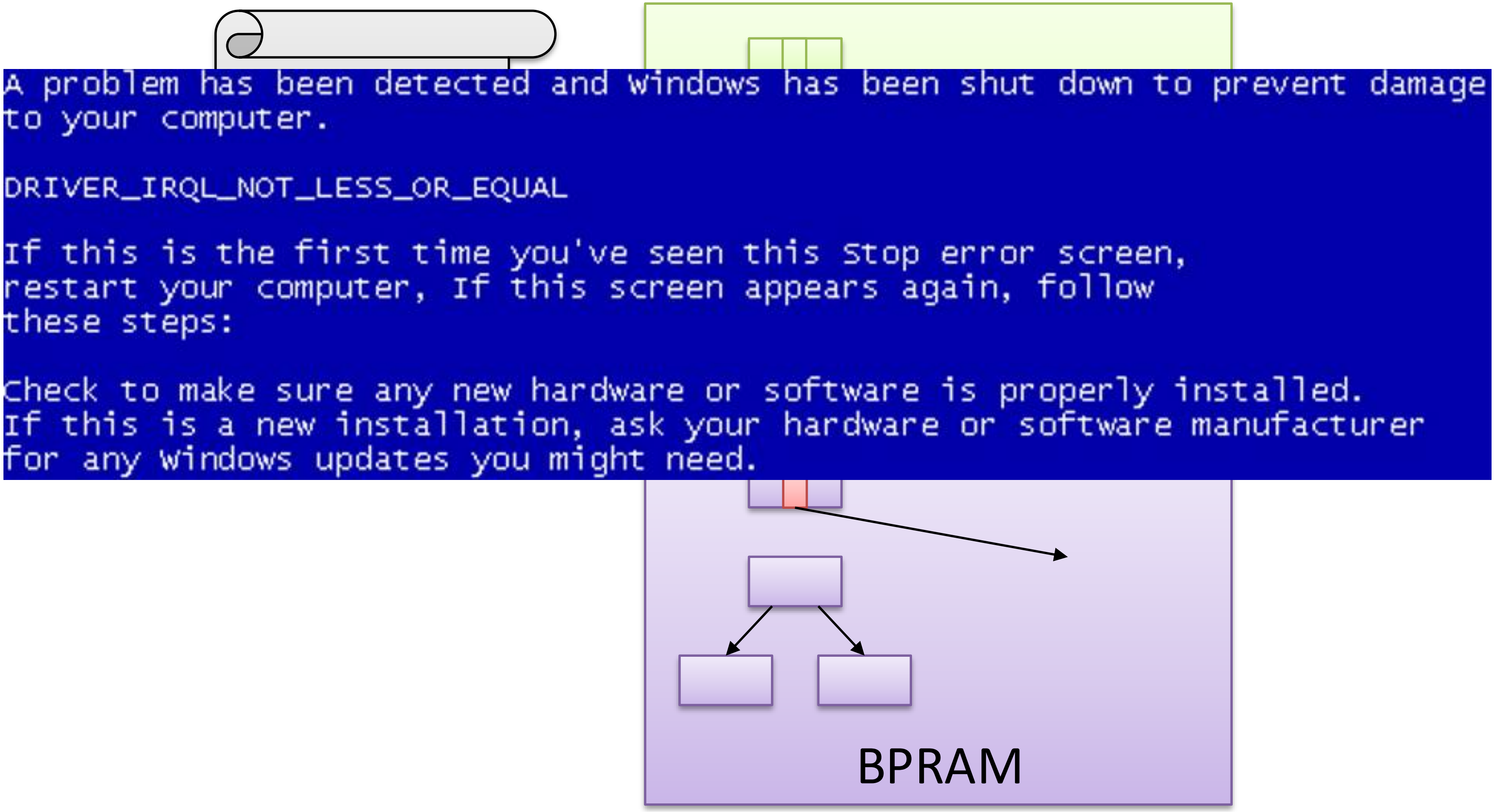
Commit

. . .

L1 / L2

BPRAM

Problem 1: Ordering



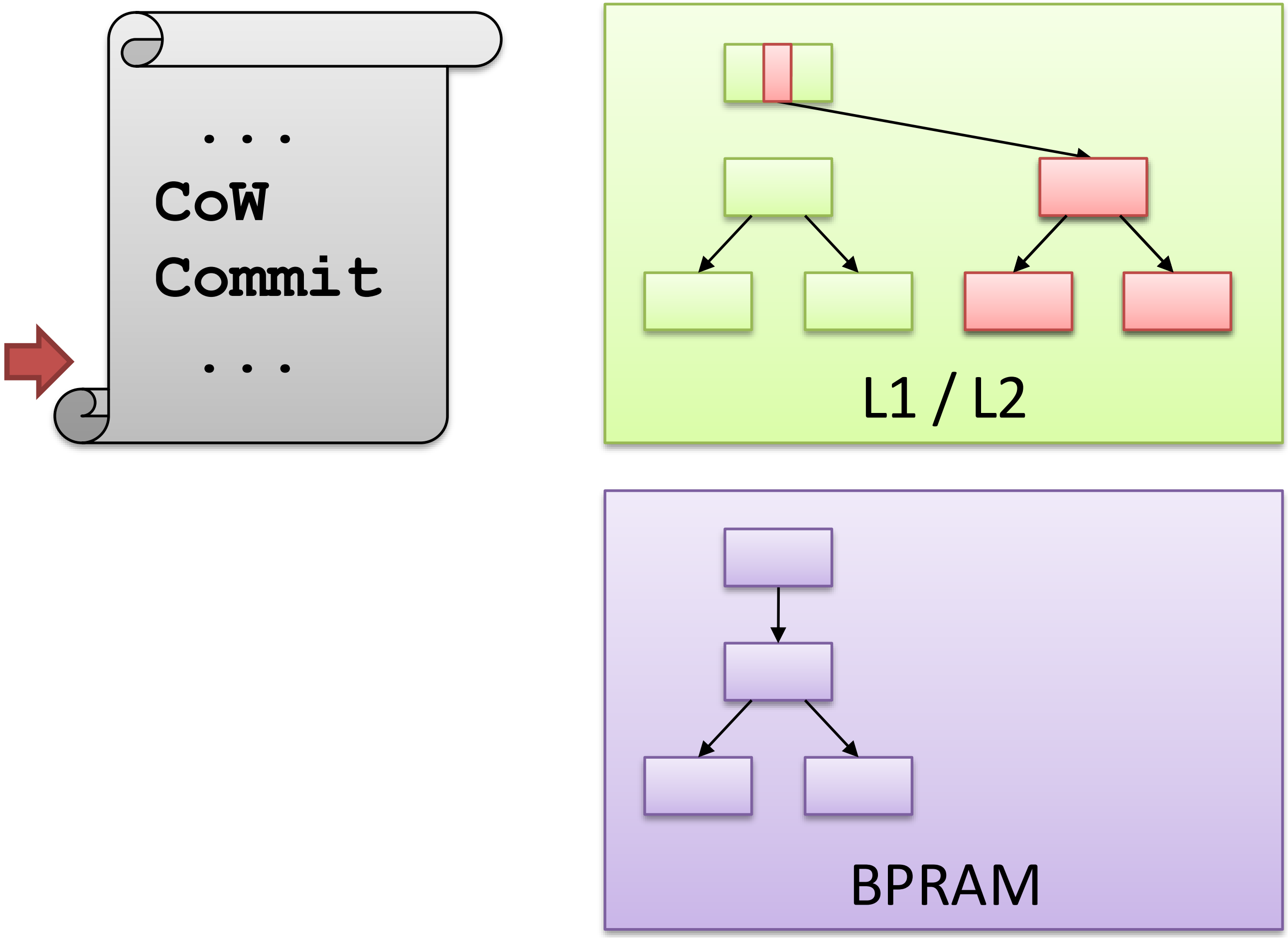
```
A problem has been detected and windows has been shut down to prevent damage
to your computer.

DRIVER_IRQL_NOT_LESS_OR_EQUAL

If this is the first time you've seen this Stop error screen,
restart your computer, If this screen appears again, follow
these steps:

Check to make sure any new hardware or software is properly installed.
If this is a new installation, ask your hardware or software manufacturer
for any Windows updates you might need.
```

Problem 2: Atomicity

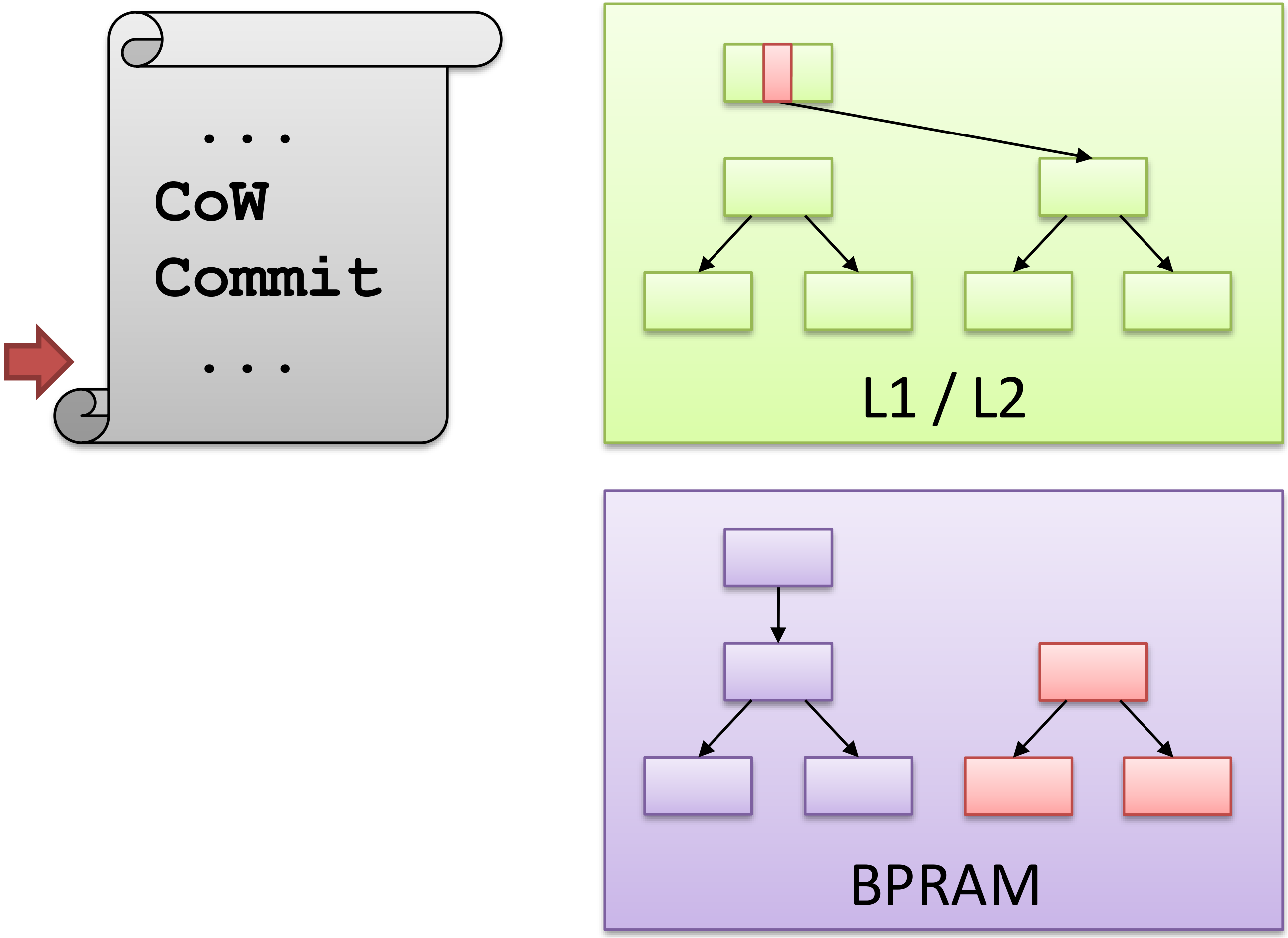


```
...
CoW
Commit
...
```

L1 / L2

BPRAM

Problem 2: Atomicity

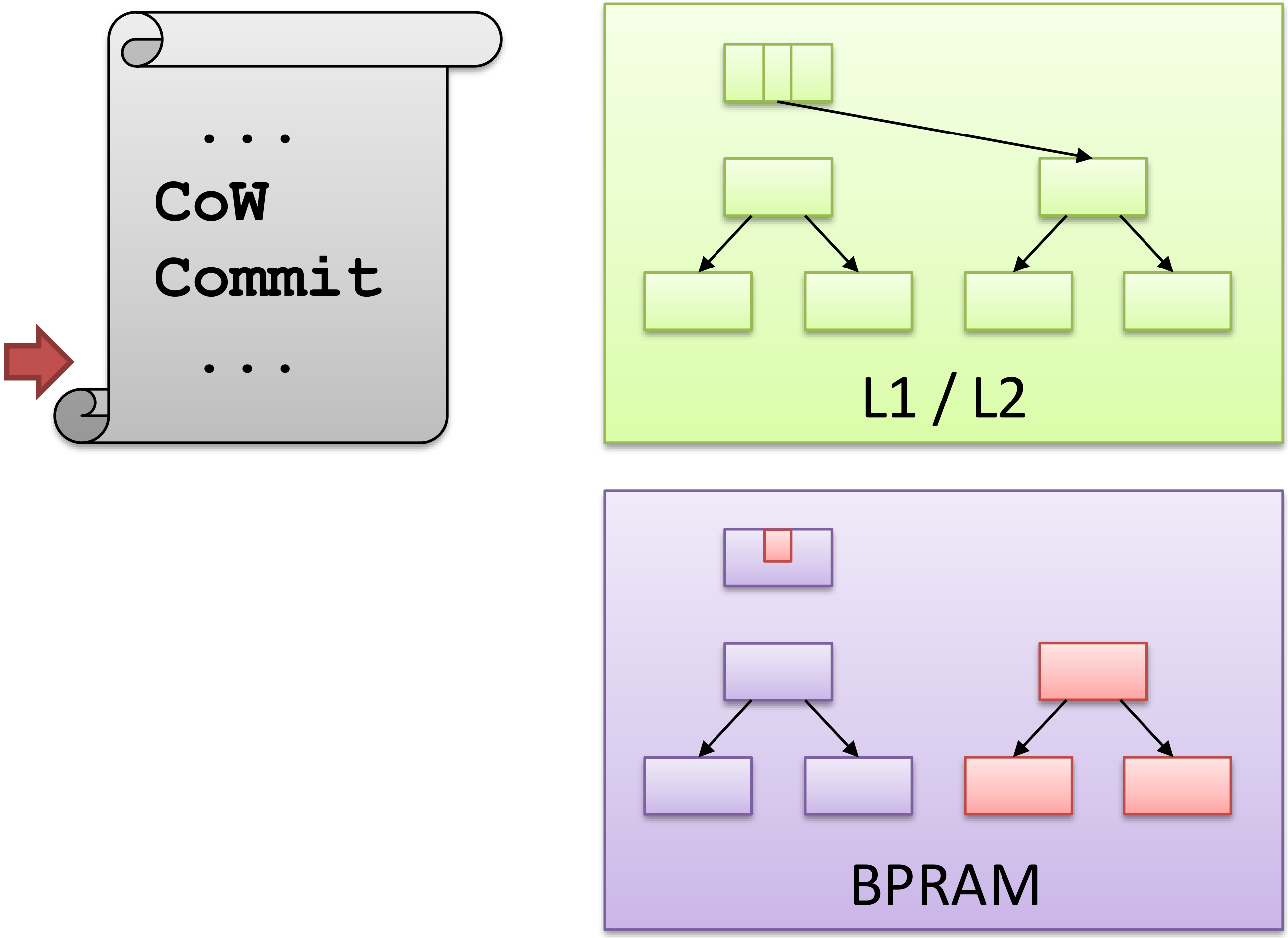


```
. . .
CoW
Commit
. . .
```

L1 / L2

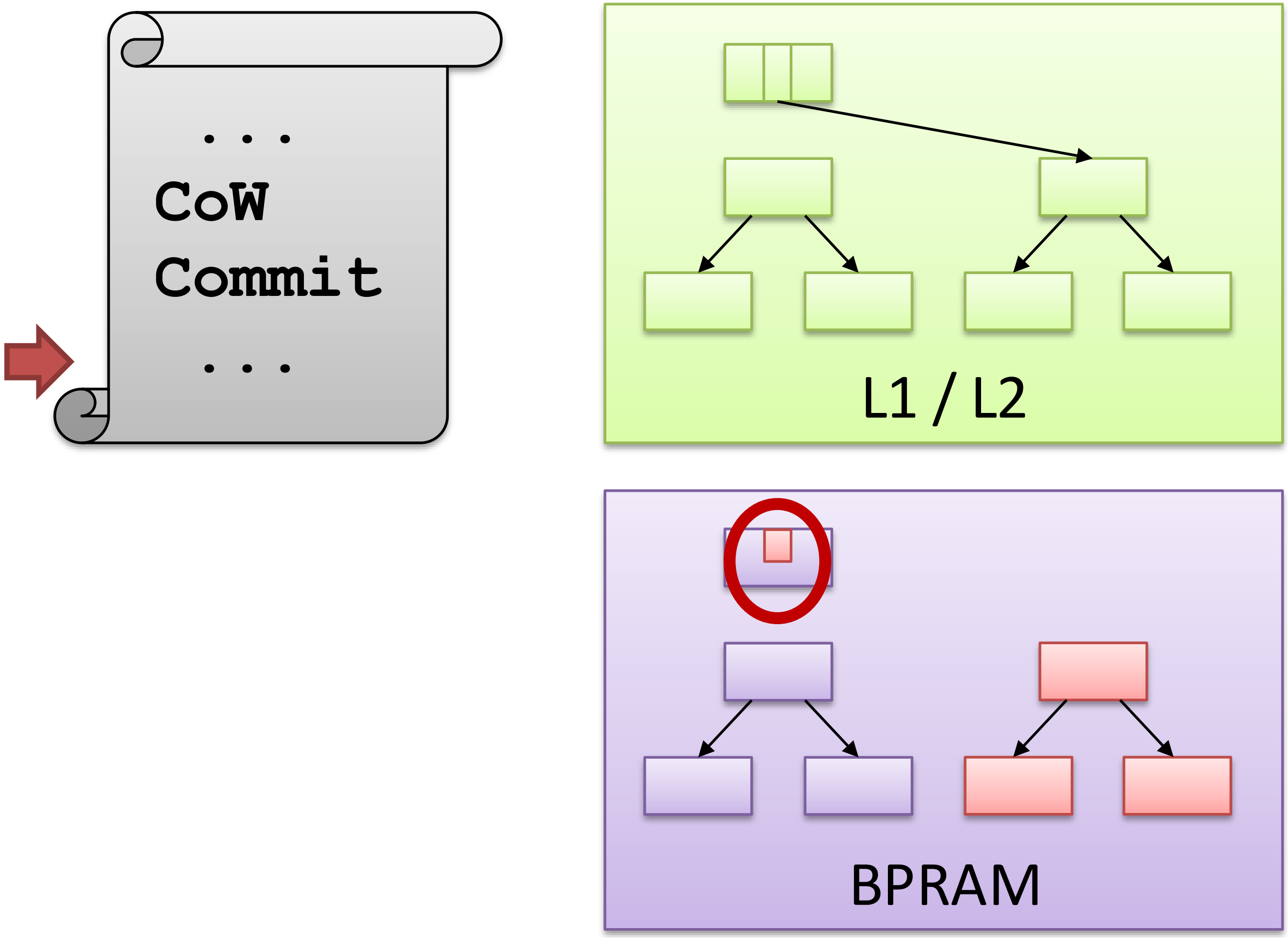
BPRAM

Problem 2: Atomicity



```
  . . .
CoW
Commit
  . . .
```

Problem 2: Atomicity



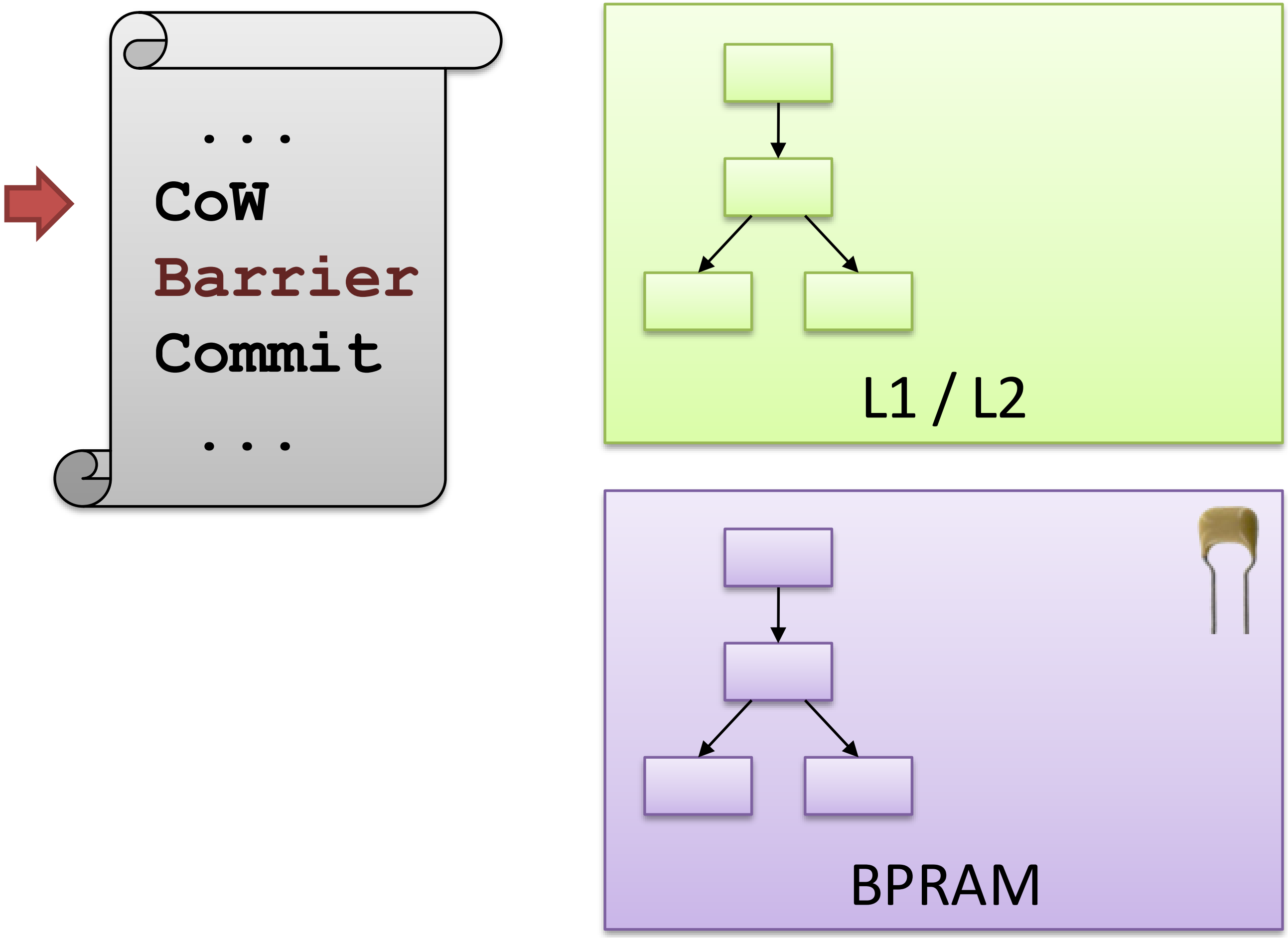
```
...
CoW
Commit
...
```

Enforcing Ordering and Atomicity

- Ordering
  - Solution: **Epoch barriers** to declare constraints
  - Faster than write-through
  - Important hardware primitive (cf. SCSI TCQ)

- Atomicity
  - Solution: **Capacitor** on DIMM
  - Simple and cheap!

Ordering and Atomicity

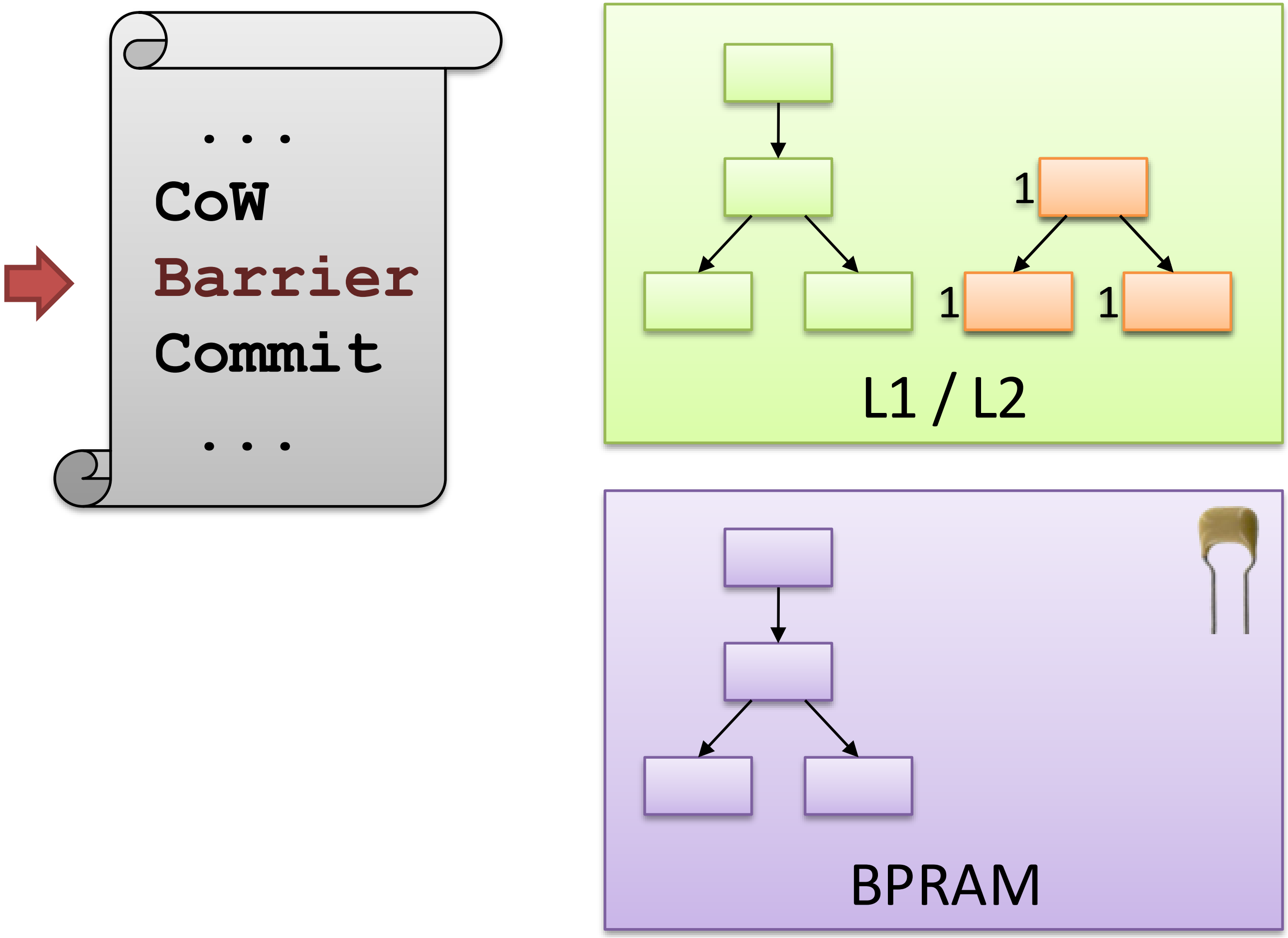


```
. . .
CoW
Barrier
Commit
. . .
```

L1 / L2

BPRAM

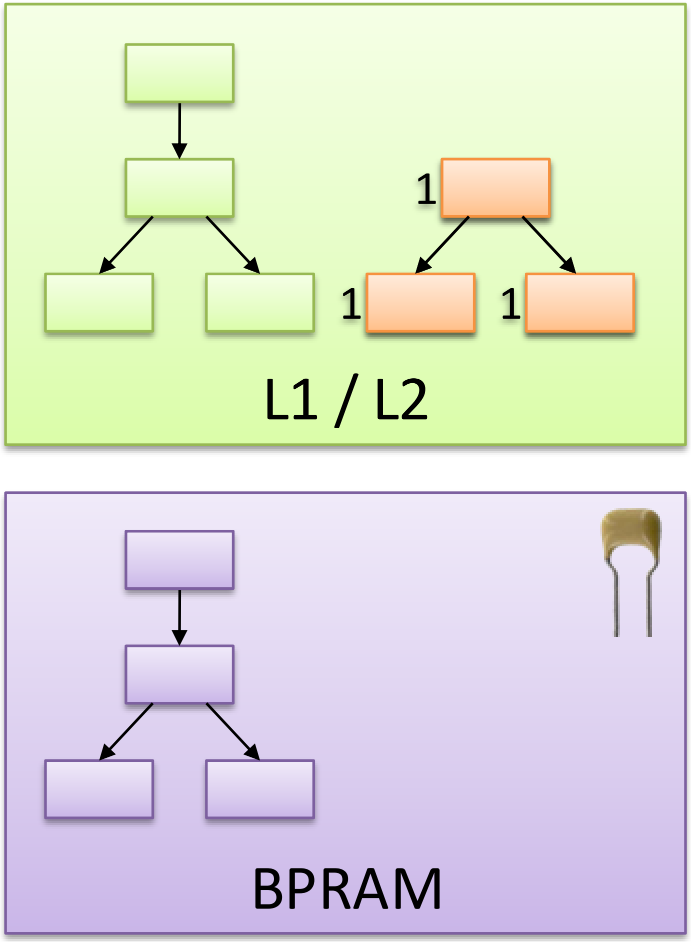
Ordering and Atomicity



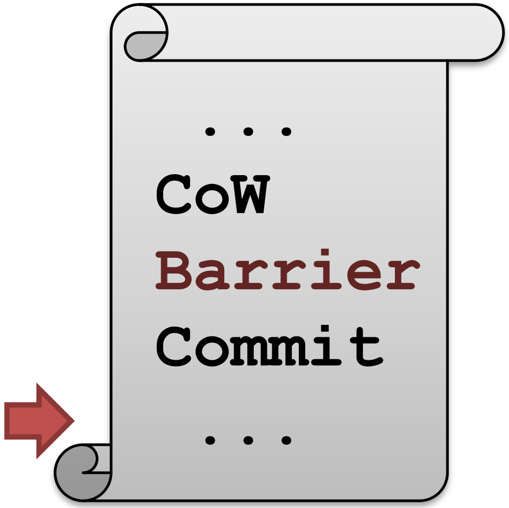
```
. . .
CoW
Barrier
Commit
. . .
```

Ordering and Atomicity

```
...
CoW
Barrier
Commit
...
```



Ordering and Atomicity



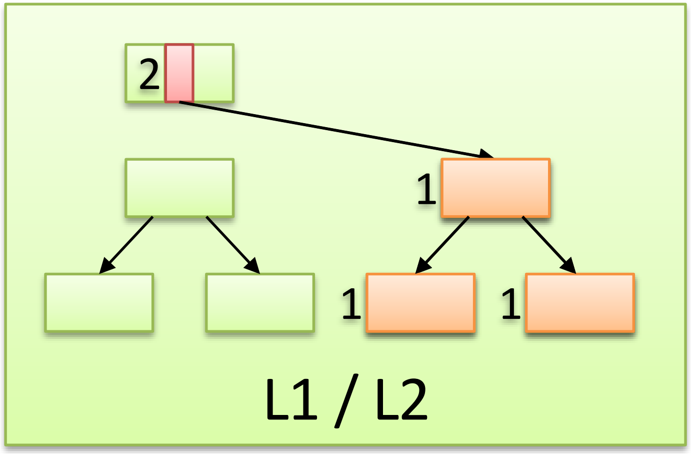
. . .

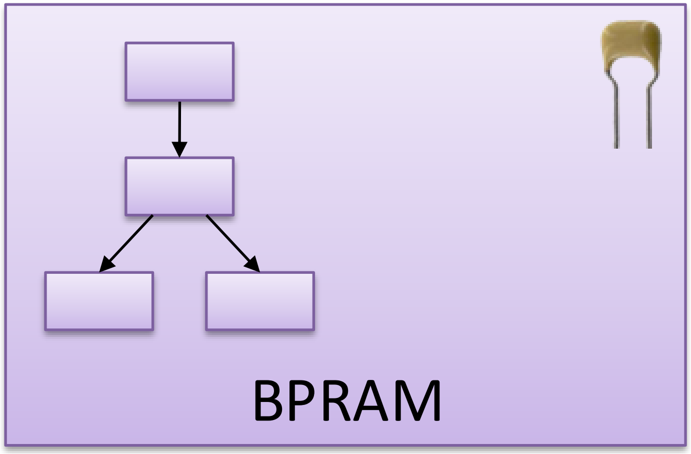
CoW

Barrier

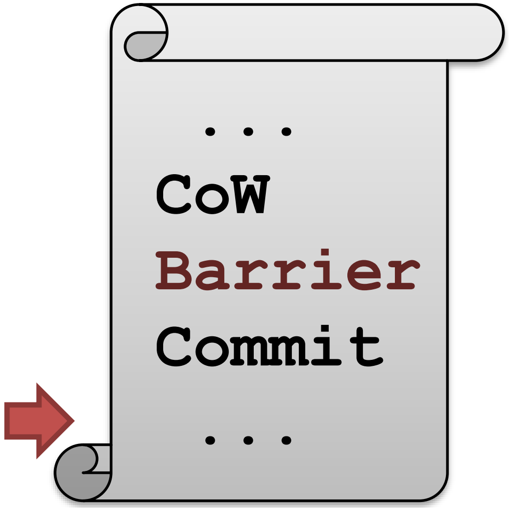
Commit

. . .

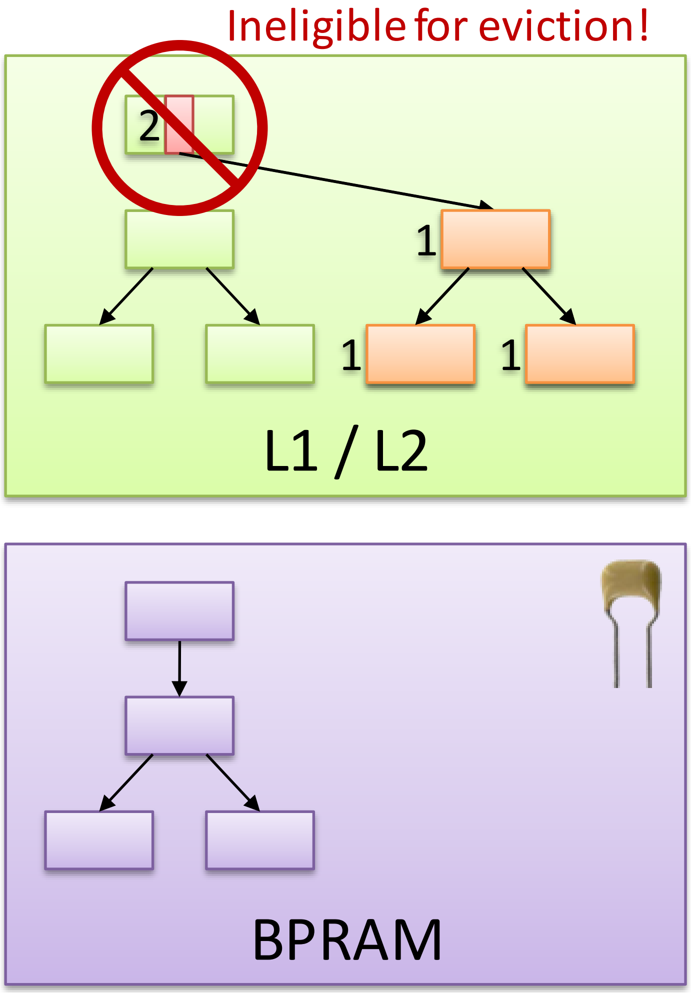




Ordering and Atomicity



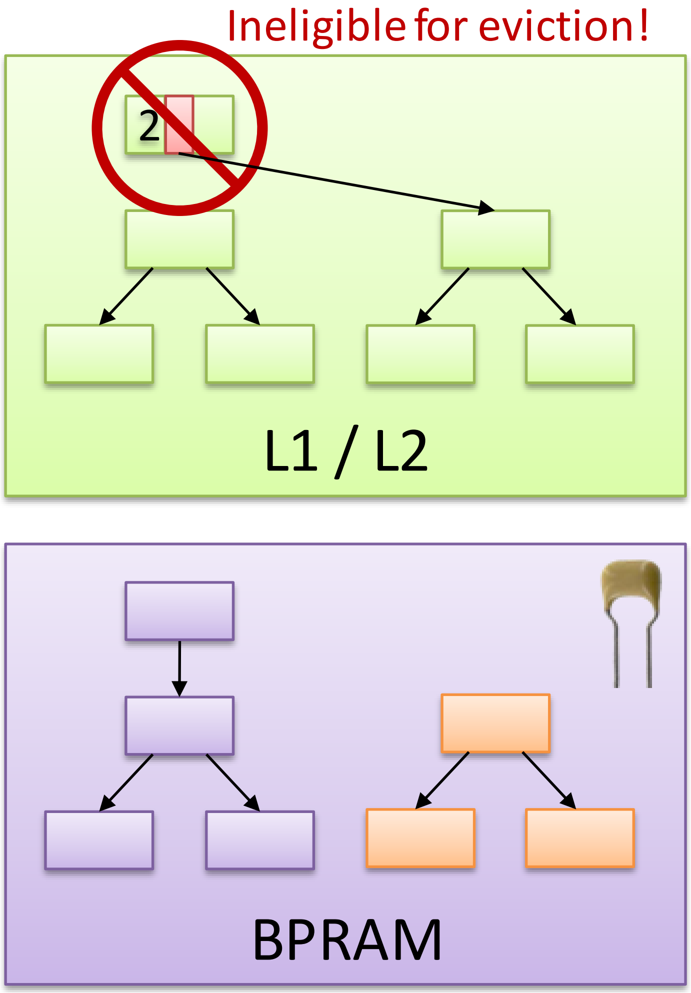
```
...
CoW
Barrier
Commit
...
```



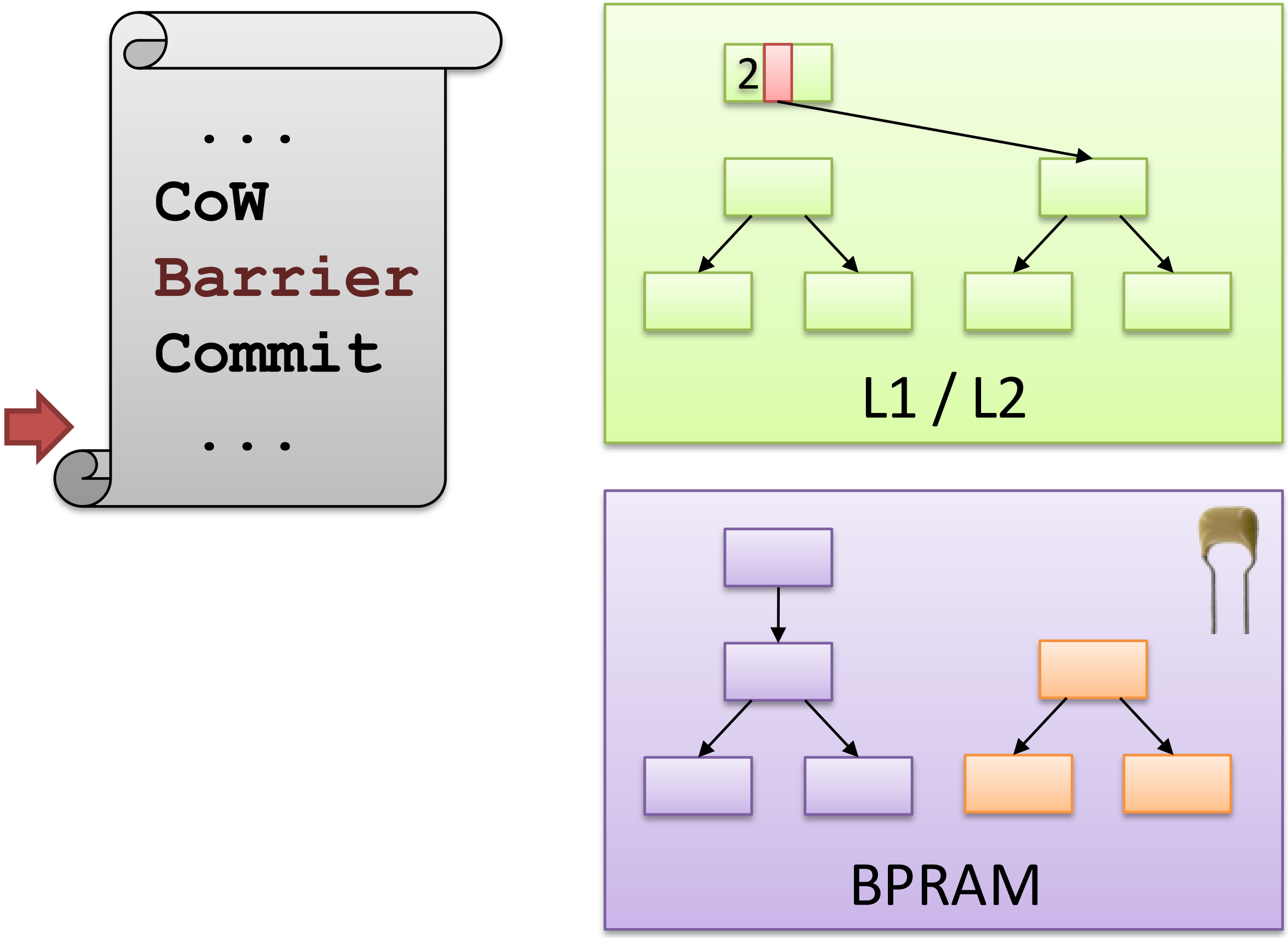
Ineligible for eviction!

Ordering and Atomicity

```
...
CoW
Barrier
Commit
...
```

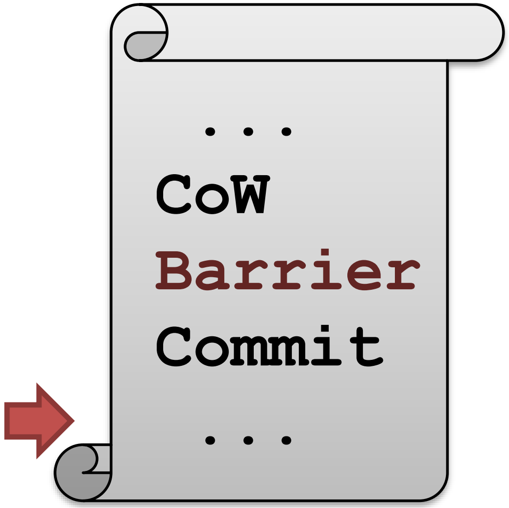


Ordering and Atomicity

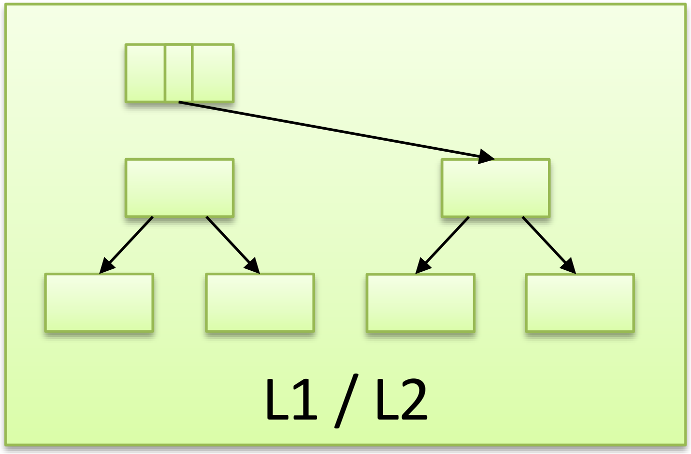


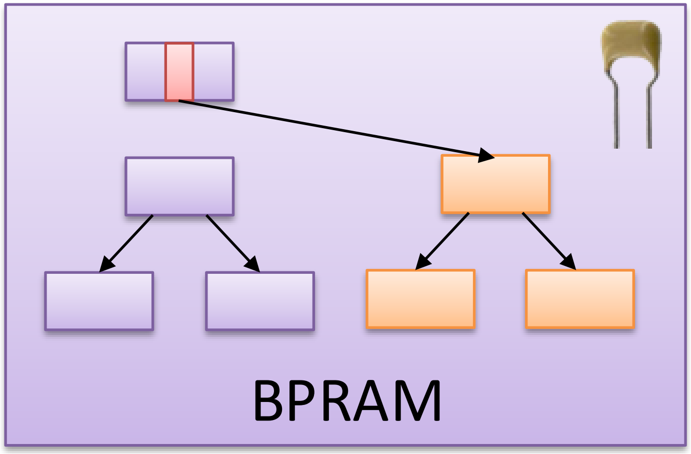
...
CoW
Barrier
Commit
...

Ordering and Atomicity

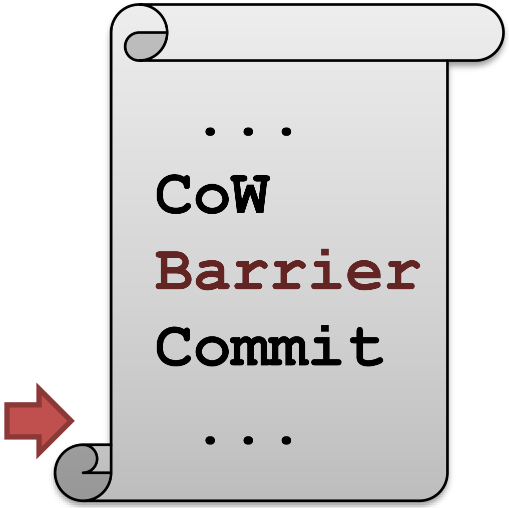


```
...
CoW
Barrier
Commit
...
```

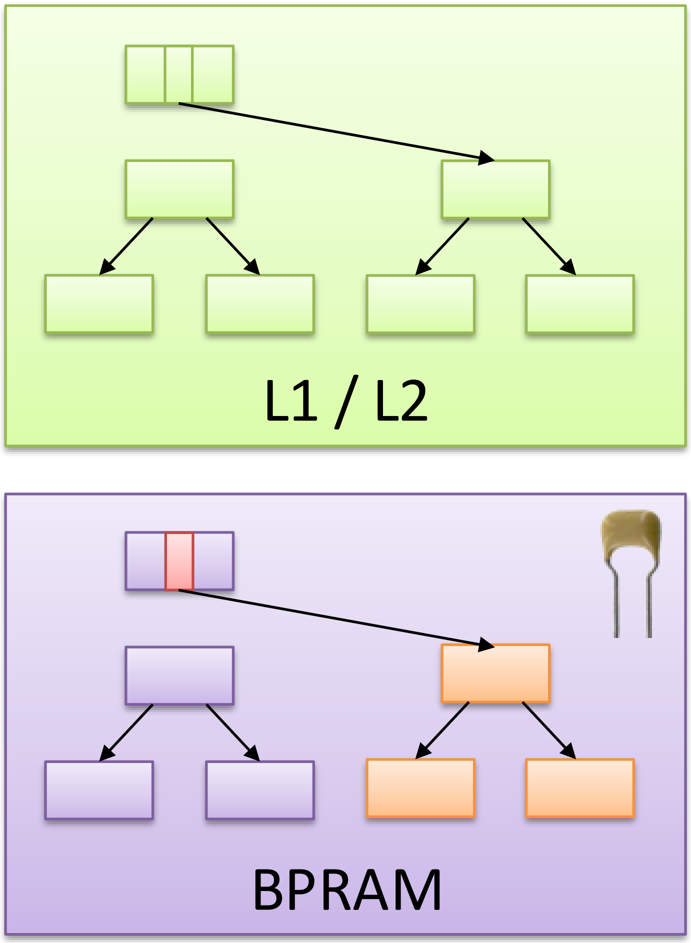




Ordering and Atomicity



```
...
CoW
Barrier
Commit
...
```



MP works too
(see paper)

Outline

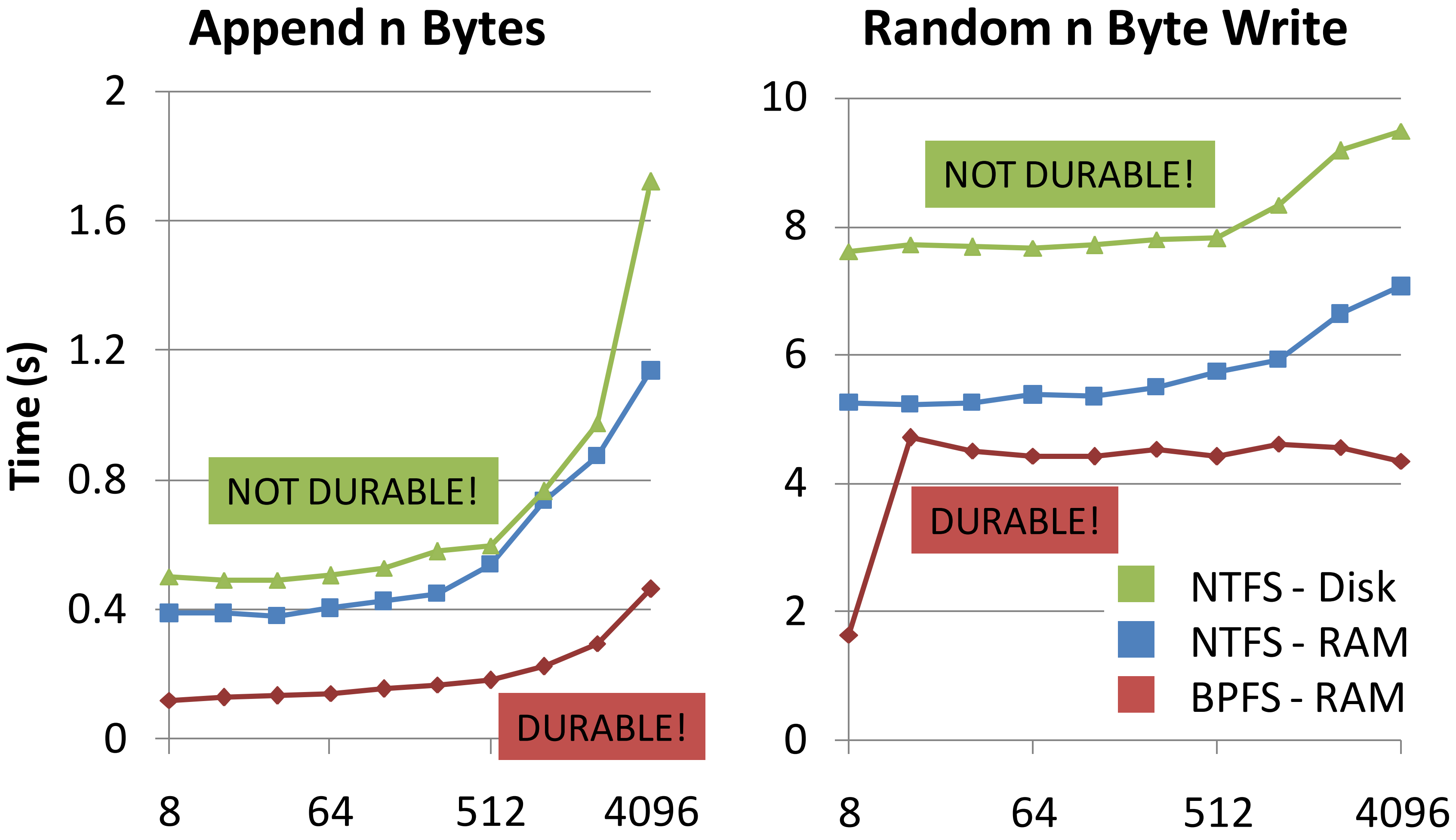
- Intro
- File System
- Hardware Support
- **Evaluation**
- Conclusion

Methodology

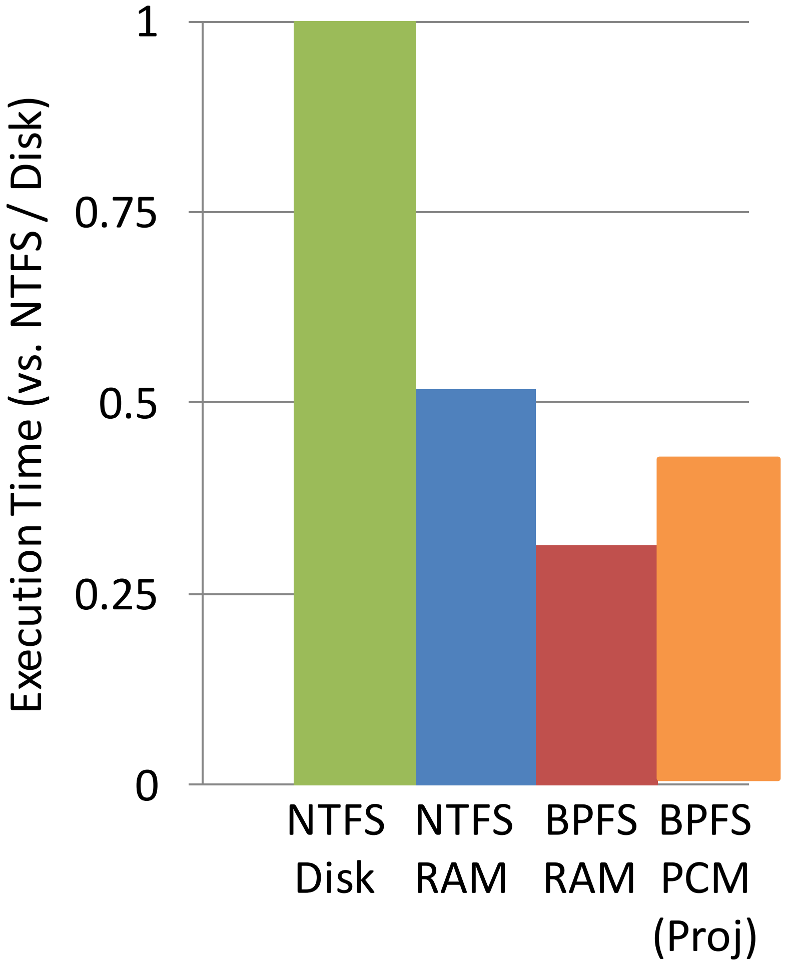
- Built and evaluated BPFS in Windows

- Three parts:
  - Experimental: BPFS vs. NTFS on DRAM
  - Simulation: Epoch barrier evaluation
  - Analytical: BPFS on PCM

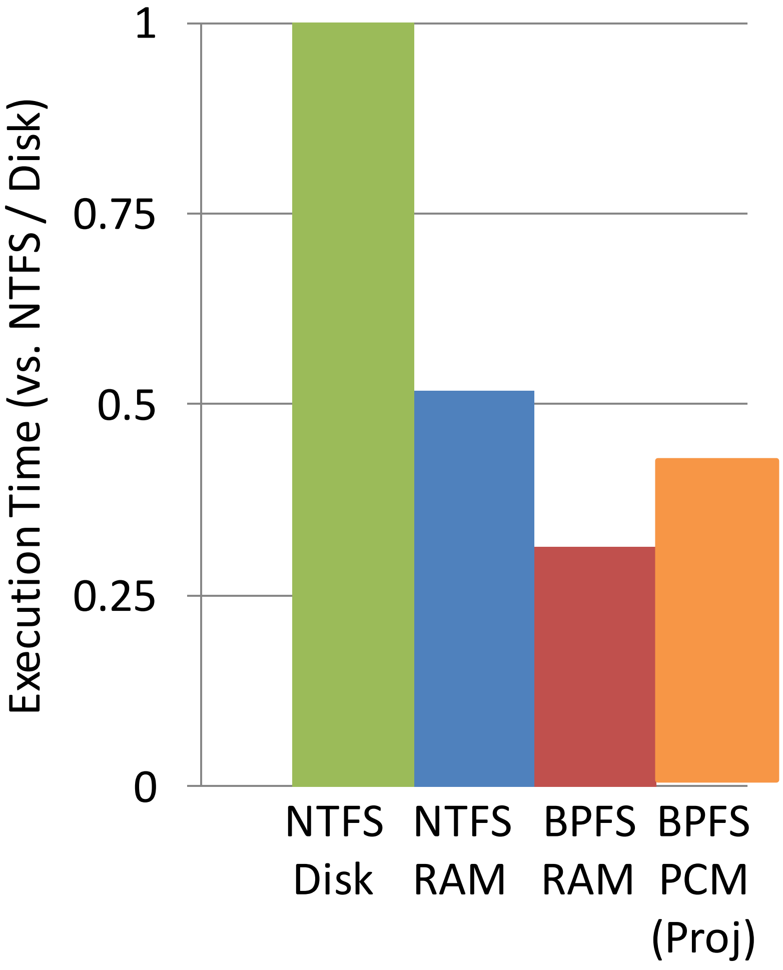
Microbenchmarks

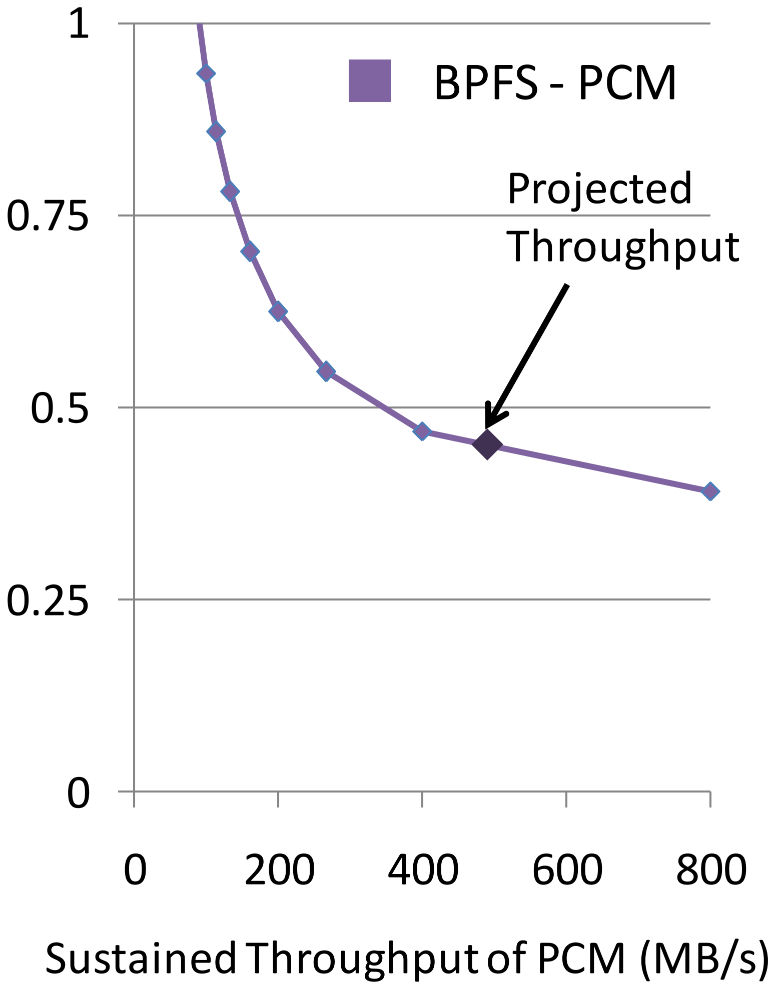


BPFS Throughput On PCM



BPFS Throughput On PCM




Projected Throughput

Conclusions

- BPRAM changes the trade-offs for storage
  - Use consistency technique designed for medium
- Short-circuit shadow paging:
  - improves performance
  - improves reliability

Bonus: PCM chips on display at poster session!