Physical Disentanglement in a Container-Based File System

Lanyue Lu, Yupu Zhang, Thanh Do, Samer Al-Kiswany
Andrea C. Arpaci-Dusseau, Remzi H. Arpaci-Dusseau
University of Wisconsin Madison

Motivation

Isolation in various subsystems

- resources: virtual machines, Linux containers
- security: BSD jail, sandbox
- reliability: address spaces

File systems lack isolation

- physical entanglement in modern file systems

Three problems due to entanglement

- global failures, slow recovery, bundled performance

Global Failures

Definition

- a failure which impacts all users of the file system or even the operating system

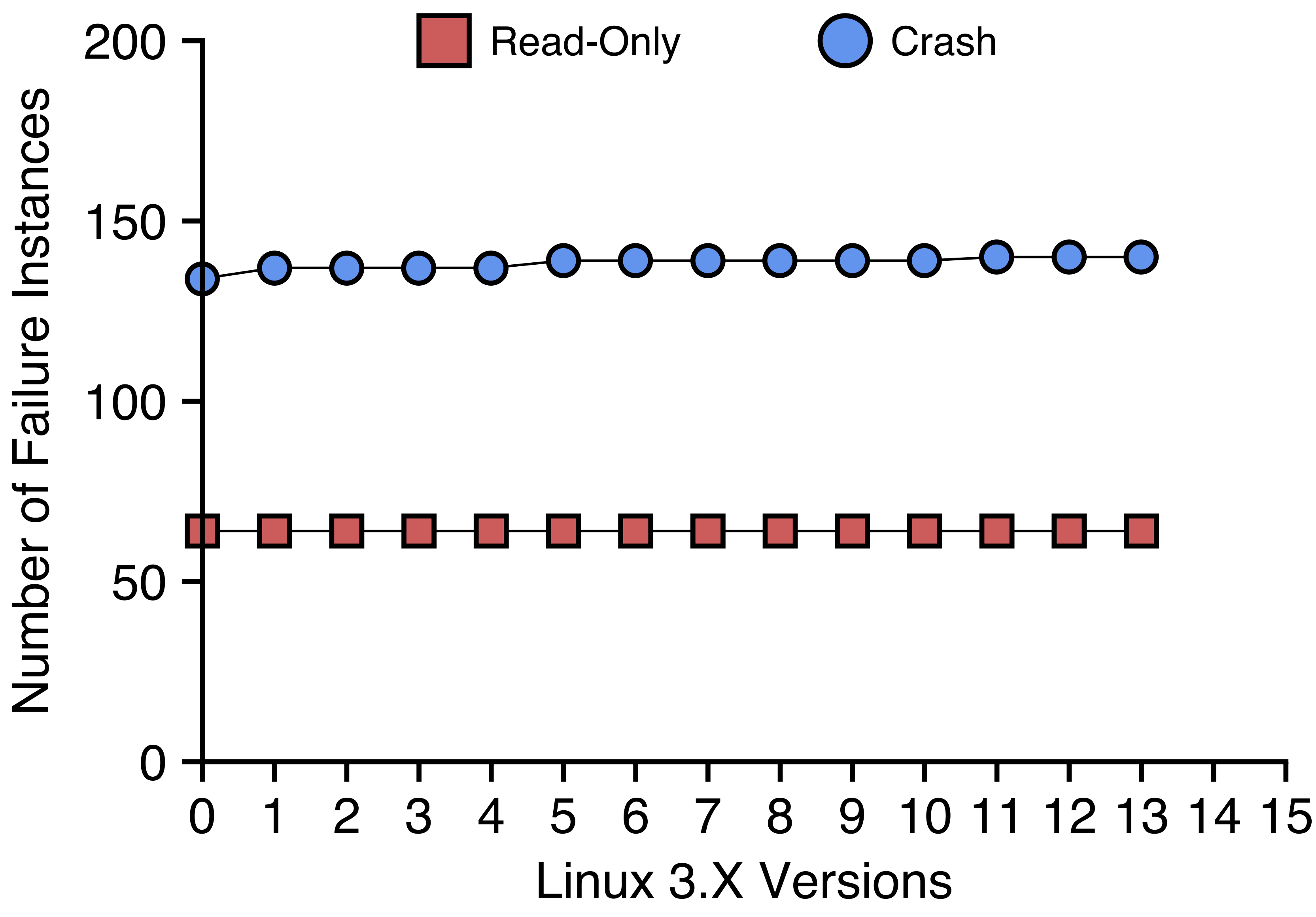
Read-Only

- mark the file system as read-only
- e.g., metadata corruption, I/O failure

Crash

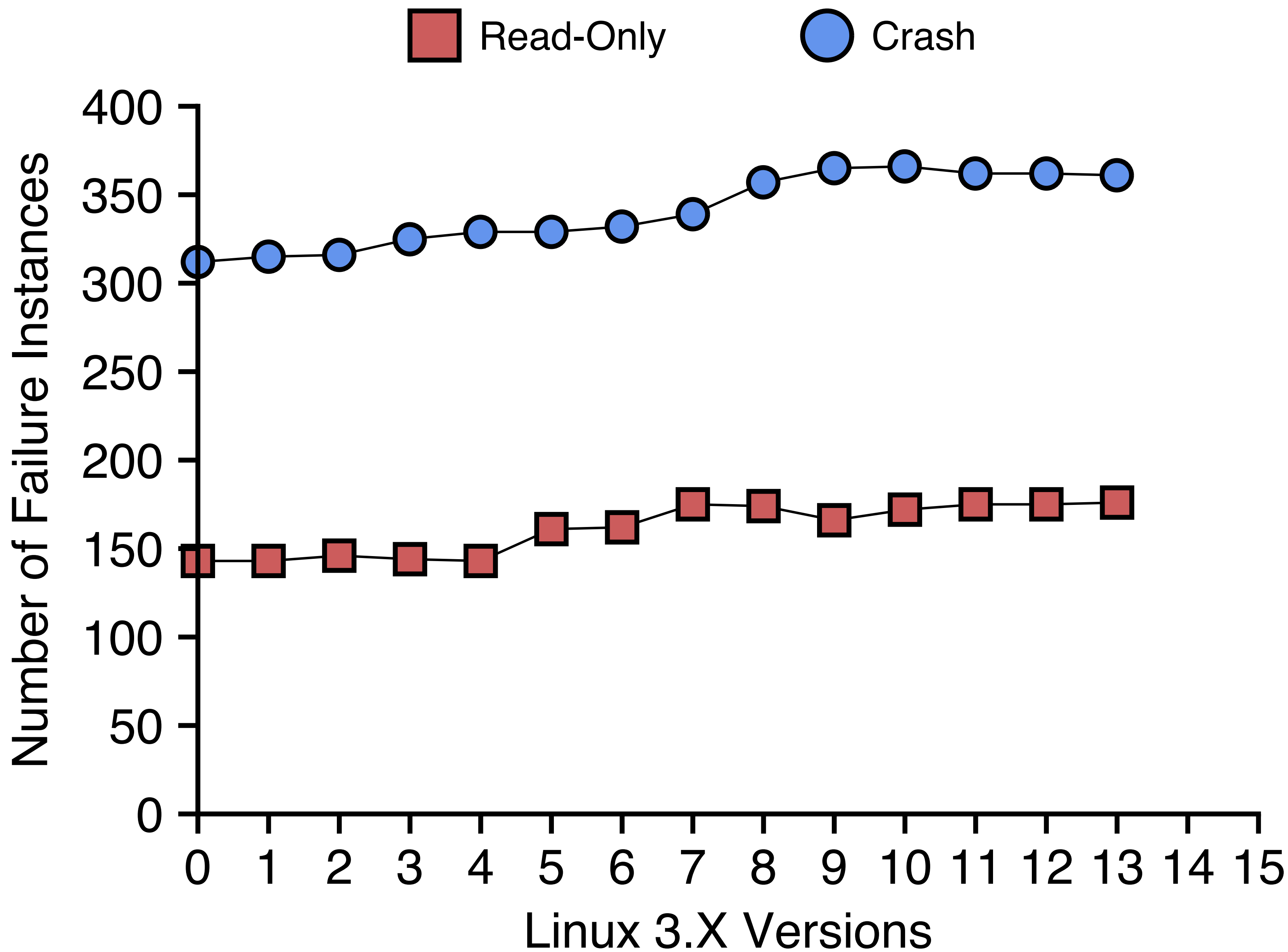
- crash the file system or the operating system
- e.g., unexpected states, pointer faults

Ext3



Read-Only
Crash
200
150
100
50
0
Number of Failure Instances
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
Linux 3.X Versions

Ext4



Read-Only
Crash
Number of Failure Instances
400
350
300
250
200
150
100
50
0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
Linux 3.X Versions

Slow Recovery

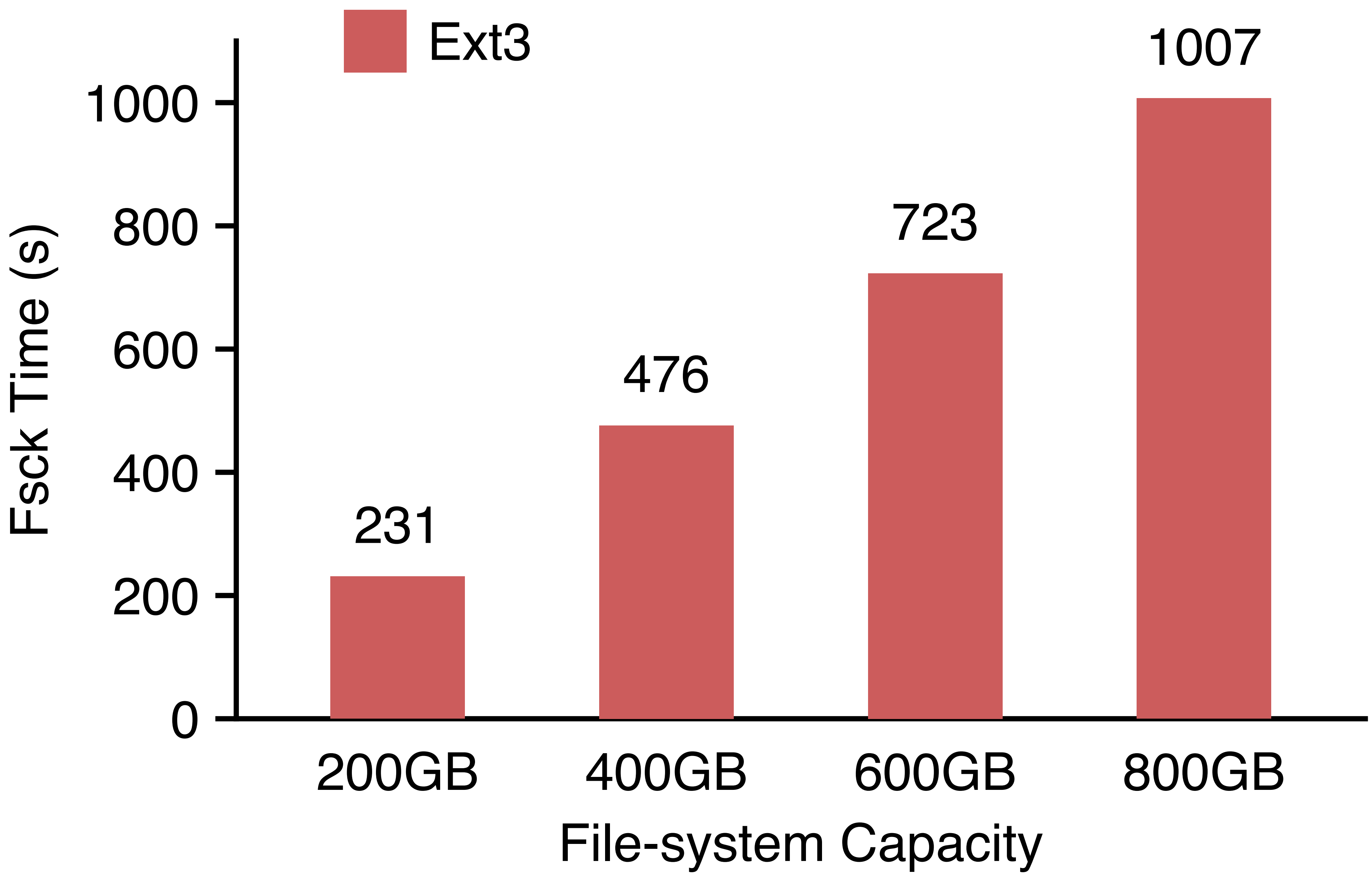
File system checker

- repair a corrupted file system
- usually offline

Current checkers are not scalable

- increasing disk capacities and file system sizes
- scan the whole file system
- checking time is unacceptably long

Scalability of fsck on Ext3



Bundled Performance

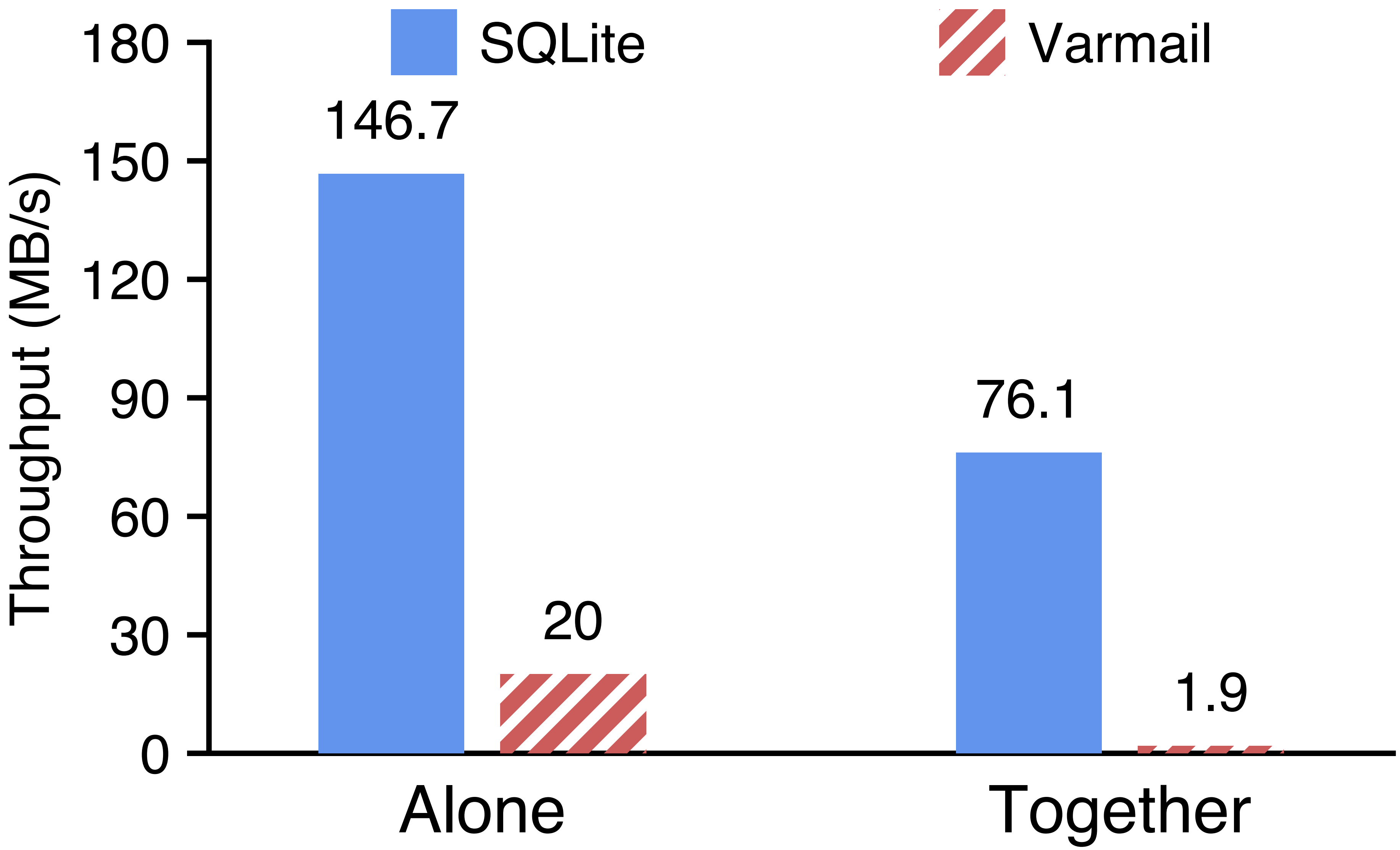
Shared transaction

- ➡ all updates share a single transaction
- ➡ unrelated workloads affect each other

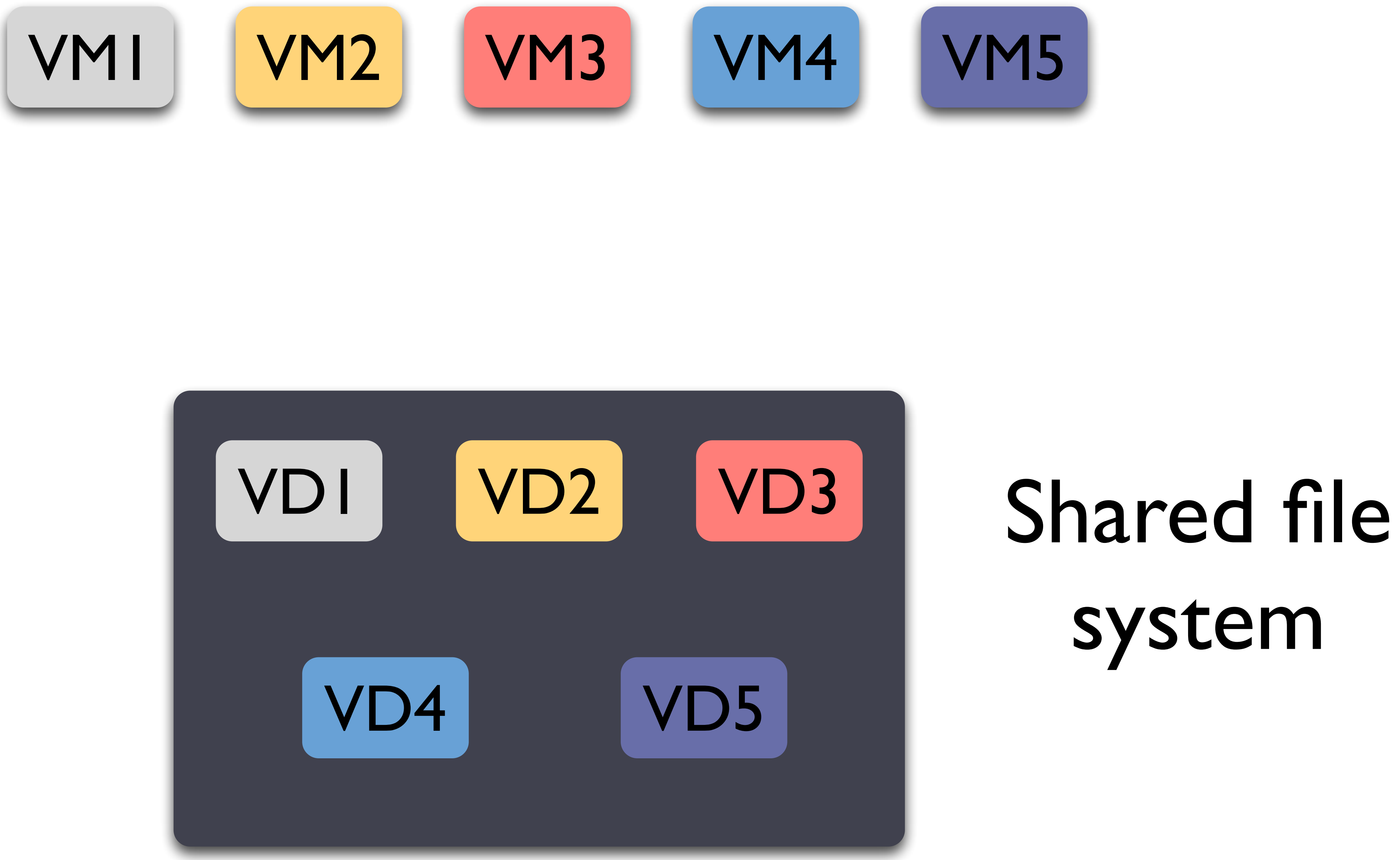
Consistency guarantee

- ➡ the same journal mode for all files
- ➡ limited flexibility for different tradeoffs

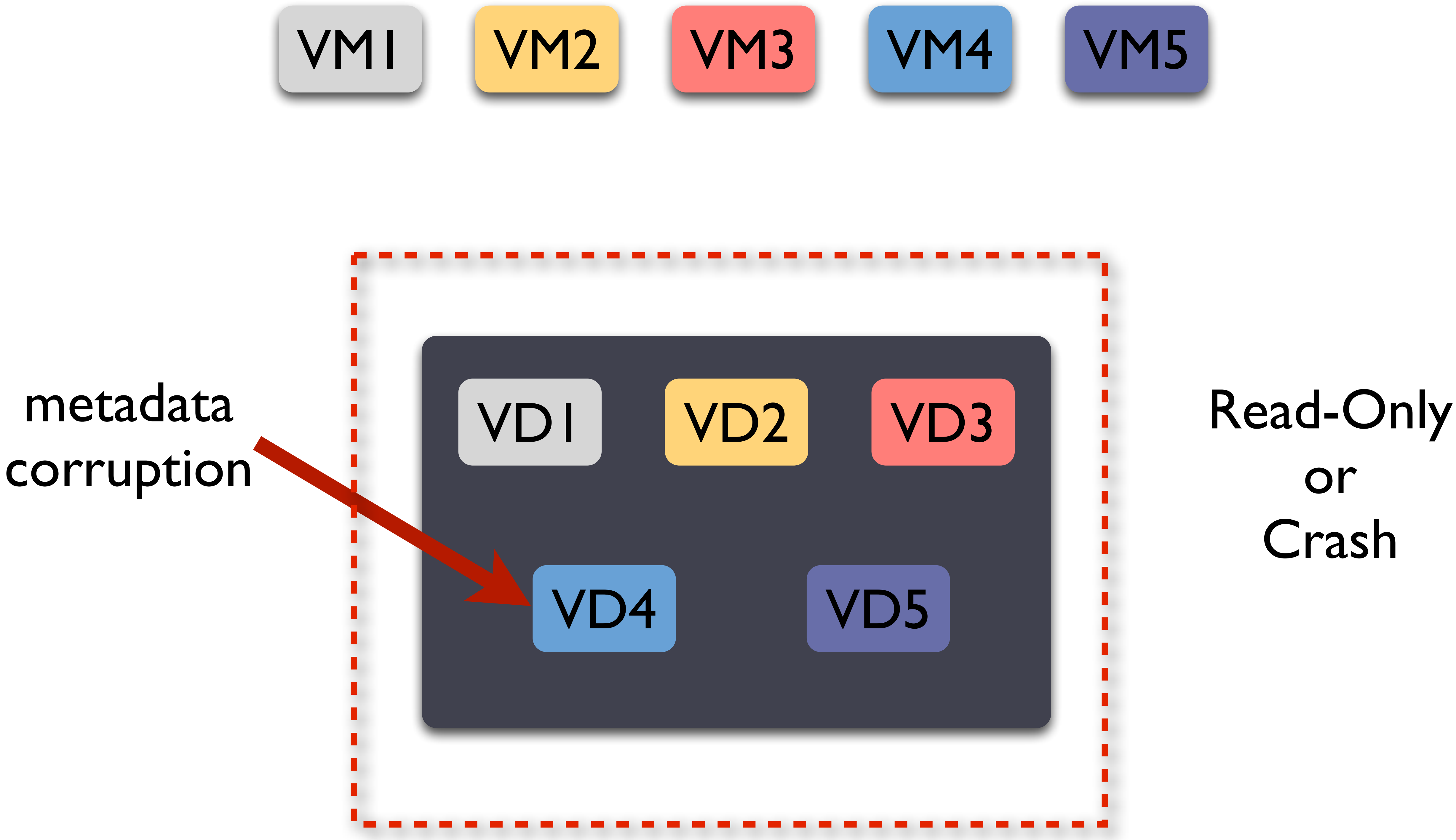
Bundle Performance on Ext3



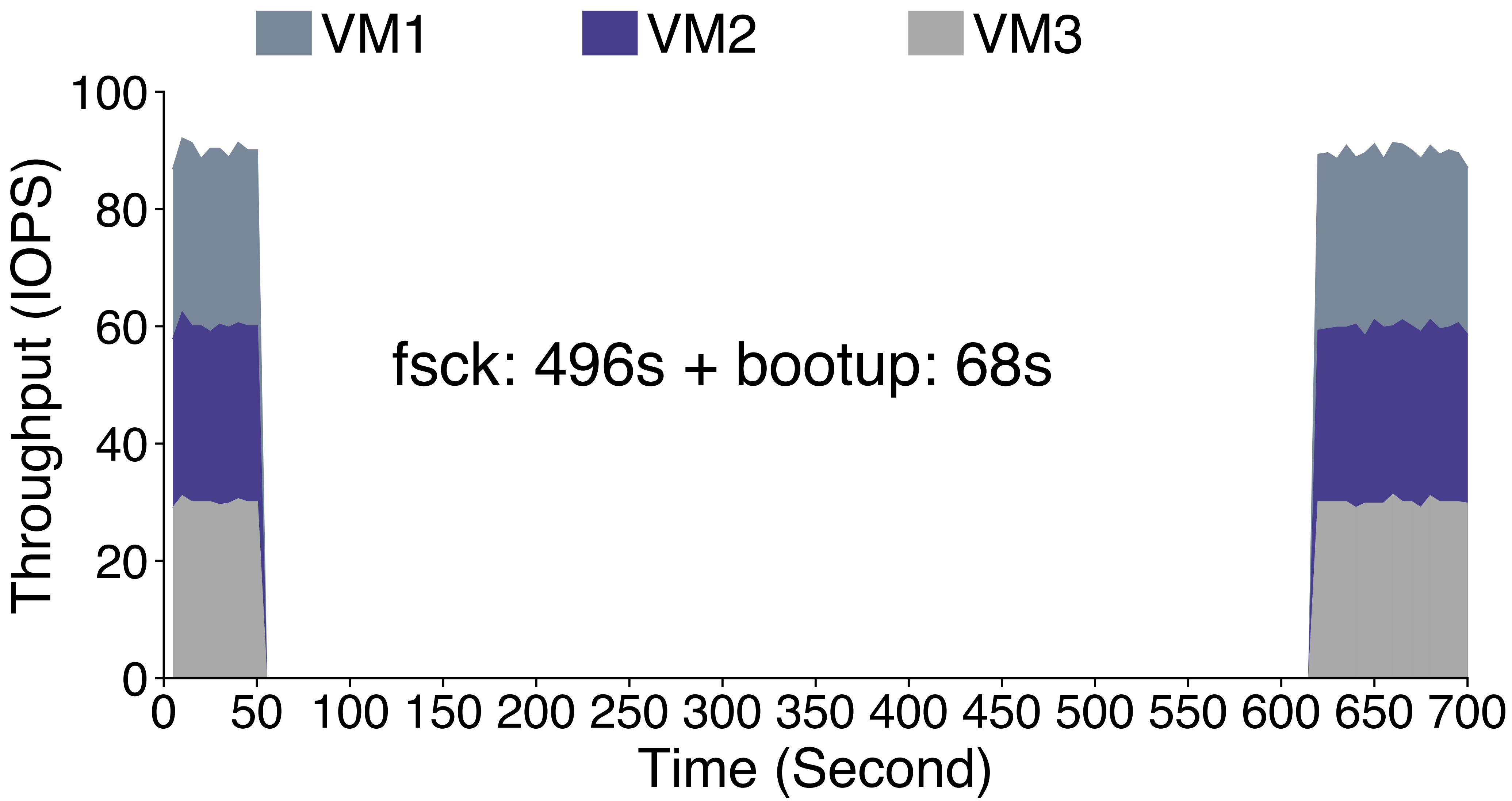
Server Virtualization



All VMs will crash



Virtual Machines



Our Solution

A new abstraction for disentanglement

A disentangled file system: IceFS

Three major benefits of IceFS

- failure isolation
- localized recovery
- specialized journaling

Motivation

IceFS

Evaluation

Conclusion

A New Abstraction: Cube

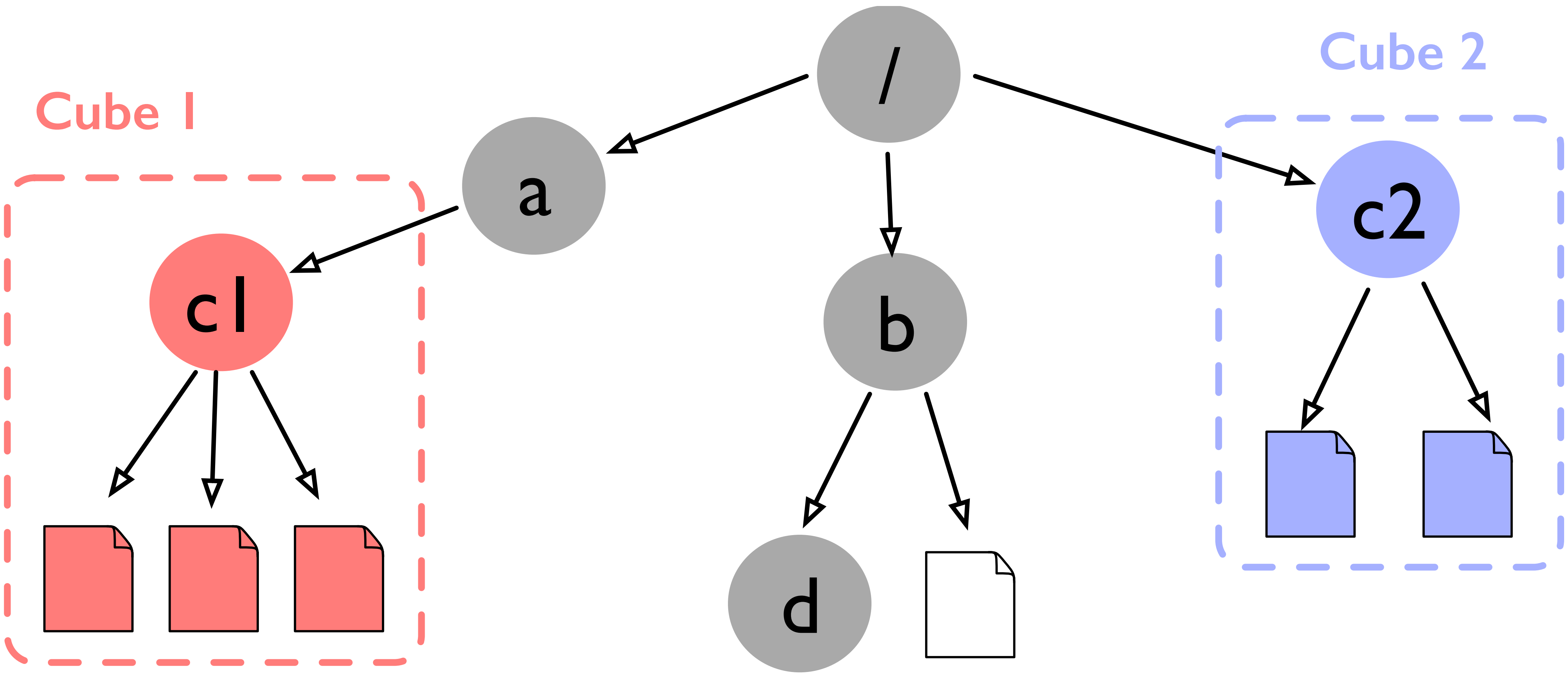
What is a cube

- an independent container for a group of files
- physically isolated

Interface

- create a cube: mkdir(cube_flag)
- delete a cube: rmdir()
- add files to a cube
- remove files to a cube
- set cube attributes

Cube Example



A Disentangled File System

No shared physical resources

- no shared metadata: e.g., block groups
- no shared disk blocks or buffers

No access dependency

- partition linked lists or trees
- avoid directory hierarchy dependency

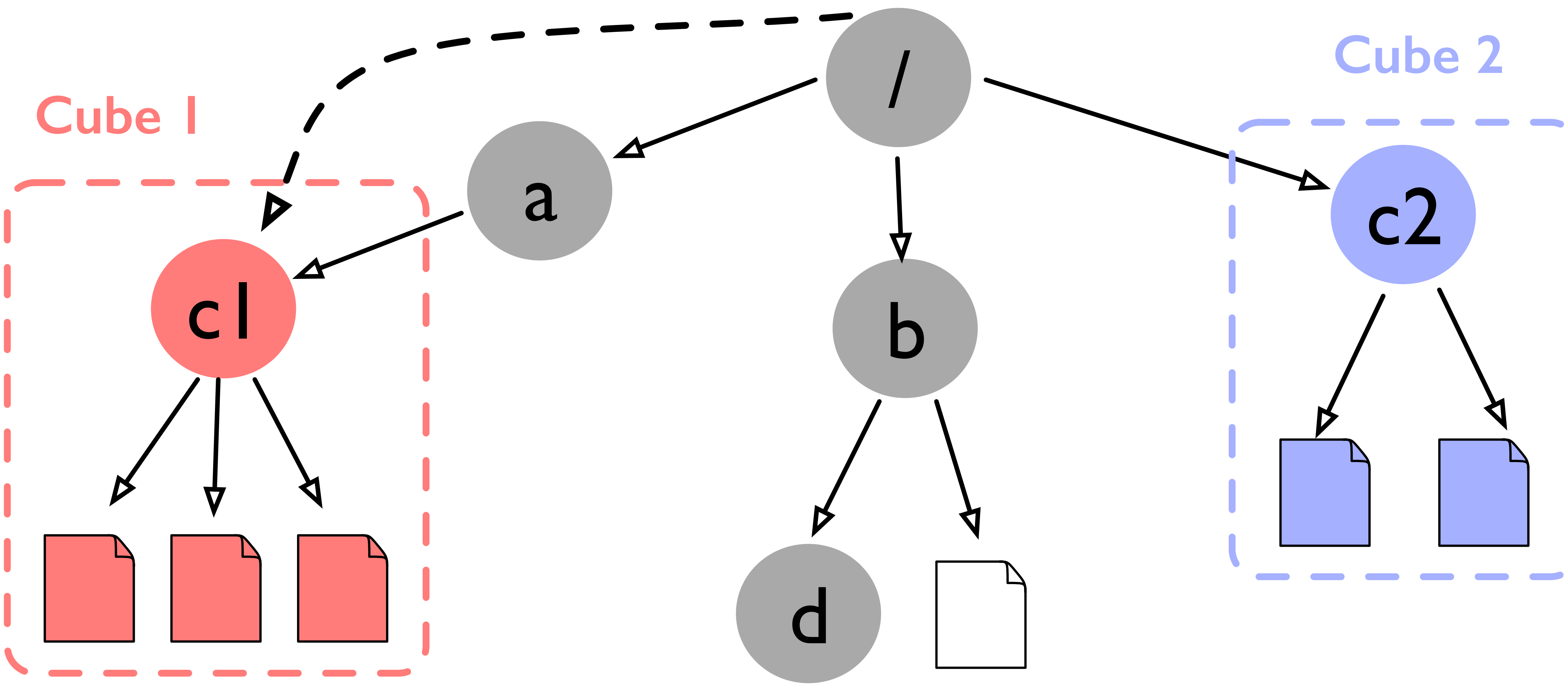
No bundled transactions

- use separate transactions
- enable customized journaling modes

IceFS

- A prototype based on Ext3 in Linux 3.5
- Disentanglement
  - ➡ directory indirection
  - ➡ transaction splitting

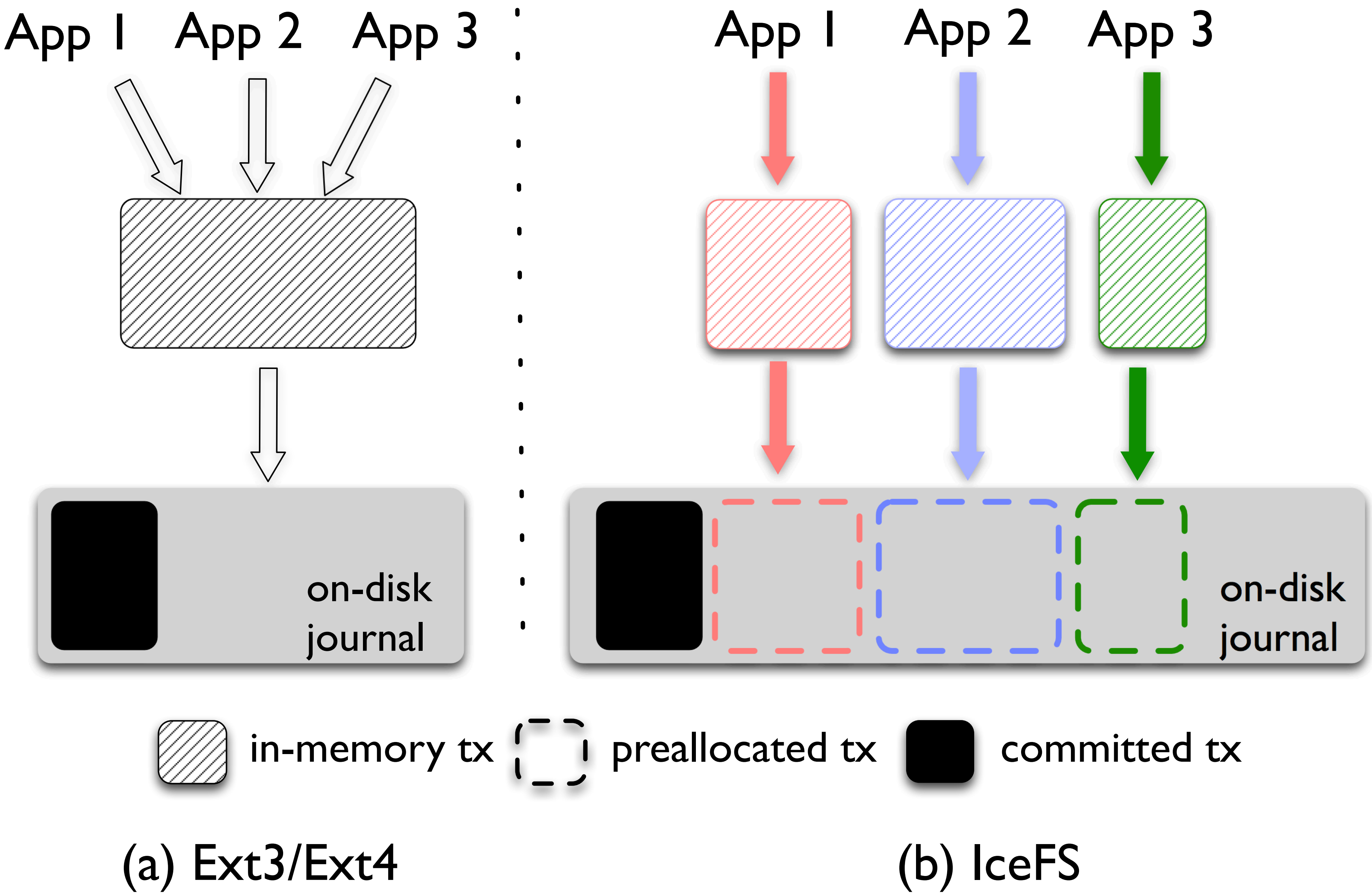
Directory Indirection



Pathname matching for cubes' top directory paths

Cubes' dentries are pinned in memory

Transaction Splitting



Separate transactions from different cubes commit to the disk journal in parallel

Benefits of Disentanglement

Localized reactions to failures

- per-cube read-only and crash

Localized recovery

- only check faulty cubes
- offline and online

Specialized journaling

- parallel journaling
- diverse journal modes
  - no journal, no fsync, writeback, ordered, data

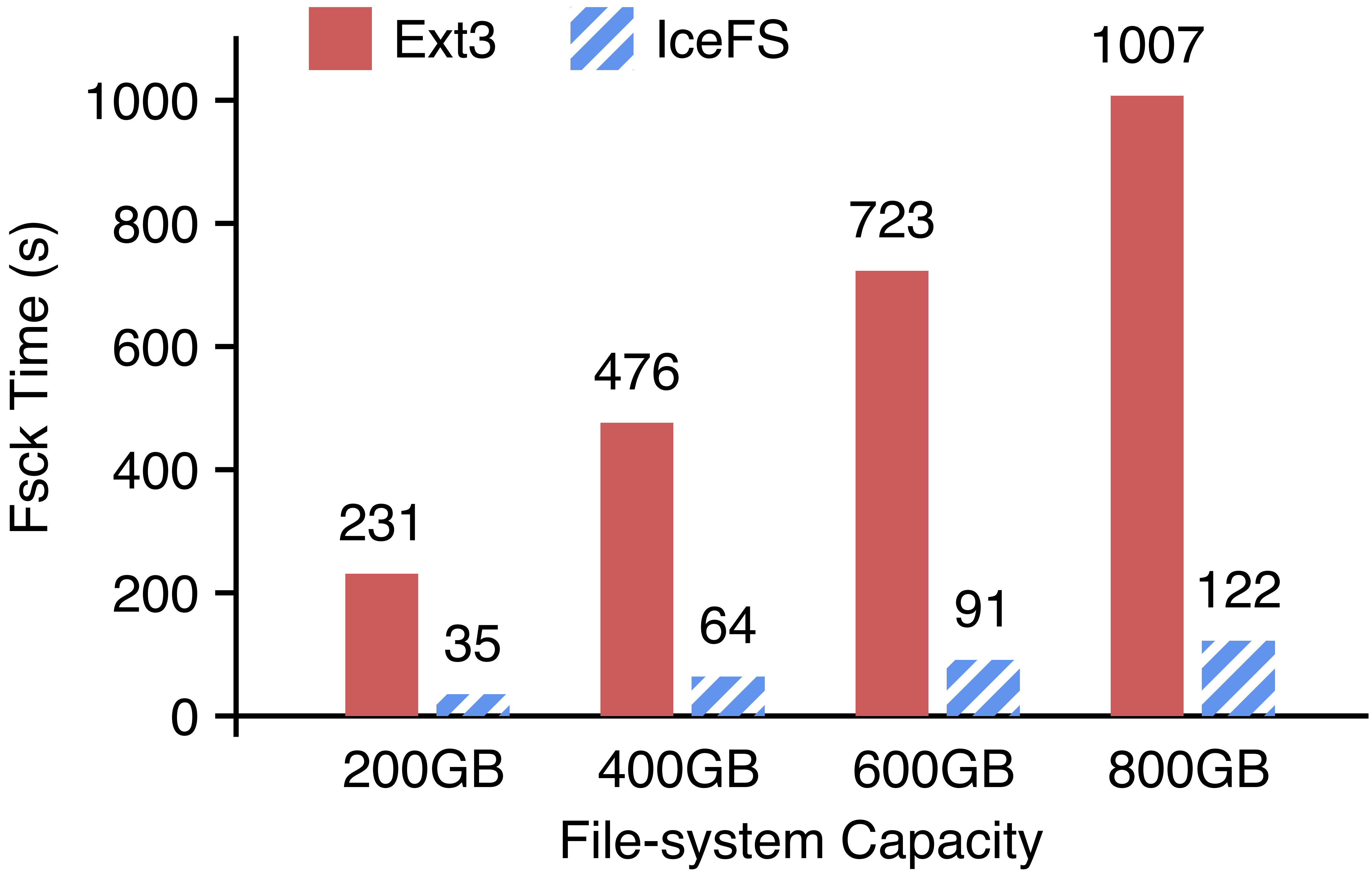
Motivation

IceFS

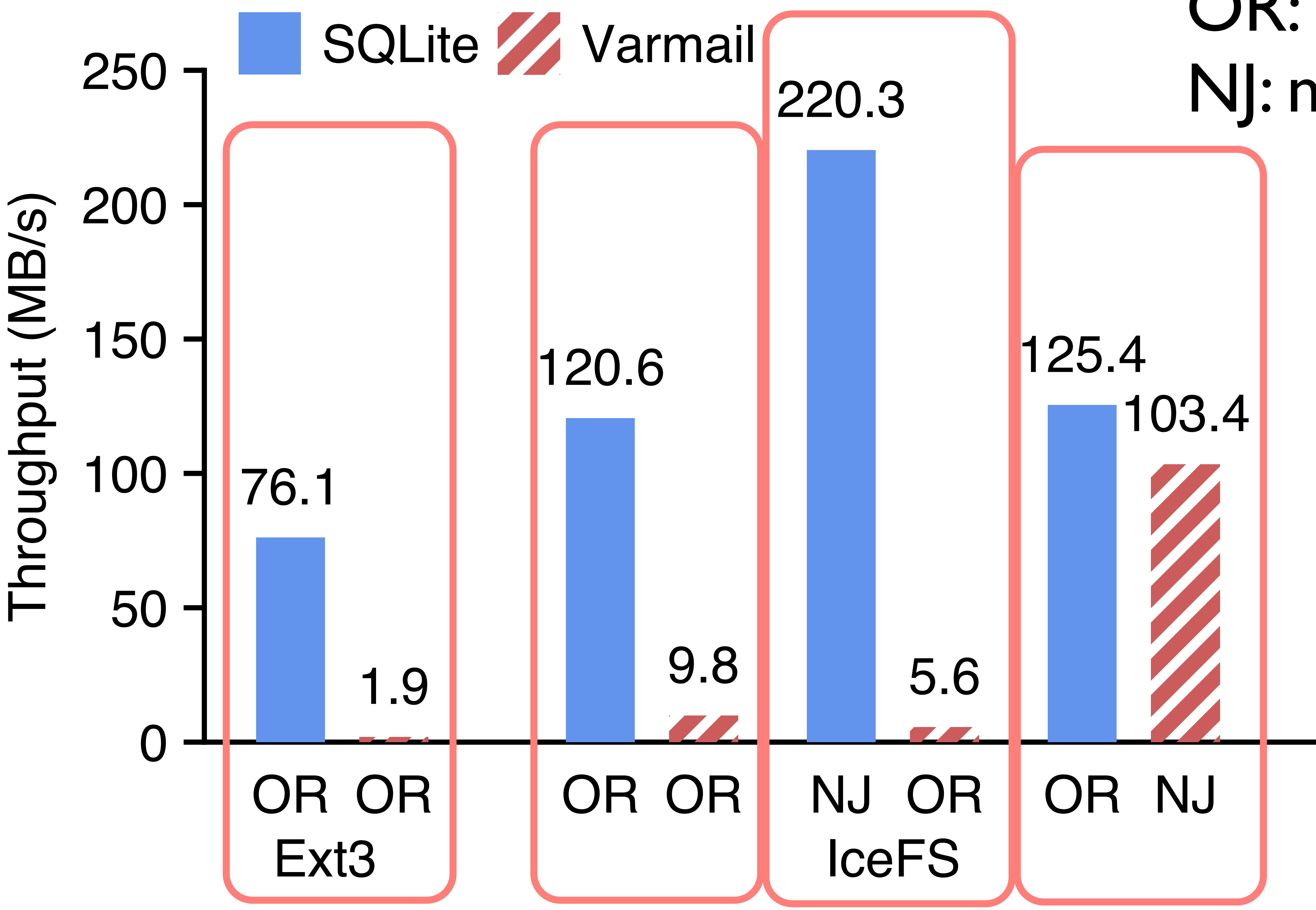
Evaluation

Conclusion

Fast Recovery

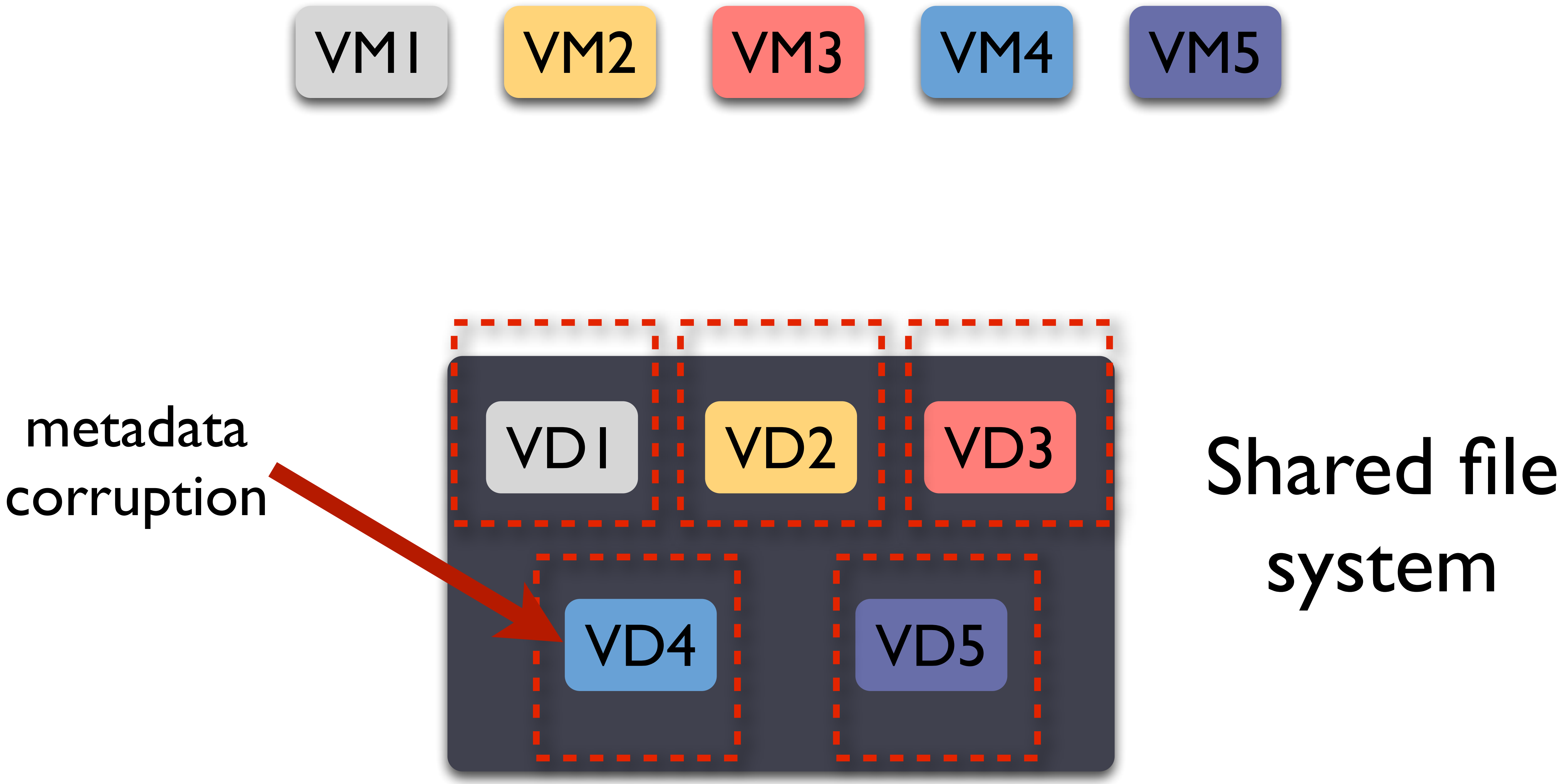


Specialized Journaling

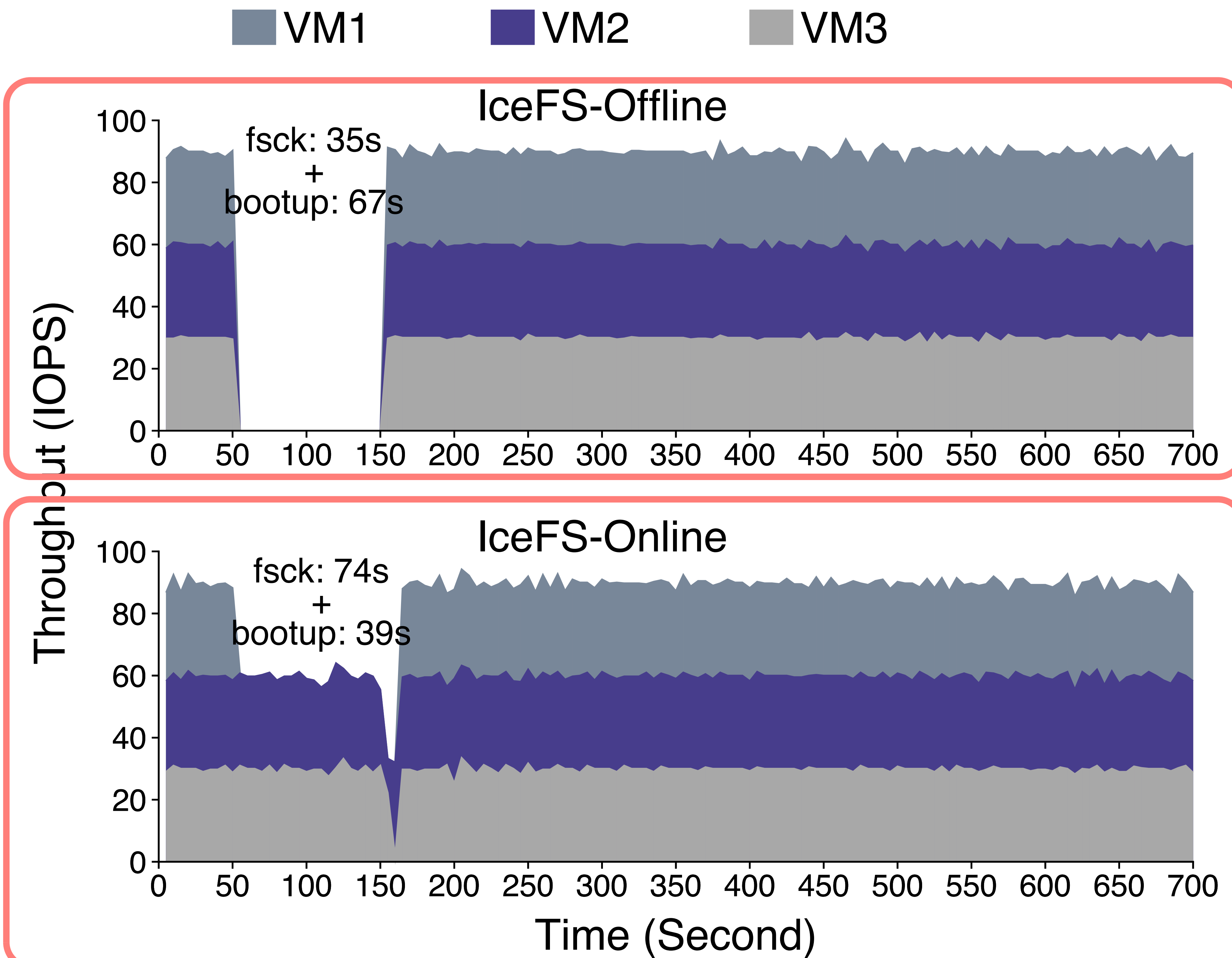


OR: ordered
NJ: no journal

Server Virtualization



Server Virtualization



Conclusion

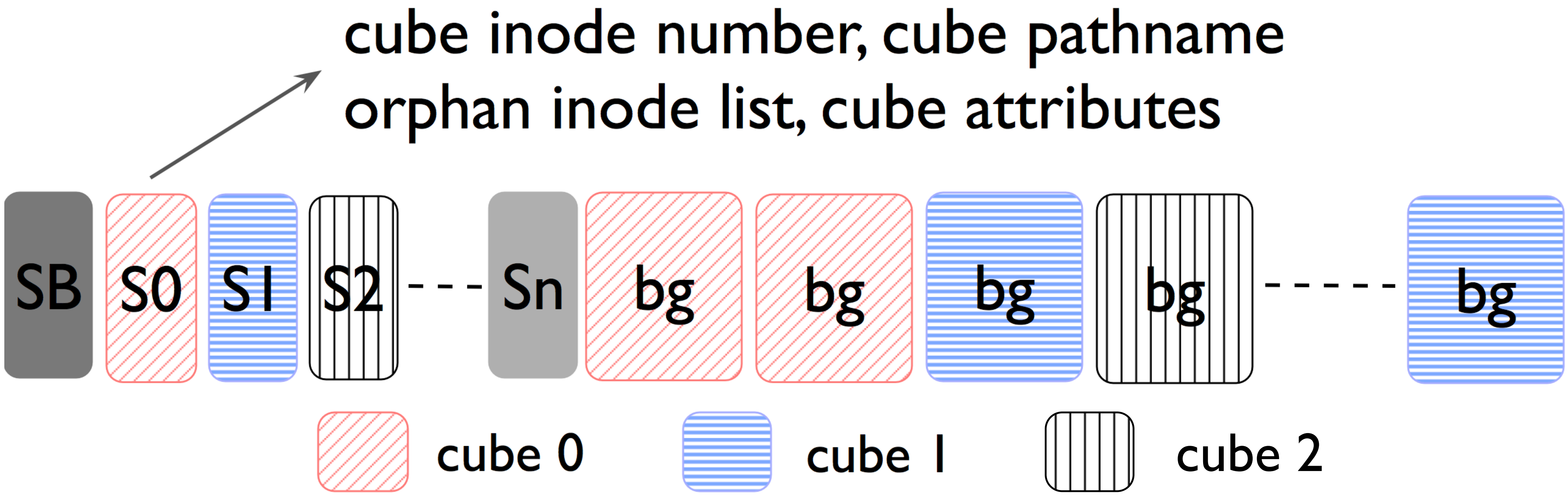
File systems lack physical isolation

Our contribution

- a new abstraction: cube
- a disentangled file system: IceFS
- demonstrate its benefits:
  - isolated failures
  - localized recovery
  - specialized journaling
  - improving both reliability and performance

Questions ?

IceFS Disk Layout



Each cube has one sub-super block (Si) and its own block groups (bg)