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

Isolation Is Important

Isolation Is Important

Reliability

- independent failures and recovery

Isolation Is Important

Reliability

- independent failures and recovery

Performance

- isolated performance

Isolation Is Important

- Reliability
  - independent failures and recovery
- Performance
  - isolated performance

Isolation at various scenarios

- computing: virtual machines, Linux containers
- security: BSD jail, sandbox
- cloud: multi-tenant systems

File Systems Lack Isolation

File Systems Lack Isolation

Local file systems are core building blocks

- manage user data
- long-standing and stable
- foundation for distributed file systems

File Systems Lack Isolation

Local file systems are core building blocks

- manage user data
- long-standing and stable
- foundation for distributed file systems

Existing abstractions provide logical isolation

- file, directory, namespace
- just illusion

File Systems Lack Isolation

Local file systems are core building blocks

- manage user data
- long-standing and stable
- foundation for distributed file systems

Existing abstractions provide logical isolation

- file, directory, namespace
- just illusion

Physical entanglement in local file systems prevents isolation

- entangled data structures and transactions

Metadata Entanglement

Metadata Entanglement

Shared metadata for multiple files

Metadata Entanglement

Shared metadata for multiple files

- e.g., multiple files share one inode block

Metadata Entanglement

Shared metadata for multiple files

- e.g., multiple files share one inode block



one 4KB inode block

Metadata Entanglement

Shared metadata for multiple files

- e.g., multiple files share one inode block

foo.txt



one 4KB inode block

Metadata Entanglement

Shared metadata for multiple files

- e.g., multiple files share one inode block

foo.txt bar.c



one 4KB inode block

Metadata Entanglement

Shared metadata for multiple files

- e.g., multiple files share one inode block

foo.txt bar.c



one 4KB inode block

I/O failure
Metadata corruption

Metadata Entanglement

Shared metadata for multiple files

- e.g., multiple files share one inode block
- many shared structures: bitmap, directory block

foo.txt bar.c



one 4KB inode block

I/O failure
Metadata corruption

Metadata Entanglement

Shared metadata for multiple files

- e.g., multiple files share one inode block
- many shared structures: bitmap, directory block

foo.txt bar.c



one 4KB inode block

I/O failure
Metadata corruption

Problem: faults in shared structures lead to shared failures and recovery

Transaction Entanglement

Transaction Entanglement

A shared transaction for all updates

Transaction Entanglement

A shared transaction for all updates

foo.txt bar.c

Mem

Disk

Transaction Entanglement

A shared transaction for all updates

foo.txt bar.c

Mem



Disk

Transaction Entanglement

A shared transaction for all updates

foo.txt

bar.c

fsync(bar.c)

Mem



Disk

Transaction Entanglement

A shared transaction for all updates



Transaction Entanglement

A shared transaction for all updates

foo.txt

bar.c

fsync(bar.c)

Mem

Disk



Transaction Entanglement

A shared transaction for all updates

foo.txt

bar.c

fsync(bar.c)



Problem: shared transactions lead to entangled performance

Our Solution: IceFS

Our Solution: IceFS

Propose a data container abstraction: cube

Our Solution: IceFS

Propose a data container abstraction: cube

Disentangle data structures and transactions

Our Solution: IceFS

Propose a data container abstraction: cube

Disentangle data structures and transactions

Provide reliability and performance isolation

Our Solution: IceFS

Propose a data container abstraction: cube

Disentangle data structures and transactions

Provide reliability and performance isolation

Benefits for local file systems

- isolated failures for data containers
- up to 8x faster localized recovery
- up to 50x higher performance

Our Solution: IceFS

Propose a data container abstraction: cube

Disentangle data structures and transactions

Provide reliability and performance isolation

Benefits for local file systems

- isolated failures for data containers
- up to 8x faster localized recovery
- up to 50x higher performance

Benefits for high-level services

- virtualized systems: reduce the downtime over 5x
- HDFS: improve the recovery efficiency over 7x

Introduction

File System Disentanglement

- Data container abstraction
- Principles of disentanglement

The Ice File System

Evaluation

Conclusion

Data Container Abstraction: Cube

Data Container Abstraction: Cube

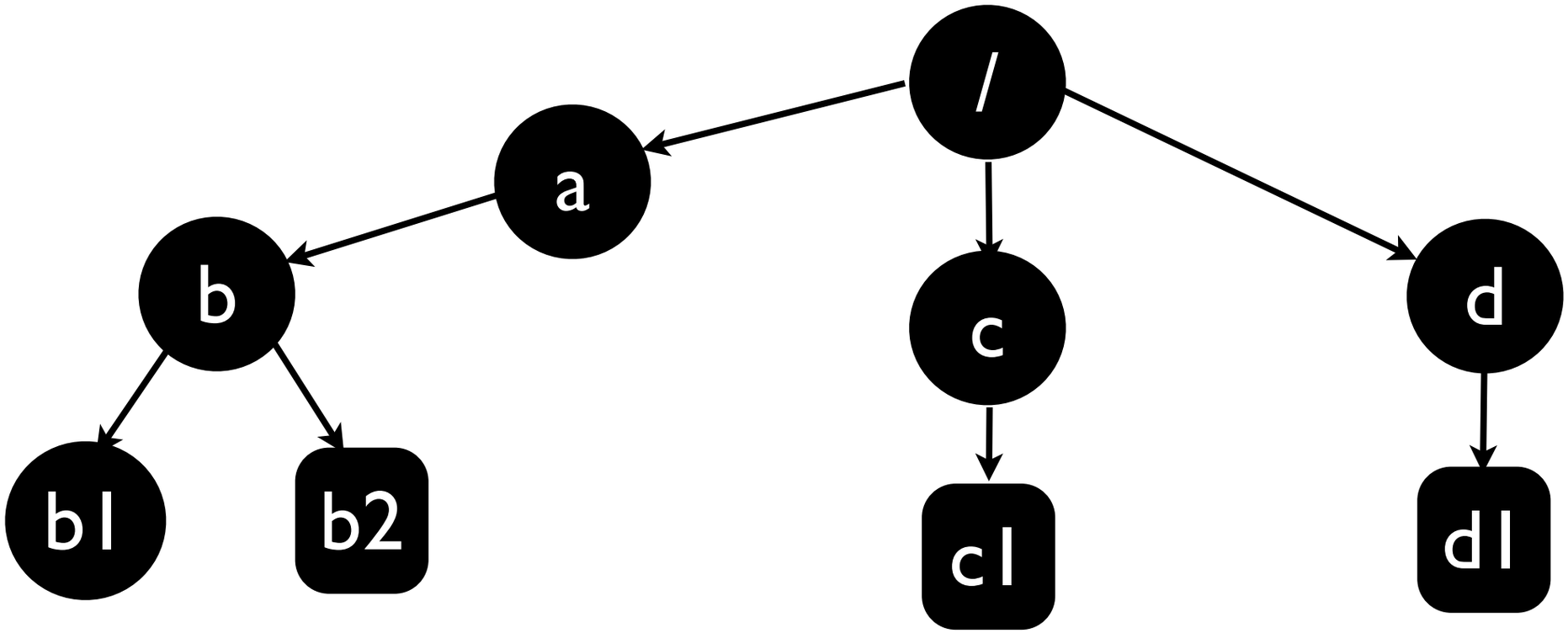
An isolated directory in a file system

- physically disentangled on disk and in memory

Data Container Abstraction: Cube

An isolated directory in a file system

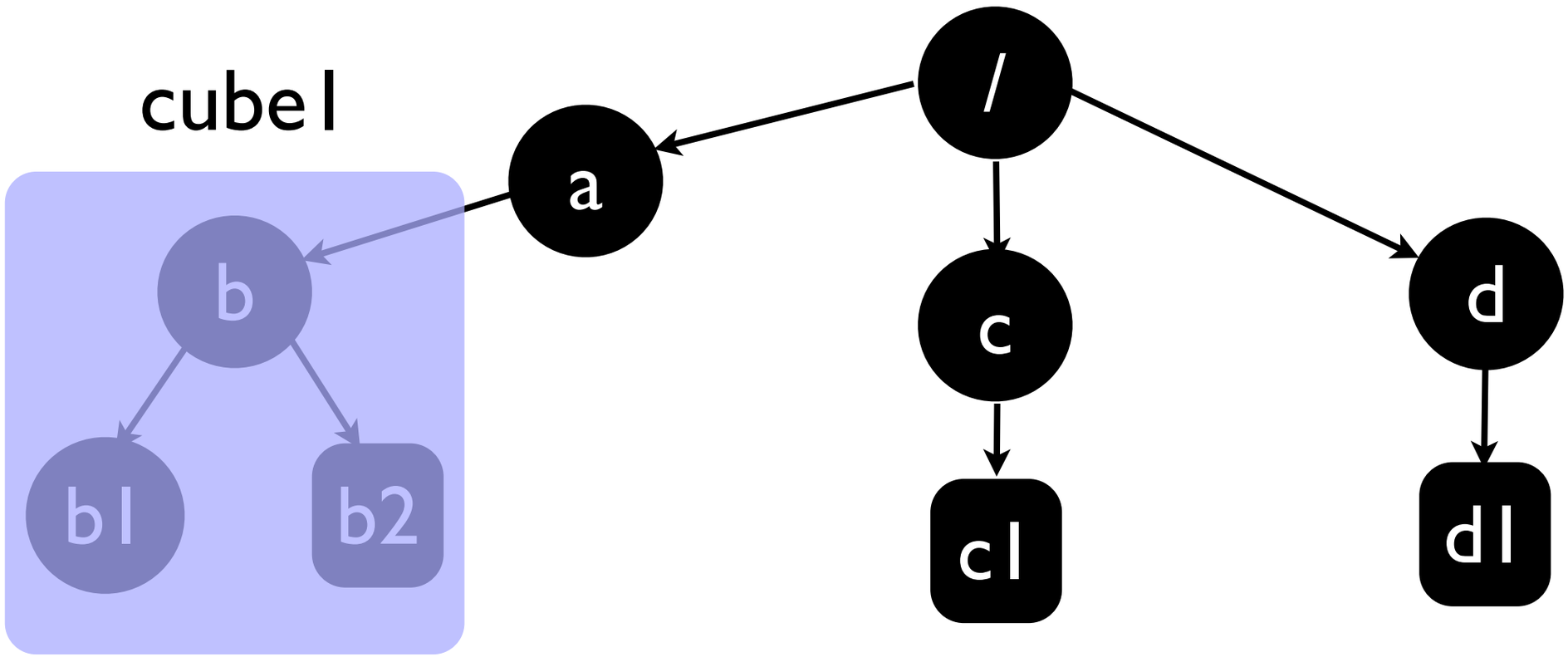
- physically disentangled on disk and in memory



Data Container Abstraction: Cube

An isolated directory in a file system

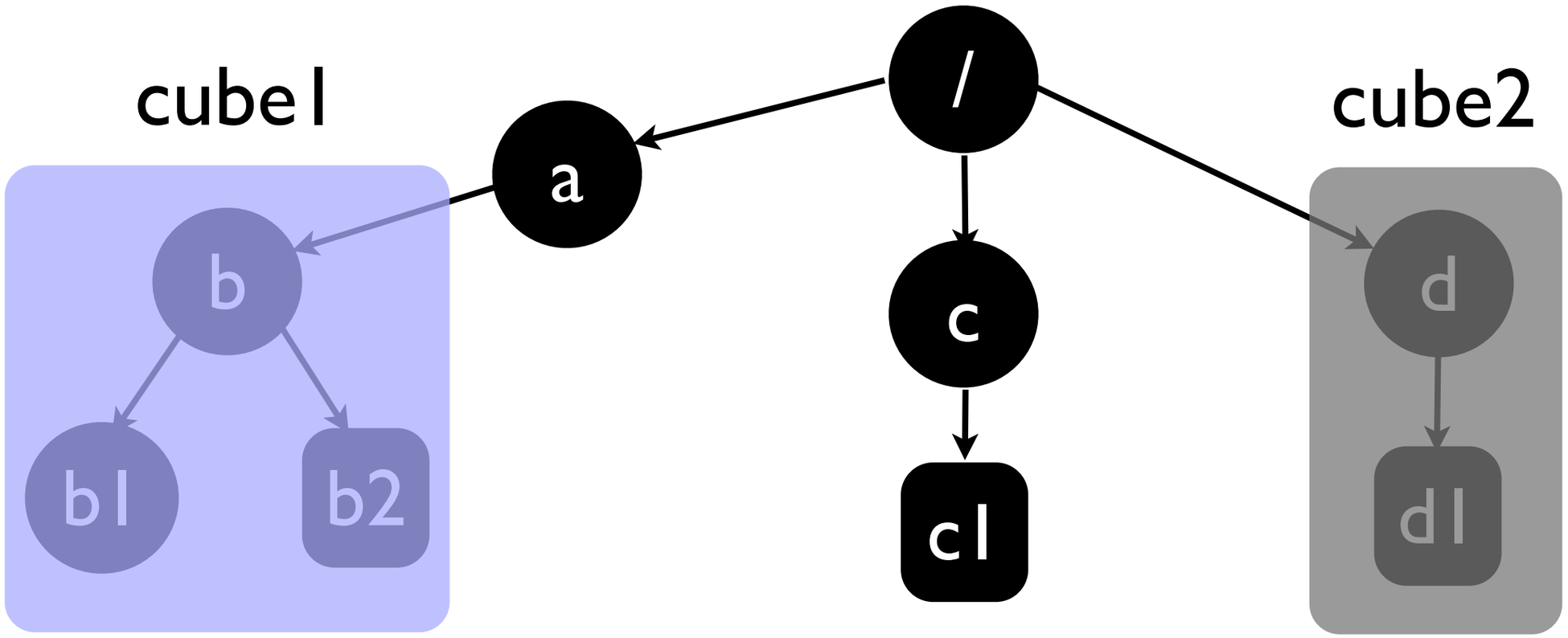
- physically disentangled on disk and in memory



Data Container Abstraction: Cube

An isolated directory in a file system

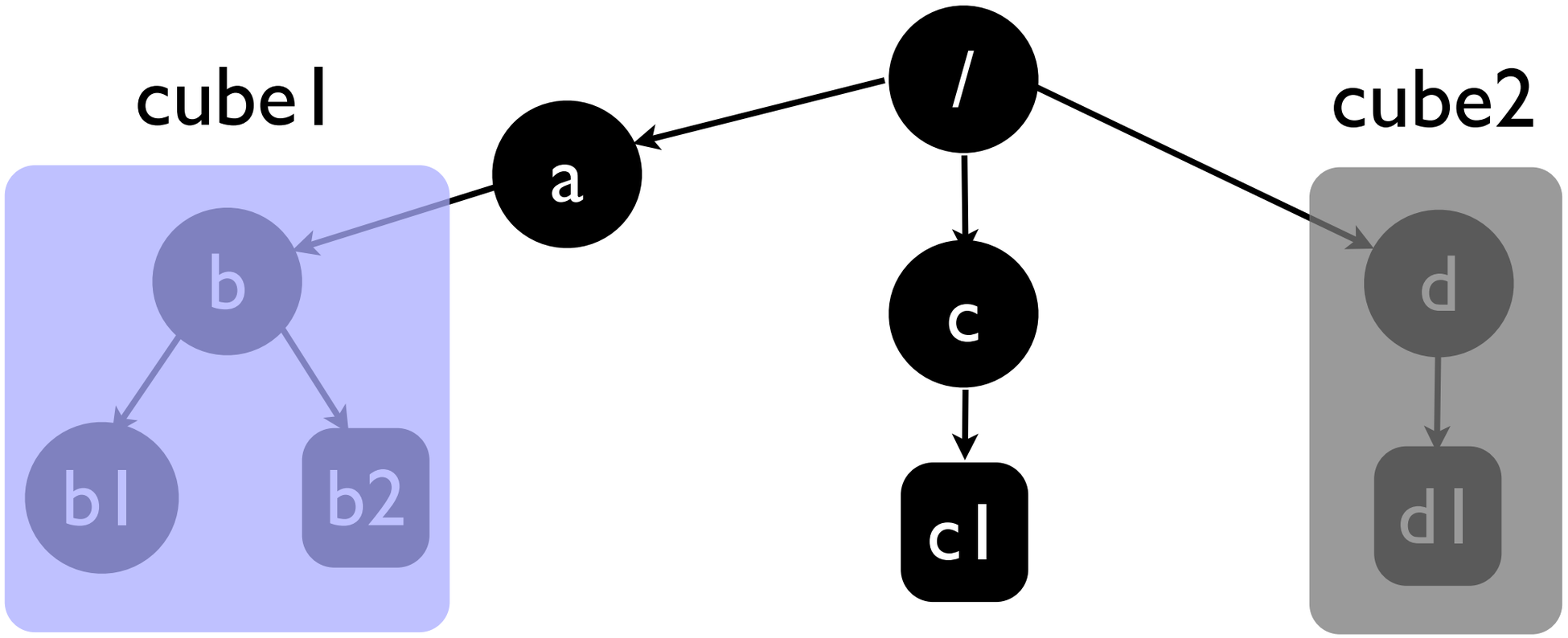
- physically disentangled on disk and in memory



Data Container Abstraction: Cube

An isolated directory in a file system

- physically disentangled on disk and in memory

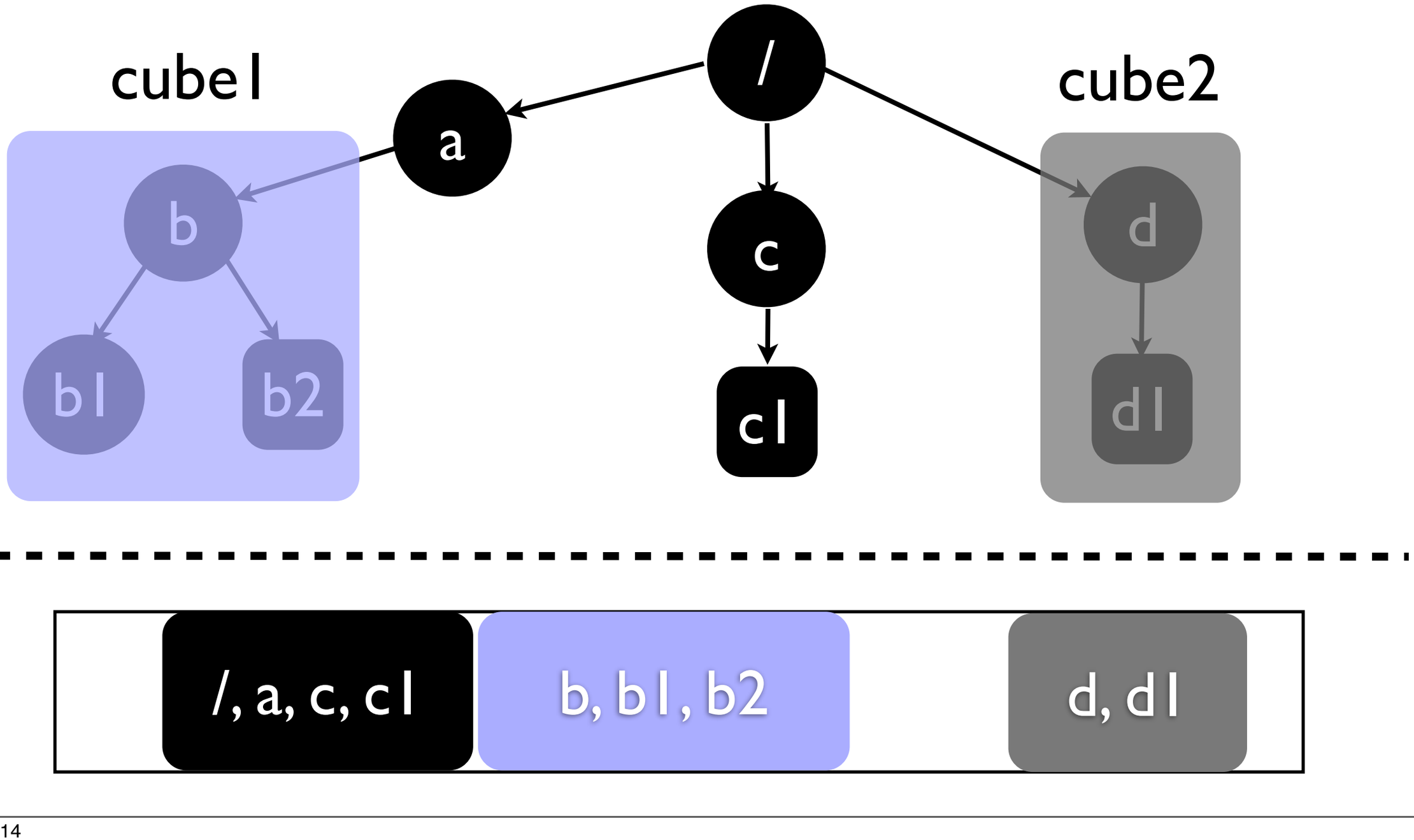


Disk

Data Container Abstraction: Cube

An isolated directory in a file system

- physically disentangled on disk and in memory



Principles of Disentanglement

Principles of Disentanglement

No shared physical resources

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

Principles of Disentanglement

No shared physical resources

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

No dependency

- partition linked lists or trees
- avoid directory hierarchy dependency

Principles of Disentanglement

No shared physical resources

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

No dependency

- partition linked lists or trees
- avoid directory hierarchy dependency

No entangled updates

- use separate transactions
- enable customized journaling modes

Introduction

File System Disentanglement

The Ice File System

- Design and implementation
- Benefits

Evaluation

Conclusion

IceFS Overview

IceFS Overview

A data container based file system

- isolated reliability and performance for containers

IceFS Overview

- A data container based file system
  - isolated reliability and performance for containers
- Disentanglement techniques
  - physical resource isolation
  - directory indirection
  - transaction splitting

IceFS Overview

- A data container based file system
  - isolated reliability and performance for containers
- Disentanglement techniques
  - physical resource isolation
  - directory indirection
  - transaction splitting
- A prototype based on Ext3
  - local file system: Ext3/JBD
  - kernel: VFS
  - user level tool: e2fsprogs

Ext3 Disk Layout

Ext3 Disk Layout

A disk is divided into block groups

- physical partition for disk locality

Ext3 Disk Layout

A disk is divided into block groups

- physical partition for disk locality



Ext3 Disk Layout

A disk is divided into block groups

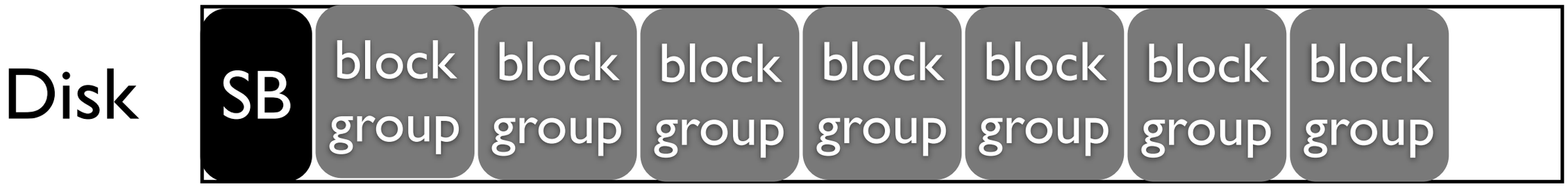
- physical partition for disk locality



Ext3 Disk Layout

A disk is divided into block groups

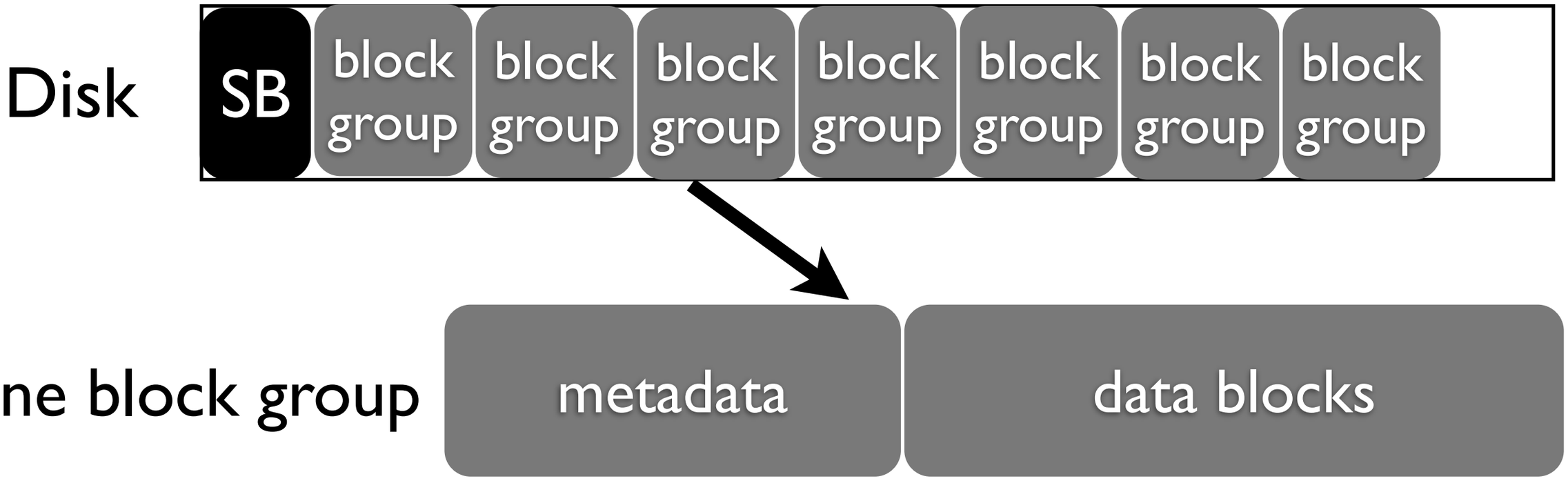
- physical partition for disk locality



Ext3 Disk Layout

A disk is divided into block groups

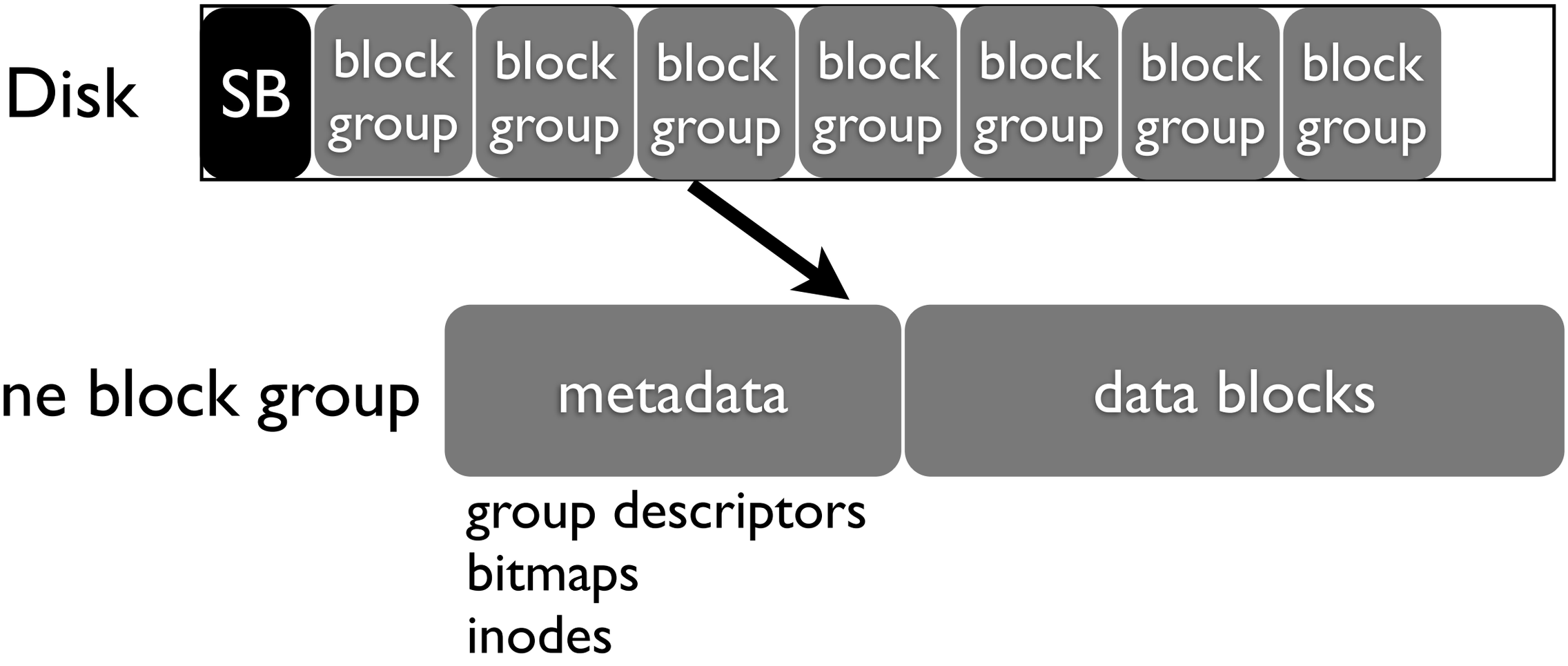
- physical partition for disk locality



Ext3 Disk Layout

A disk is divided into block groups

- physical partition for disk locality



IceFS Disk Layout

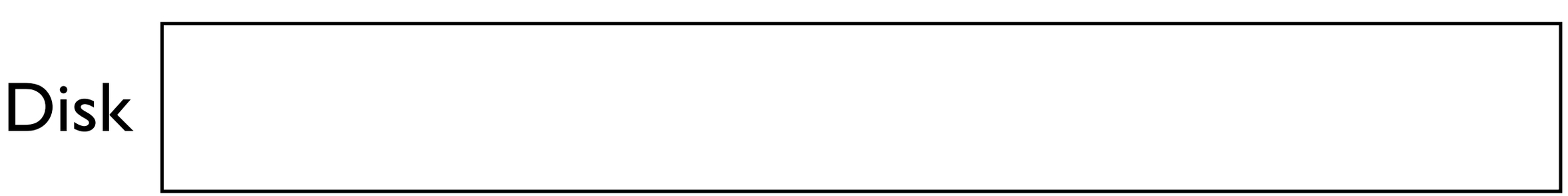
IceFS Disk Layout

- Each cube has isolated metadata
  - ➡ sub-super block (Si) and isolated block groups

IceFS Disk Layout

Each cube has isolated metadata

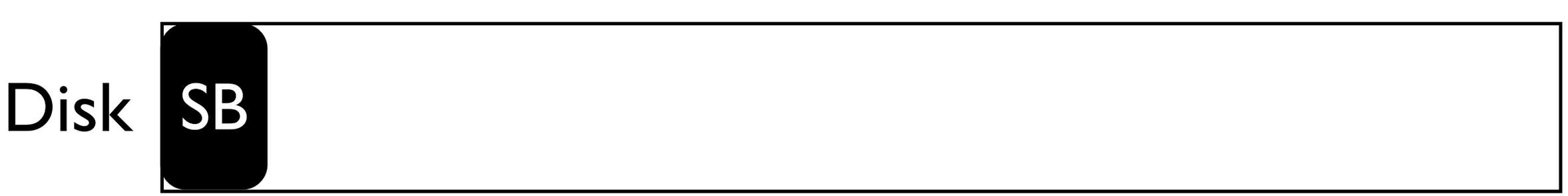
- sub-super block (Si) and isolated block groups



IceFS Disk Layout

Each cube has isolated metadata

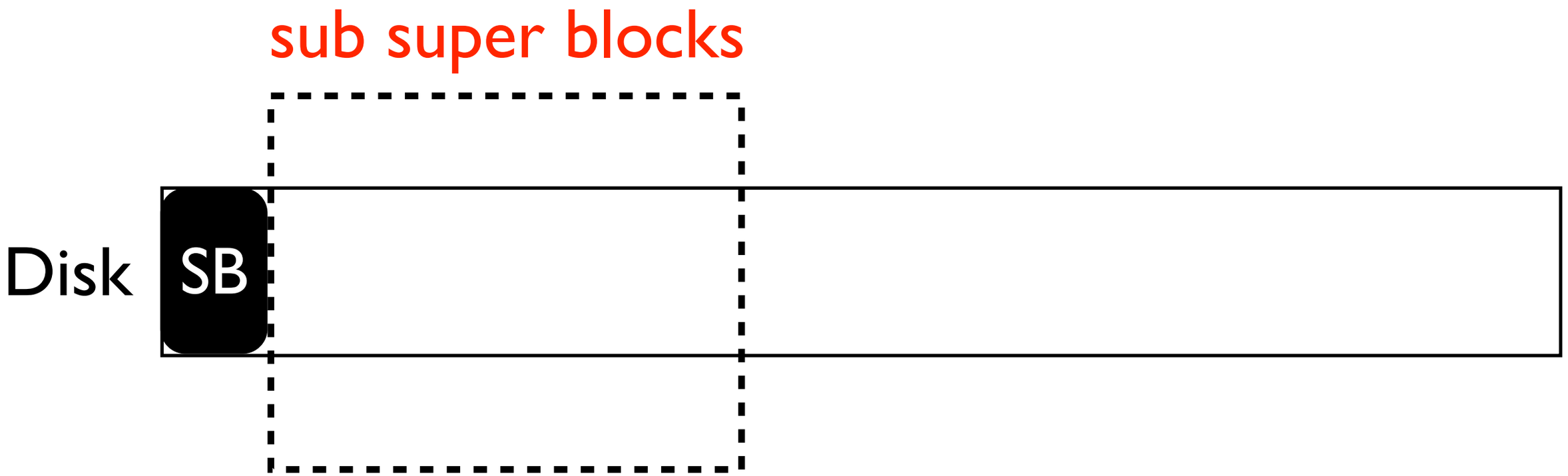
- sub-super block (Si) and isolated block groups



IceFS Disk Layout

Each cube has isolated metadata

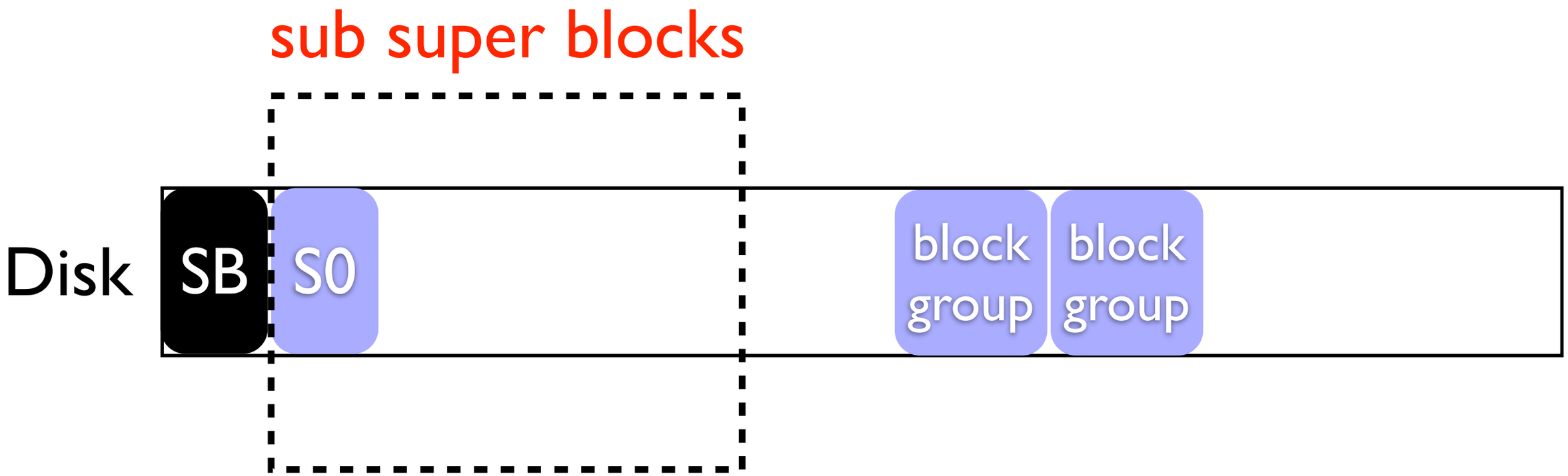
- sub-super block (Si) and isolated block groups



IceFS Disk Layout

Each cube has isolated metadata

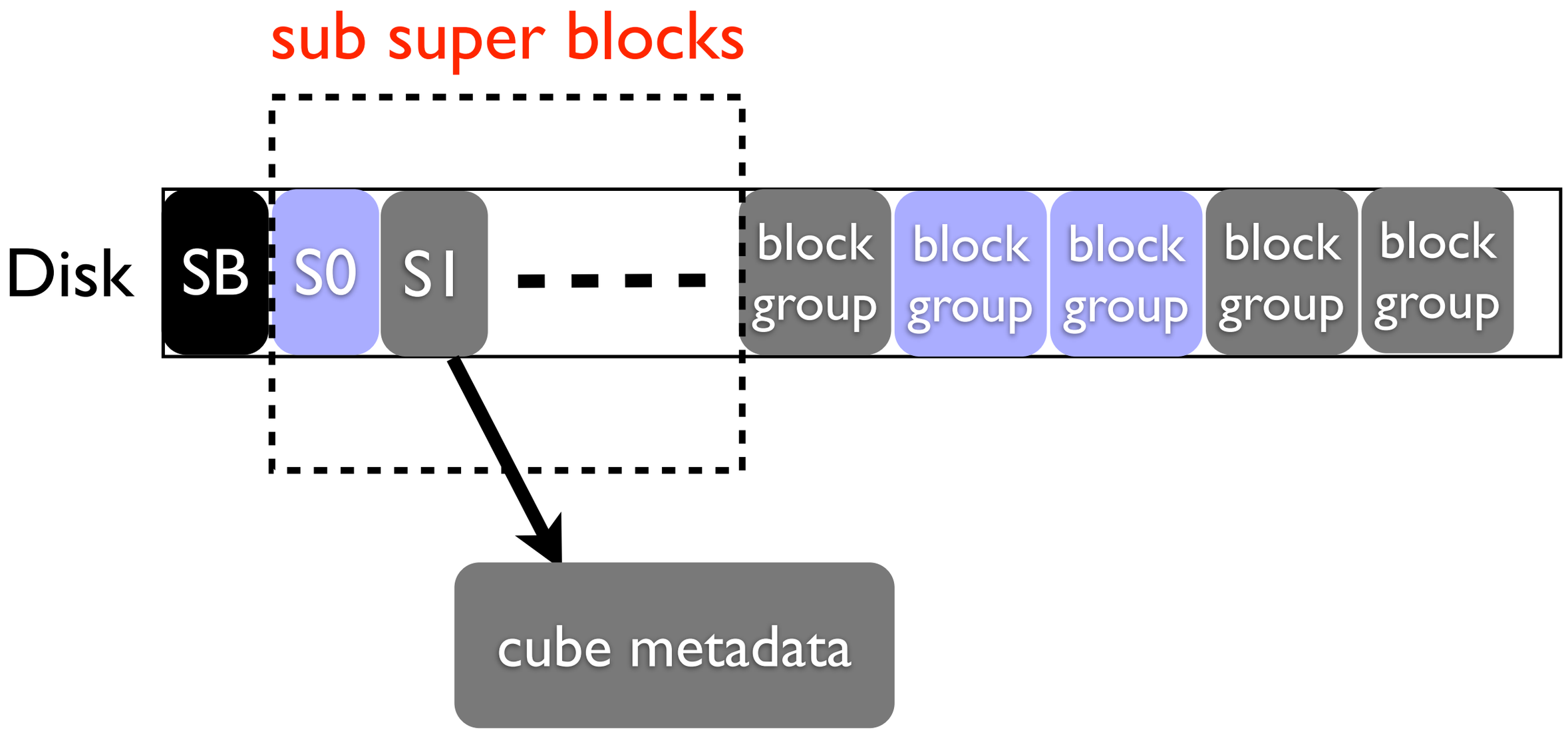
- sub-super block (Si) and isolated block groups



IceFS Disk Layout

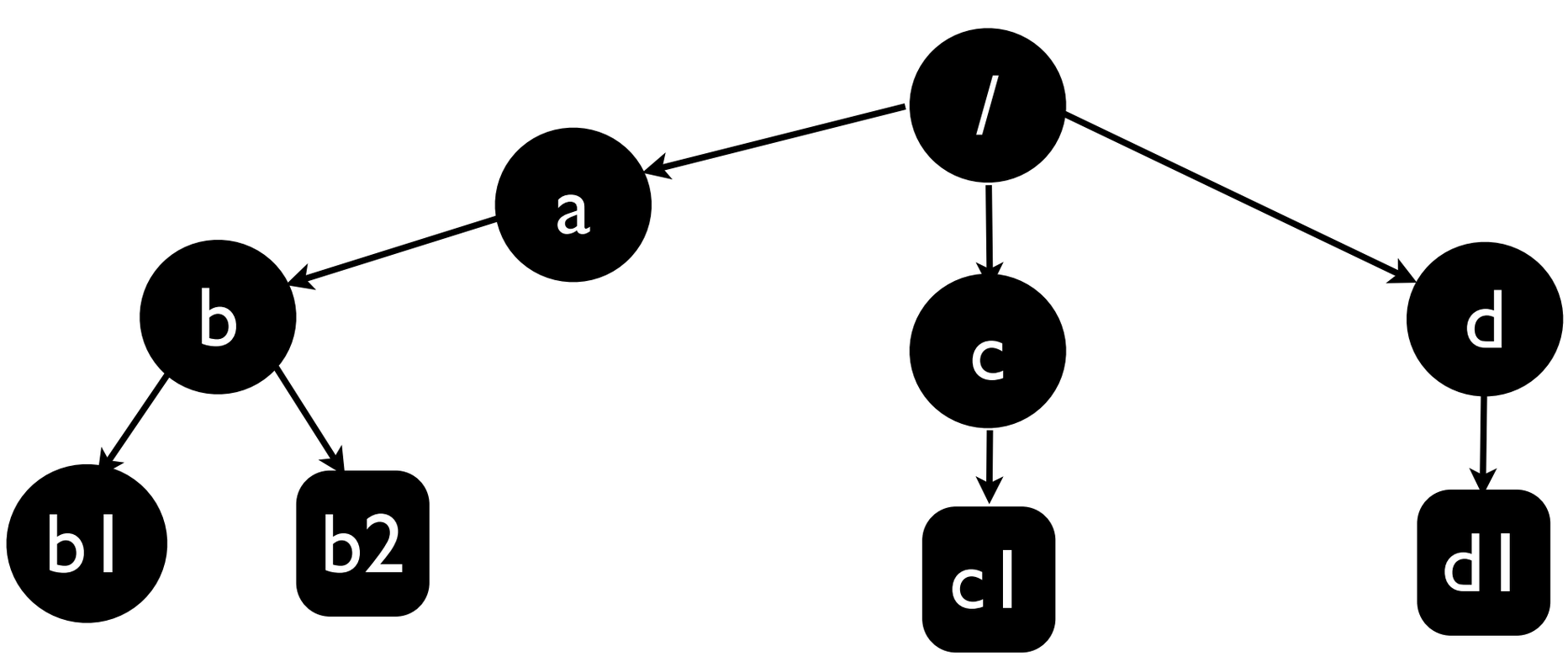
Each cube has isolated metadata

- sub-super block (Si) and isolated block groups

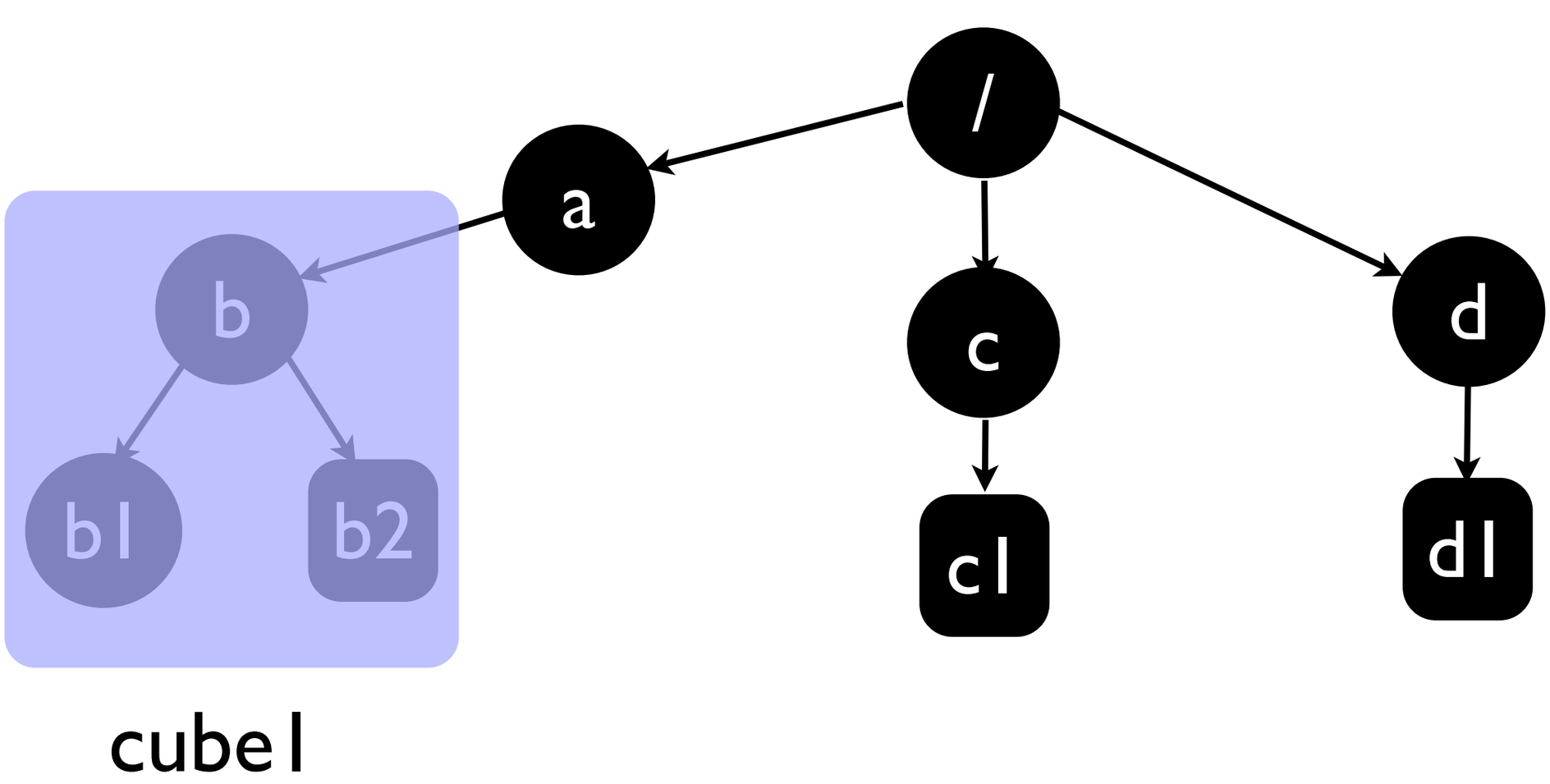


Directory Indirection

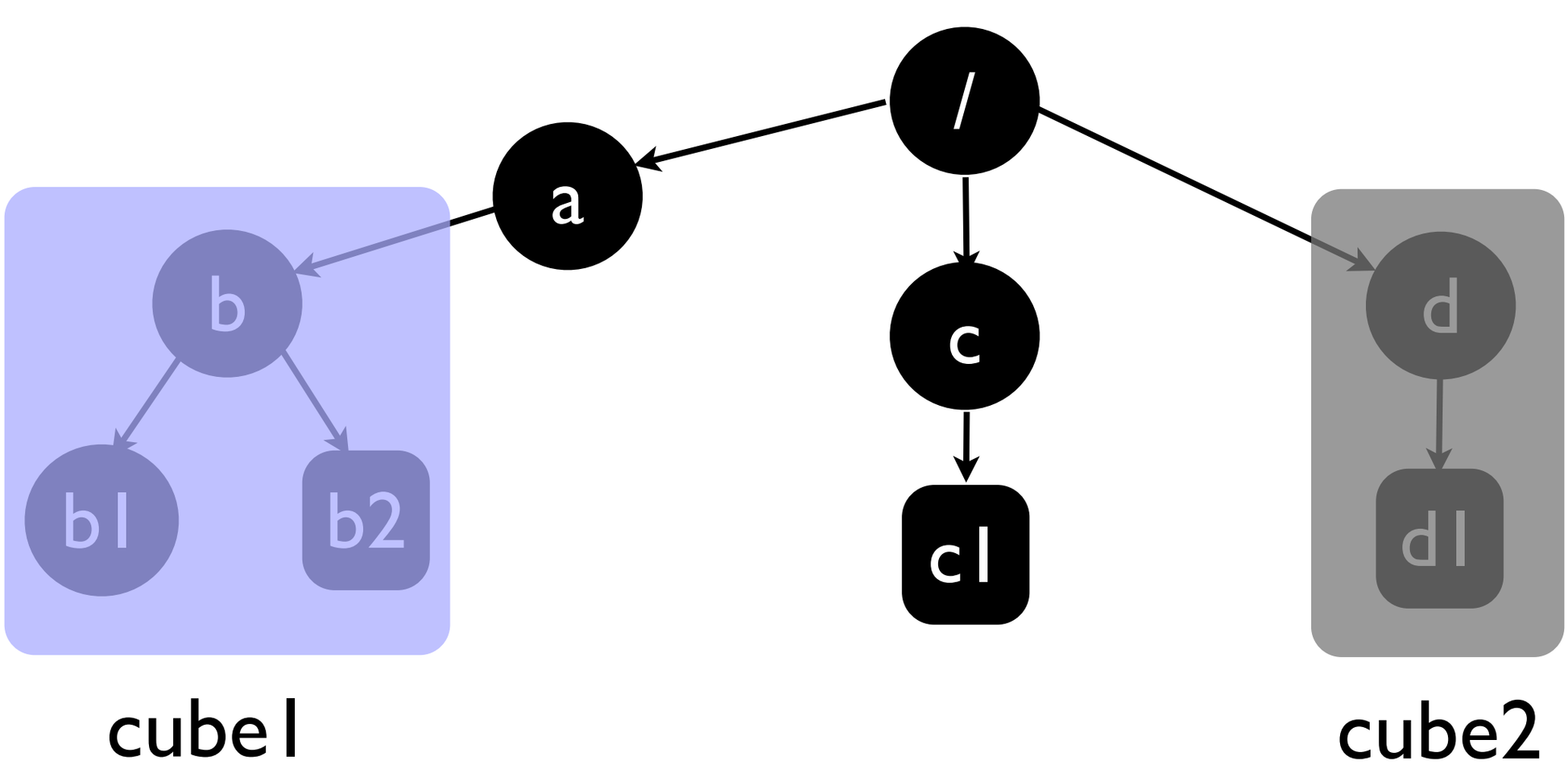
Directory Indirection



Directory Indirection



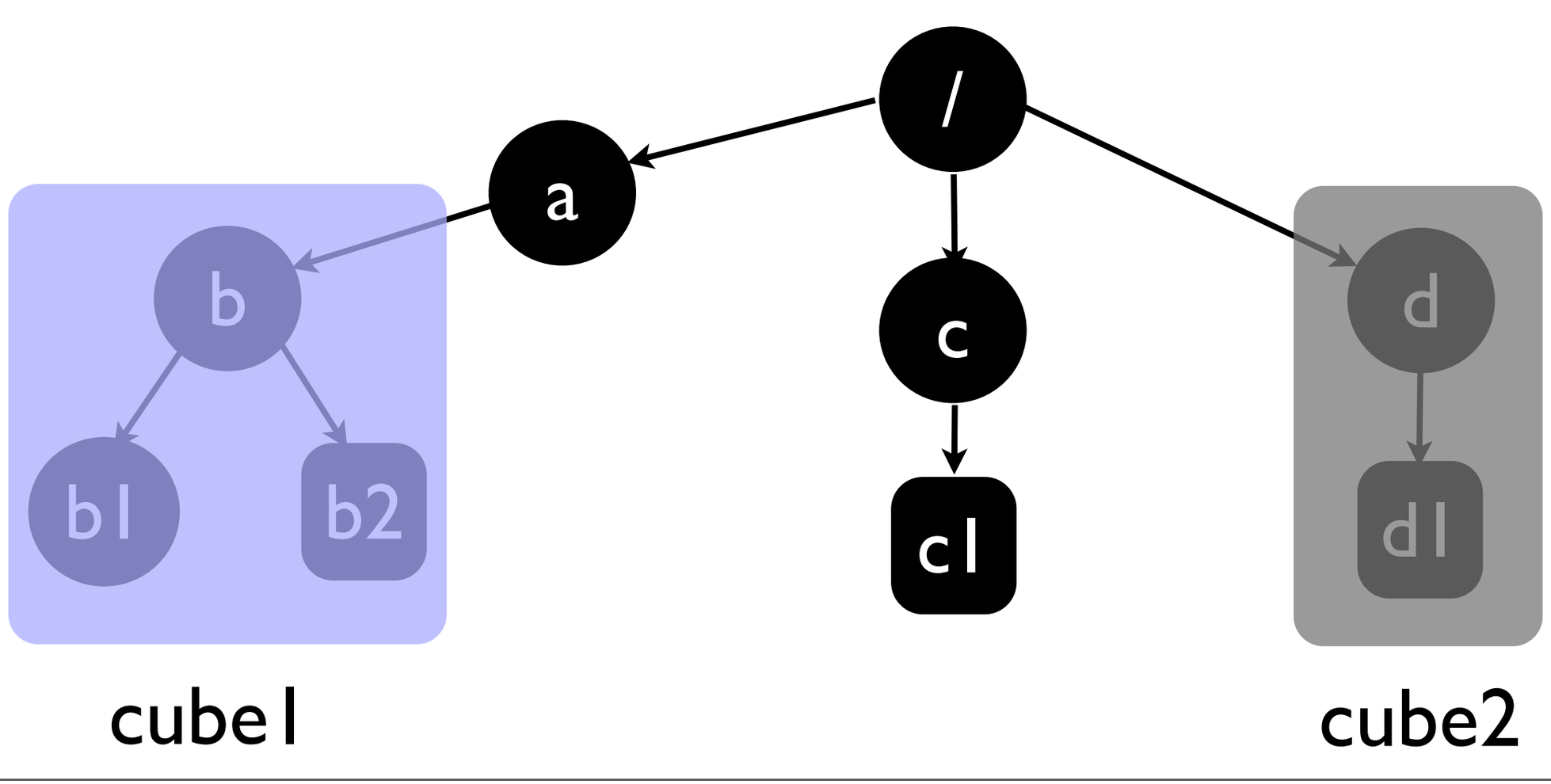
Directory Indirection



Directory Indirection

1. load cube pathnames
from sub-super blocks

/a/b/, cube1 dentry
/d/, cube2 dentry
... ...



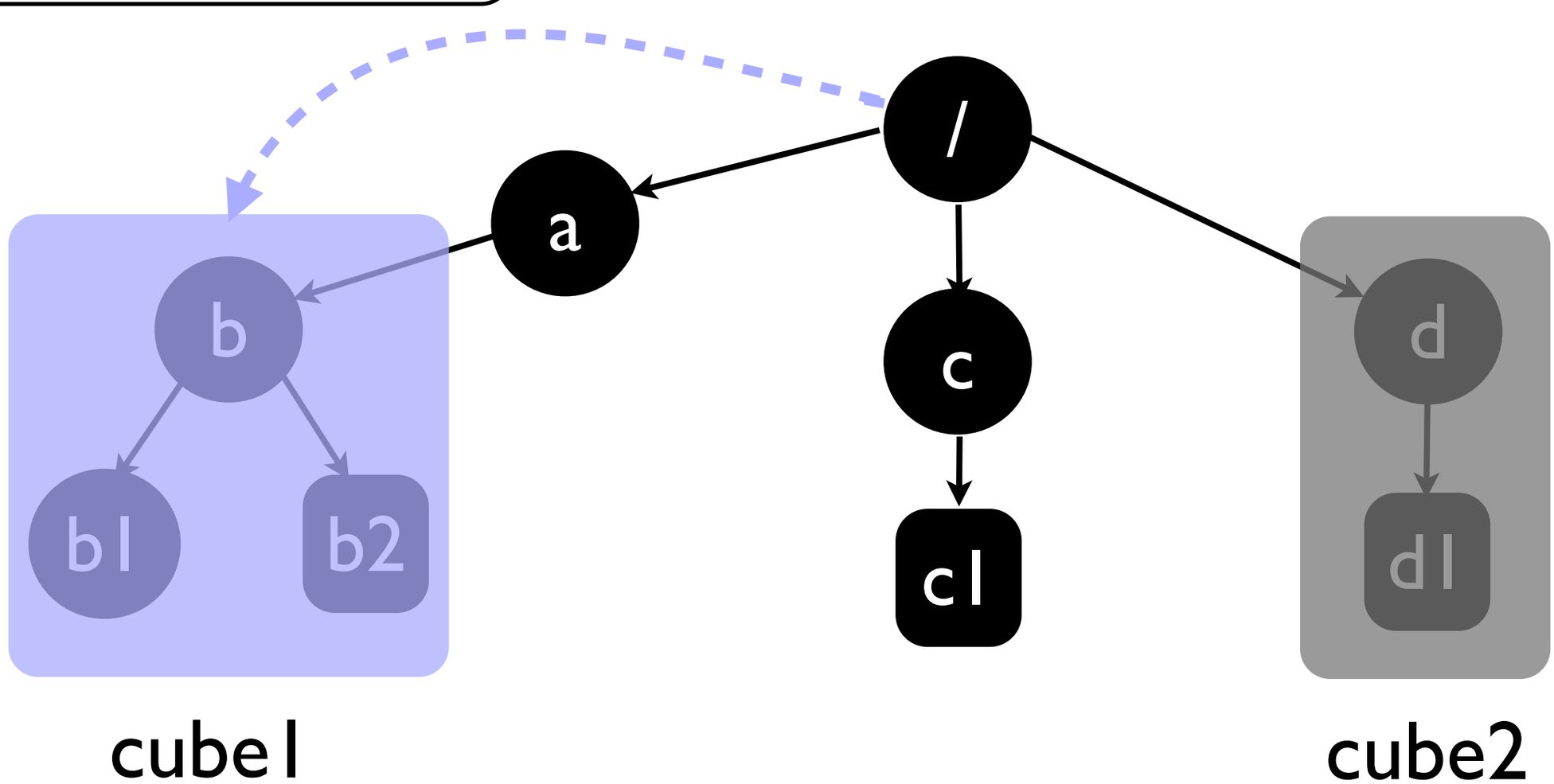
Directory Indirection

1. load cube pathnames from sub-super blocks

/a/b/, cube1 dentry
/d/, cube2 dentry
... ...

2. pathname prefix match

read file "/a/b/b2"
match cube1
jump to cube1 top directory

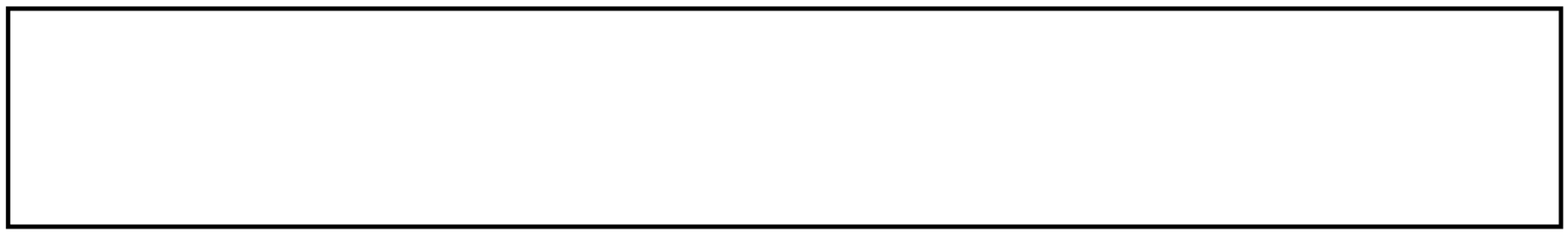


Ext3/4 Transaction

Ext3/4 Transaction

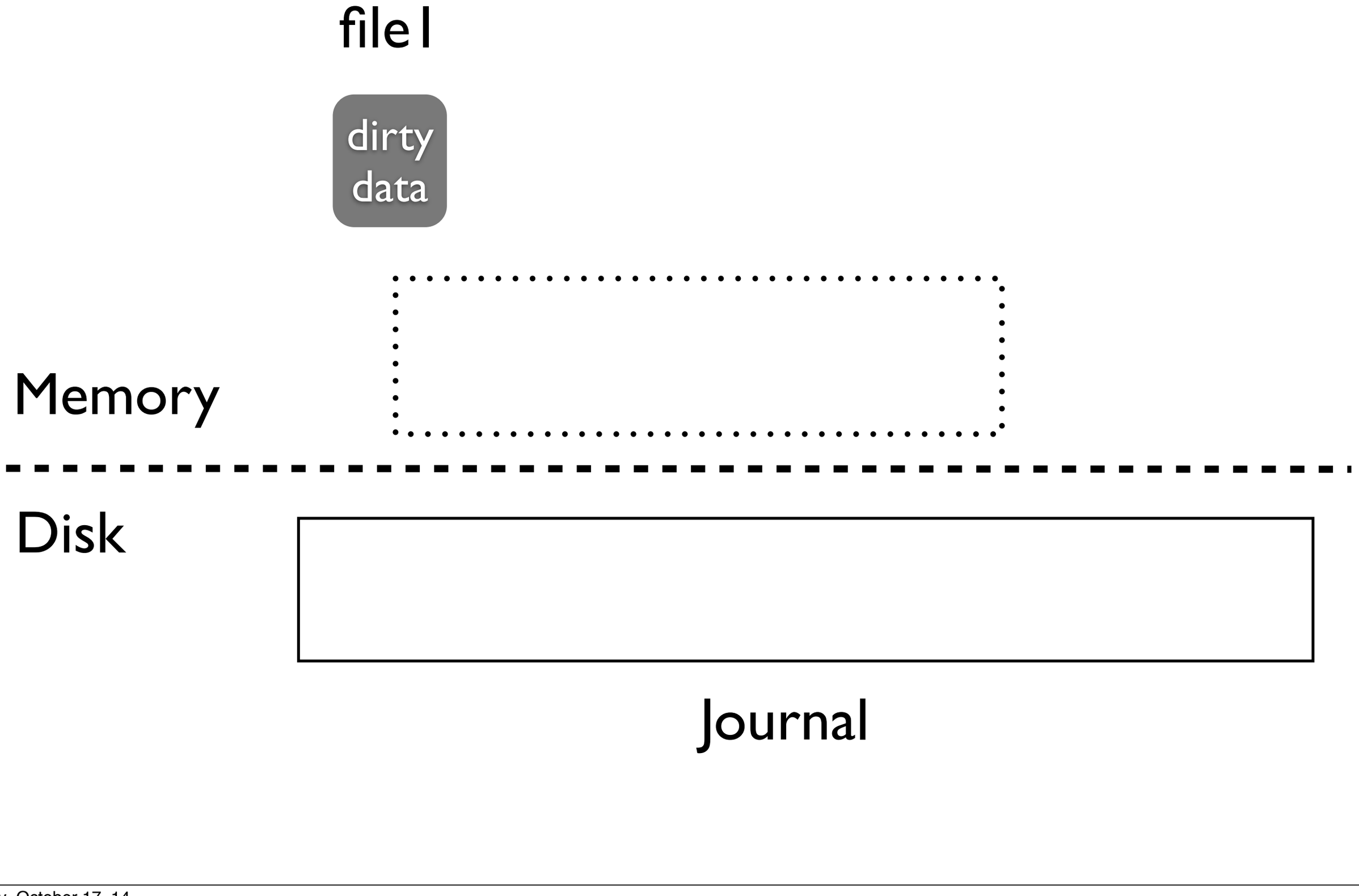
Memory

Disk



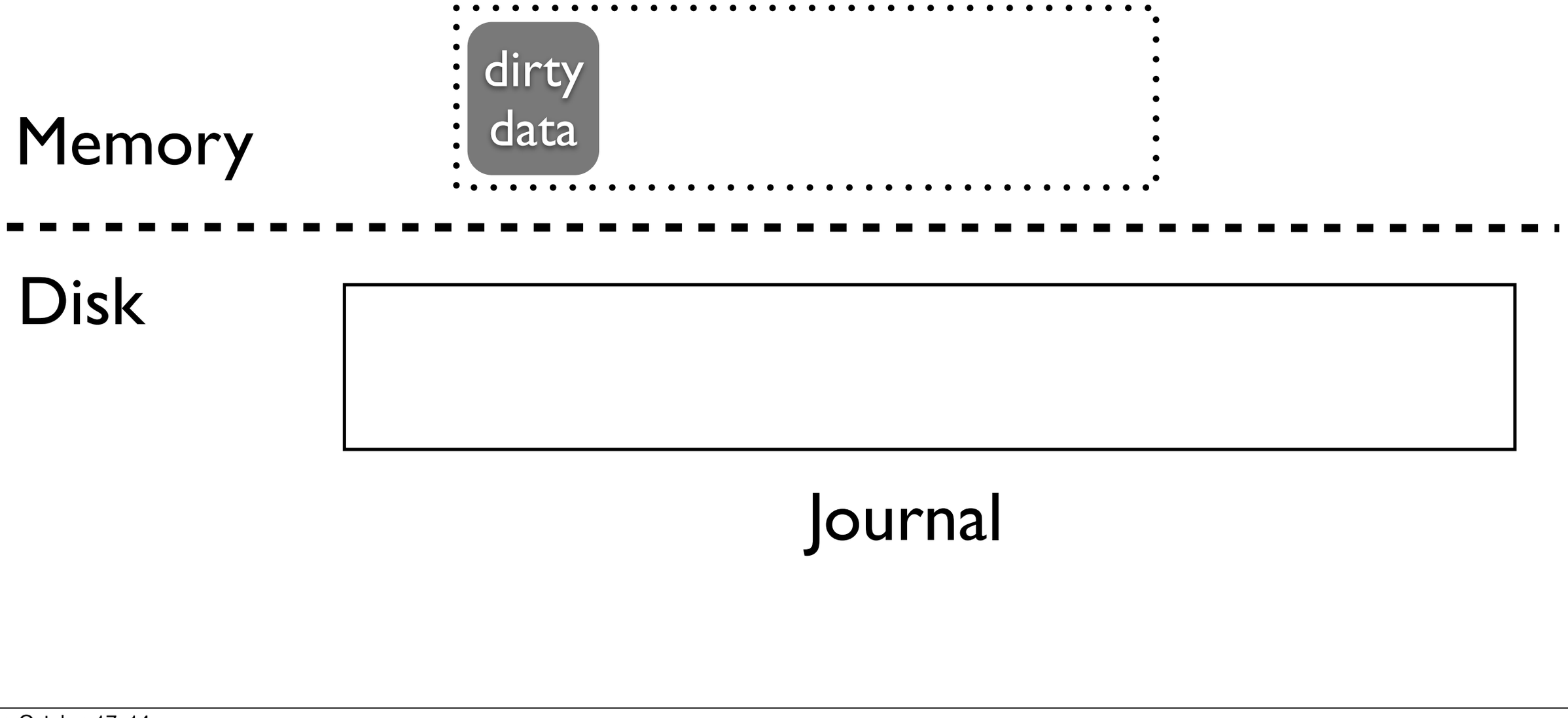
Journal

Ext3/4 Transaction

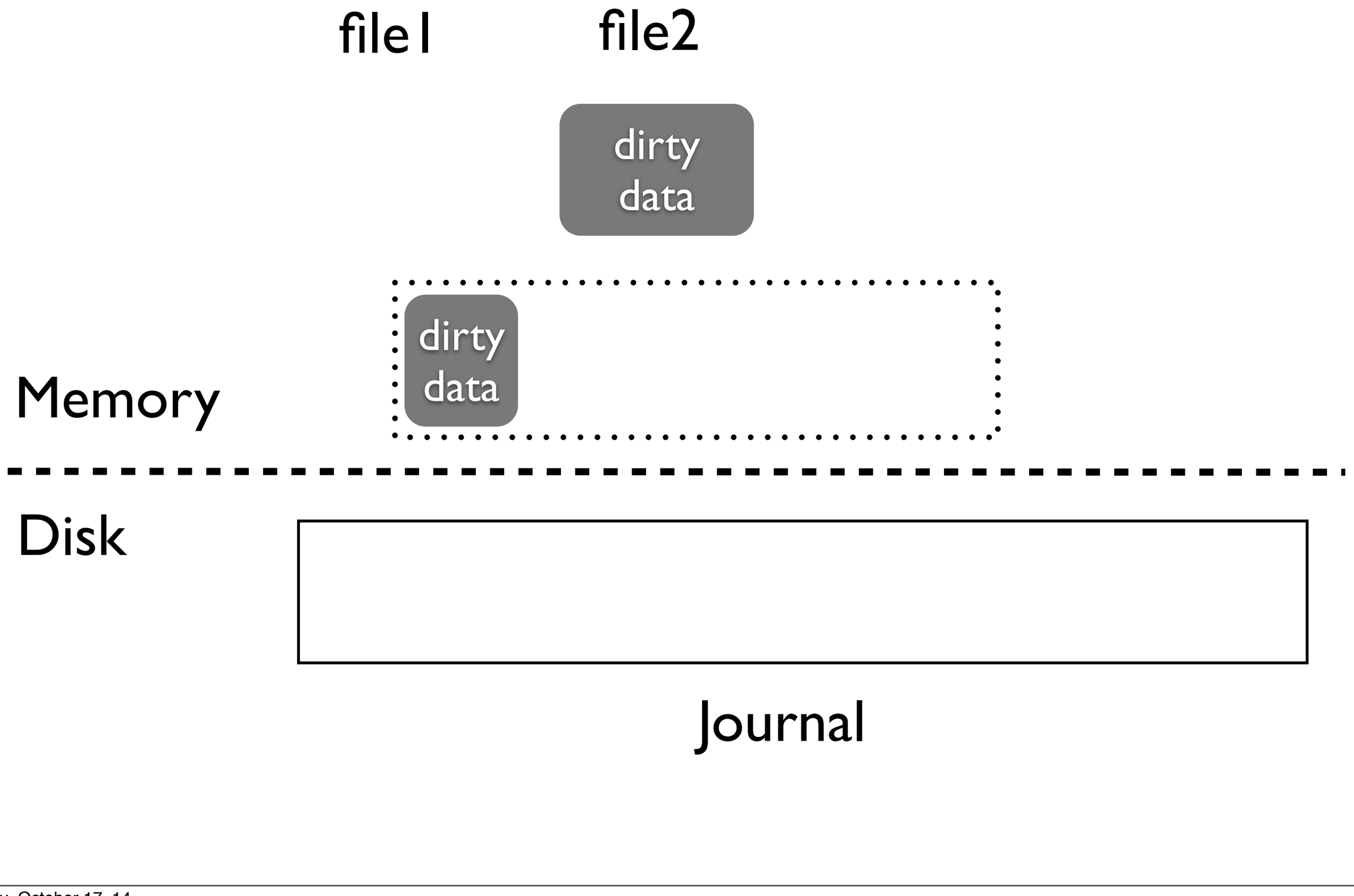


Ext3/4 Transaction

file I



Ext3/4 Transaction

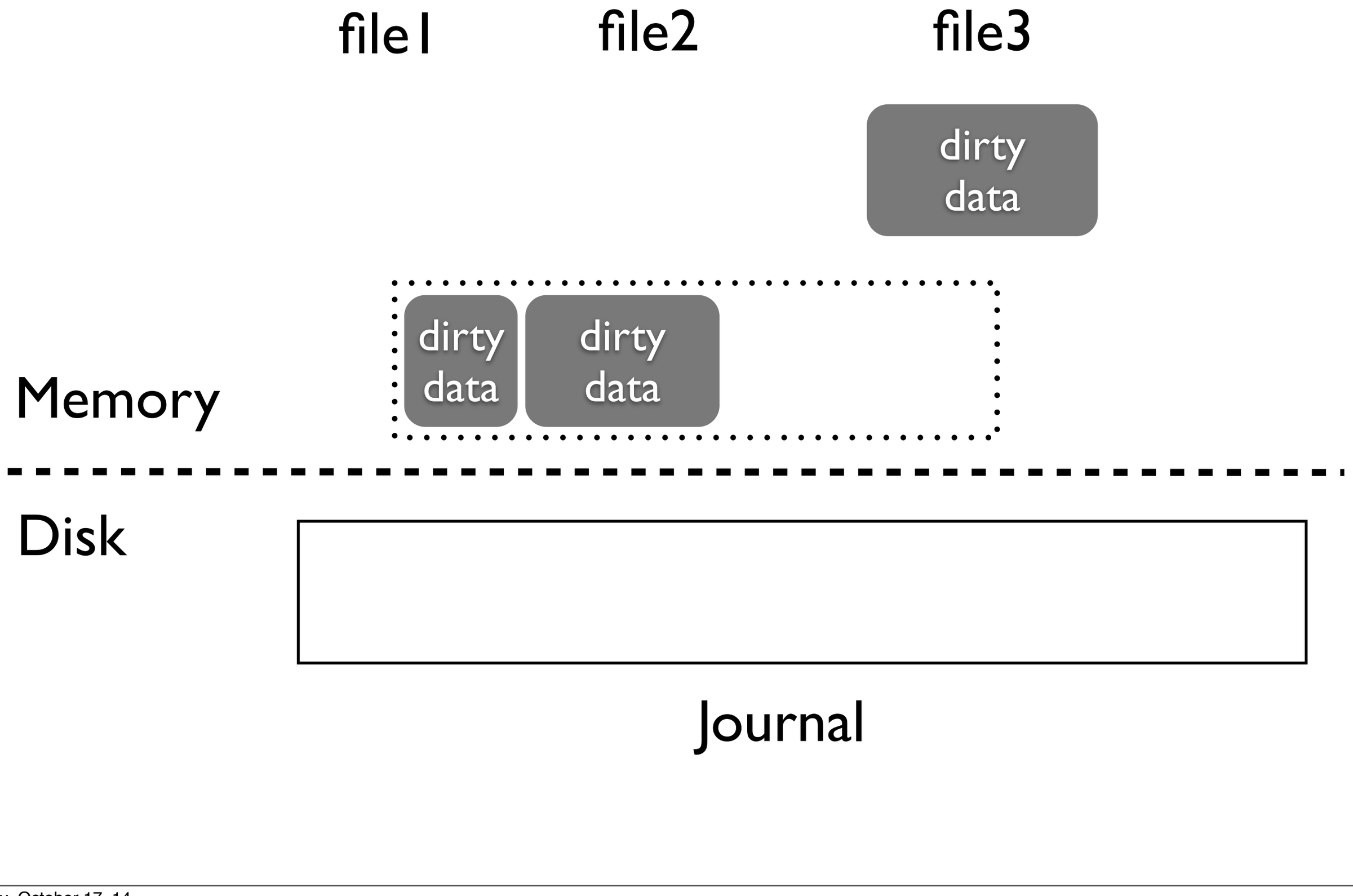


Ext3/4 Transaction

file1 file2

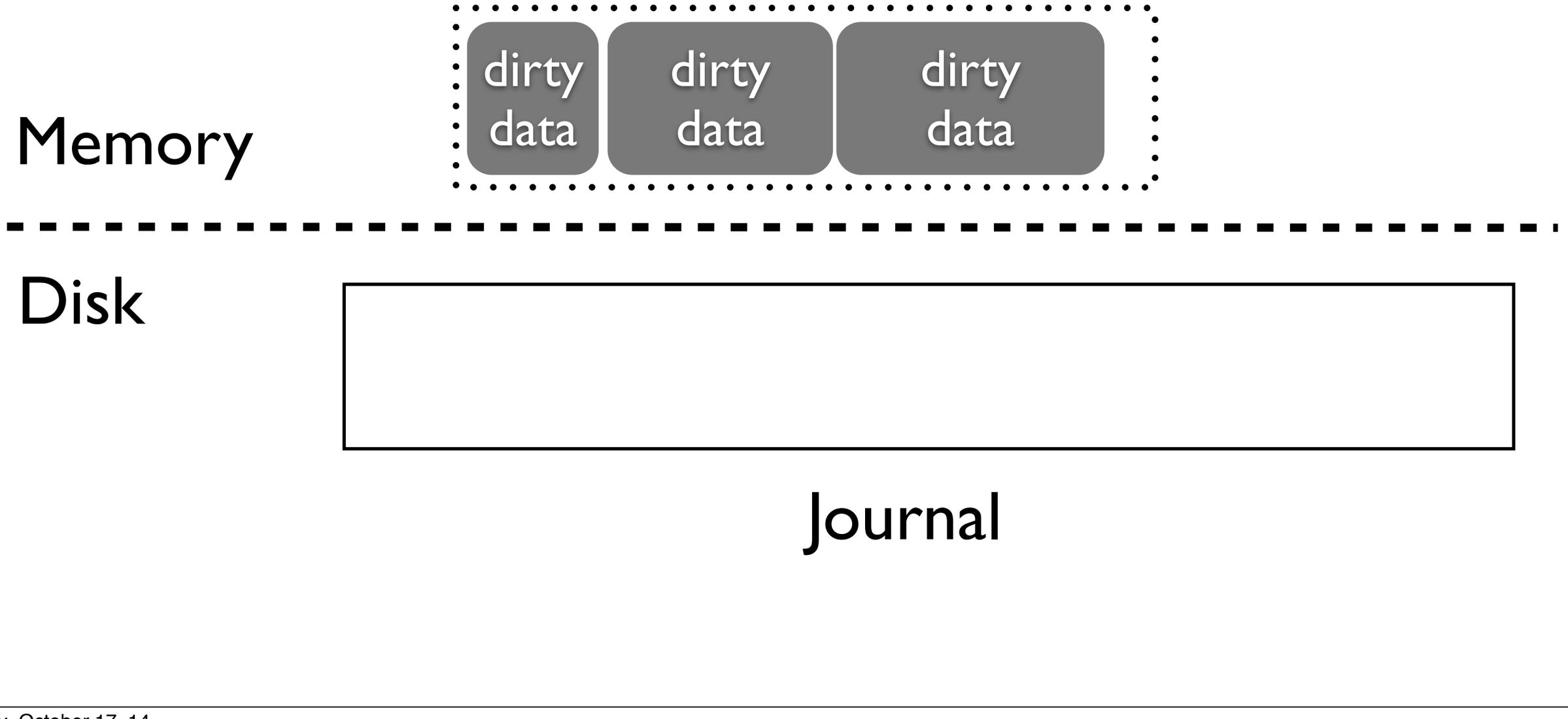


Ext3/4 Transaction



Ext3/4 Transaction

file1 file2 file3



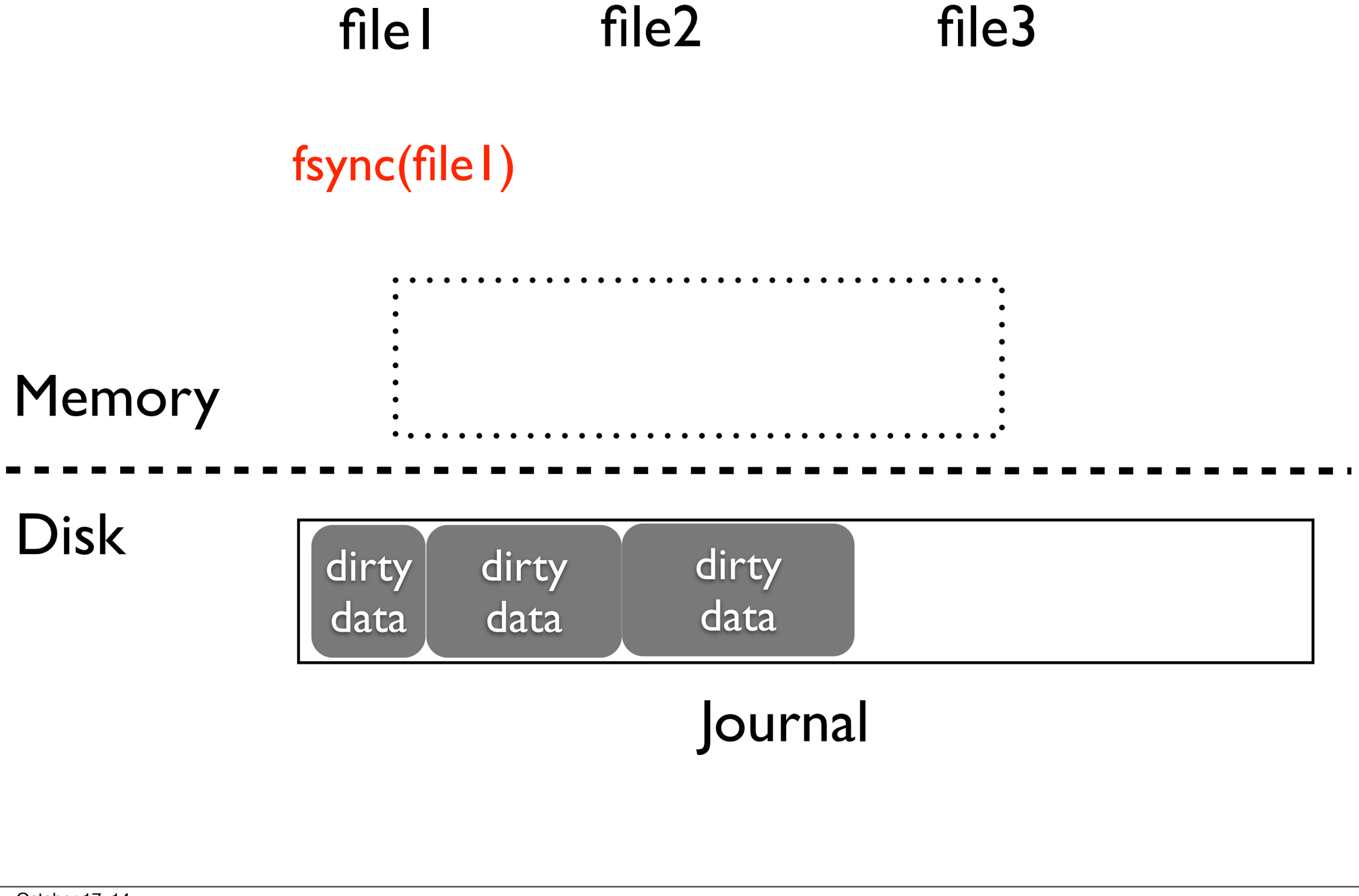
Ext3/4 Transaction

file1 file2 file3

fsync(file1)



Ext3/4 Transaction



Ext3/4 Transaction

file1 file2 file3

fsync(file1)

Memory

Disk



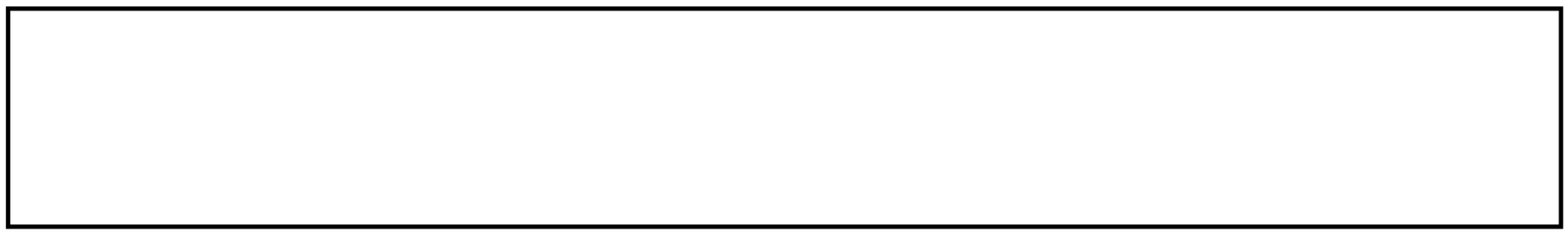
Journal

IceFS Transaction Splitting

IceFS Transaction Splitting

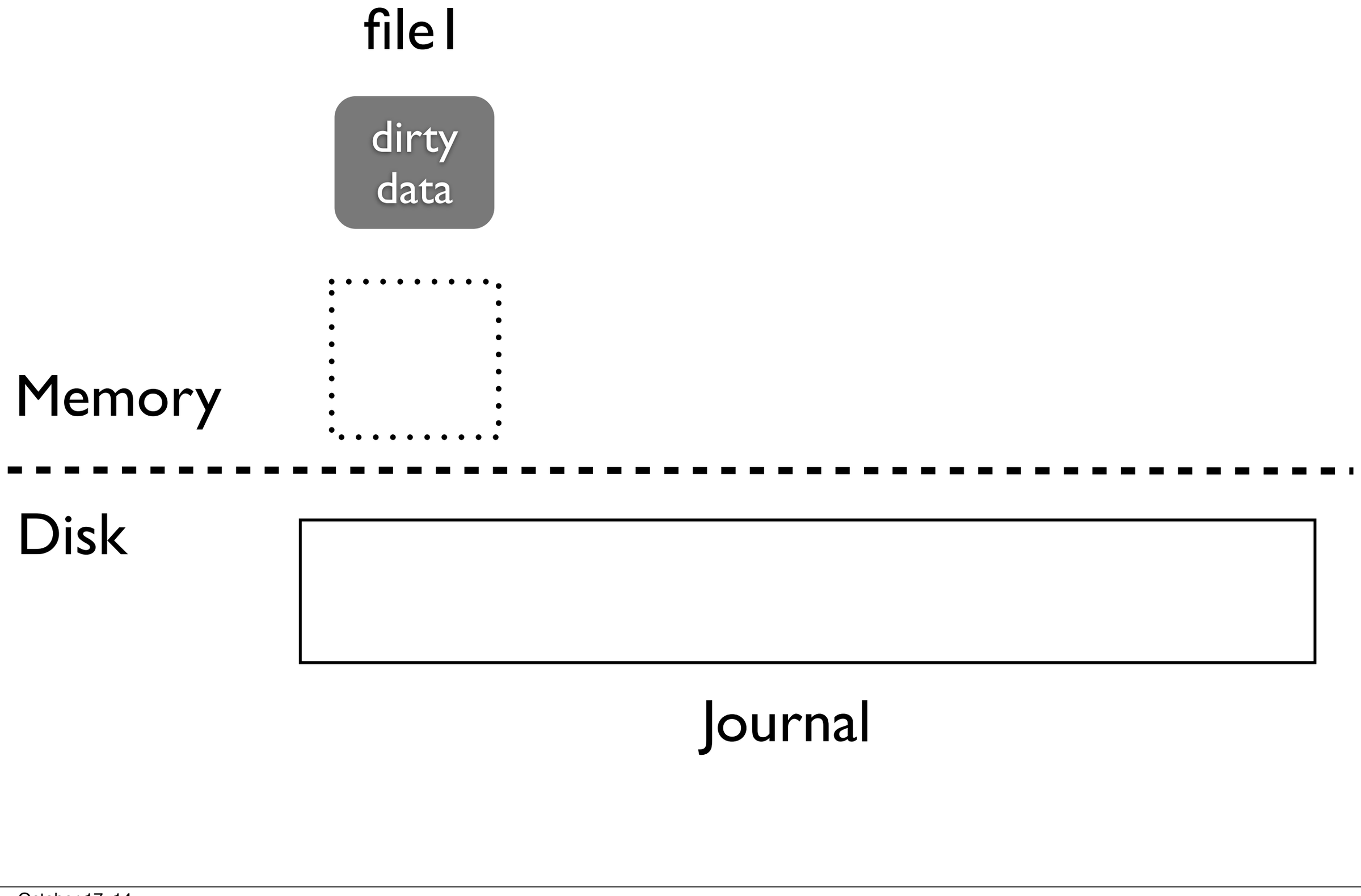
Memory

Disk



Journal

IceFS Transaction Splitting

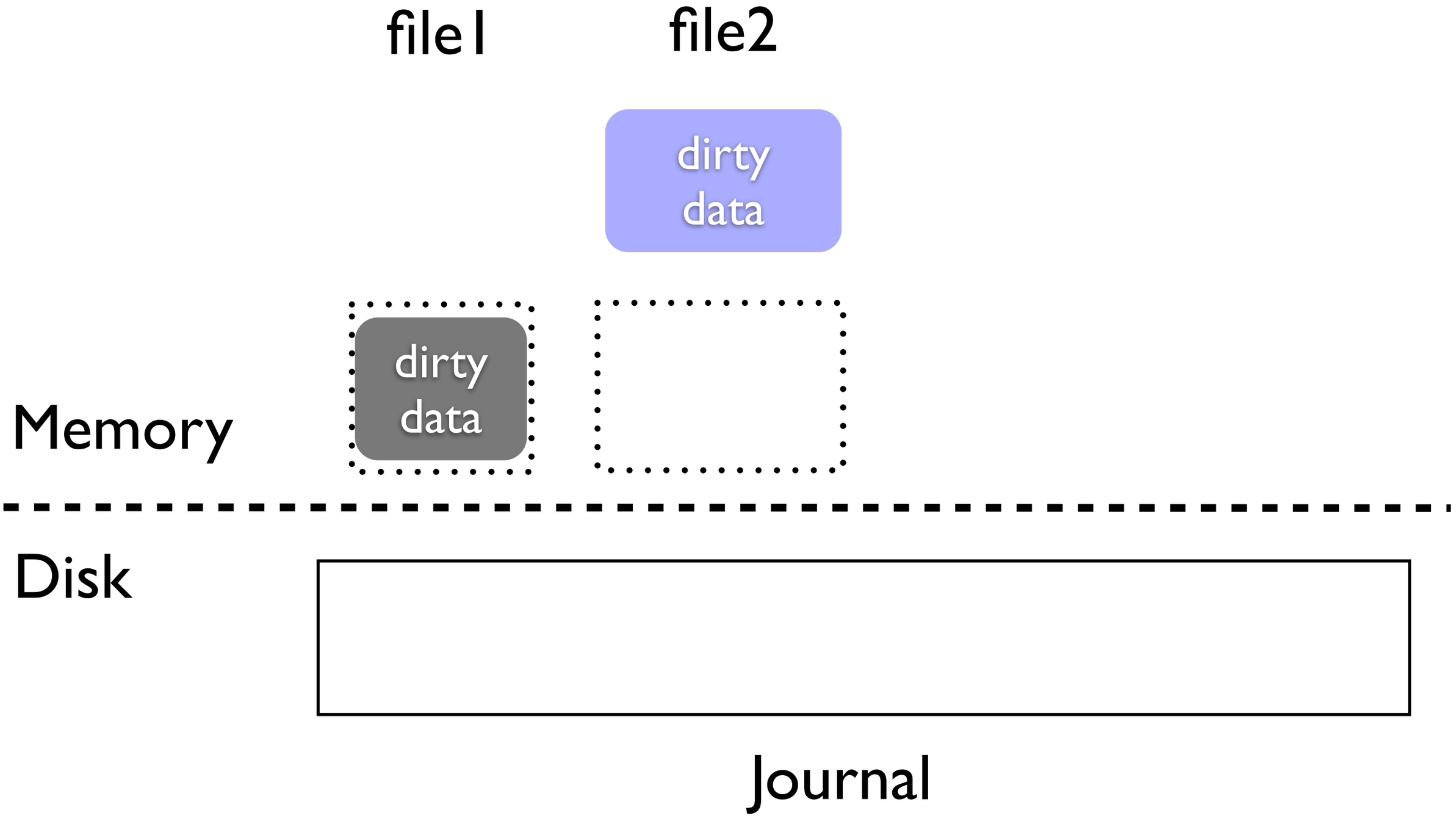


IceFS Transaction Splitting

file 1



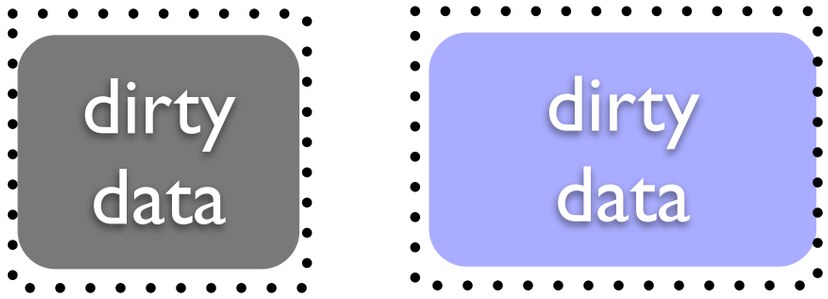
IceFS Transaction Splitting



IceFS Transaction Splitting

file1 file2

Memory

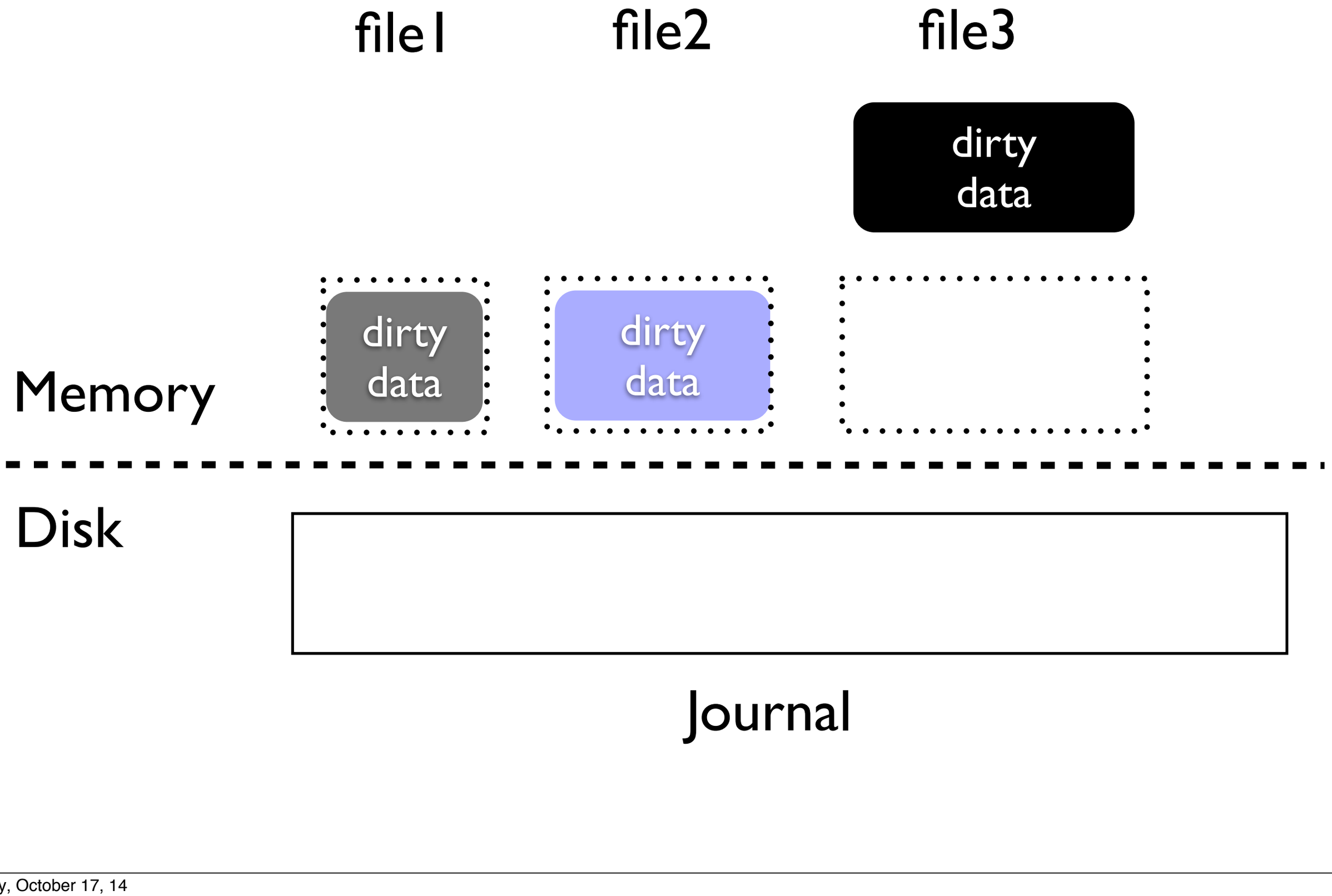


Disk

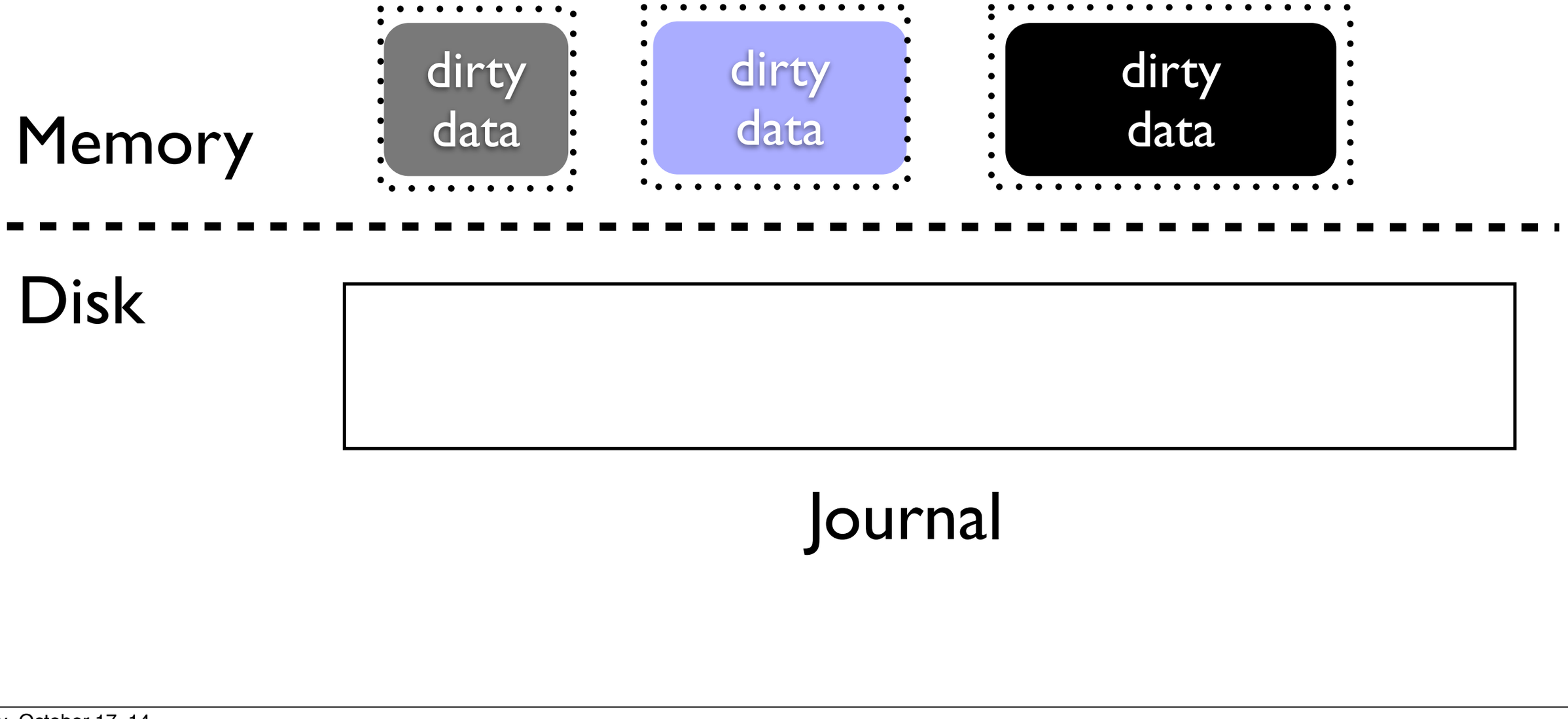


Journal

IceFS Transaction Splitting



IceFS Transaction Splitting



IceFS Transaction Splitting

file1 file2 file3

fsync(file1)

Memory



Disk



Journal

IceFS Transaction Splitting

file1 file2 file3

fsync(file1)

Memory

dirty data

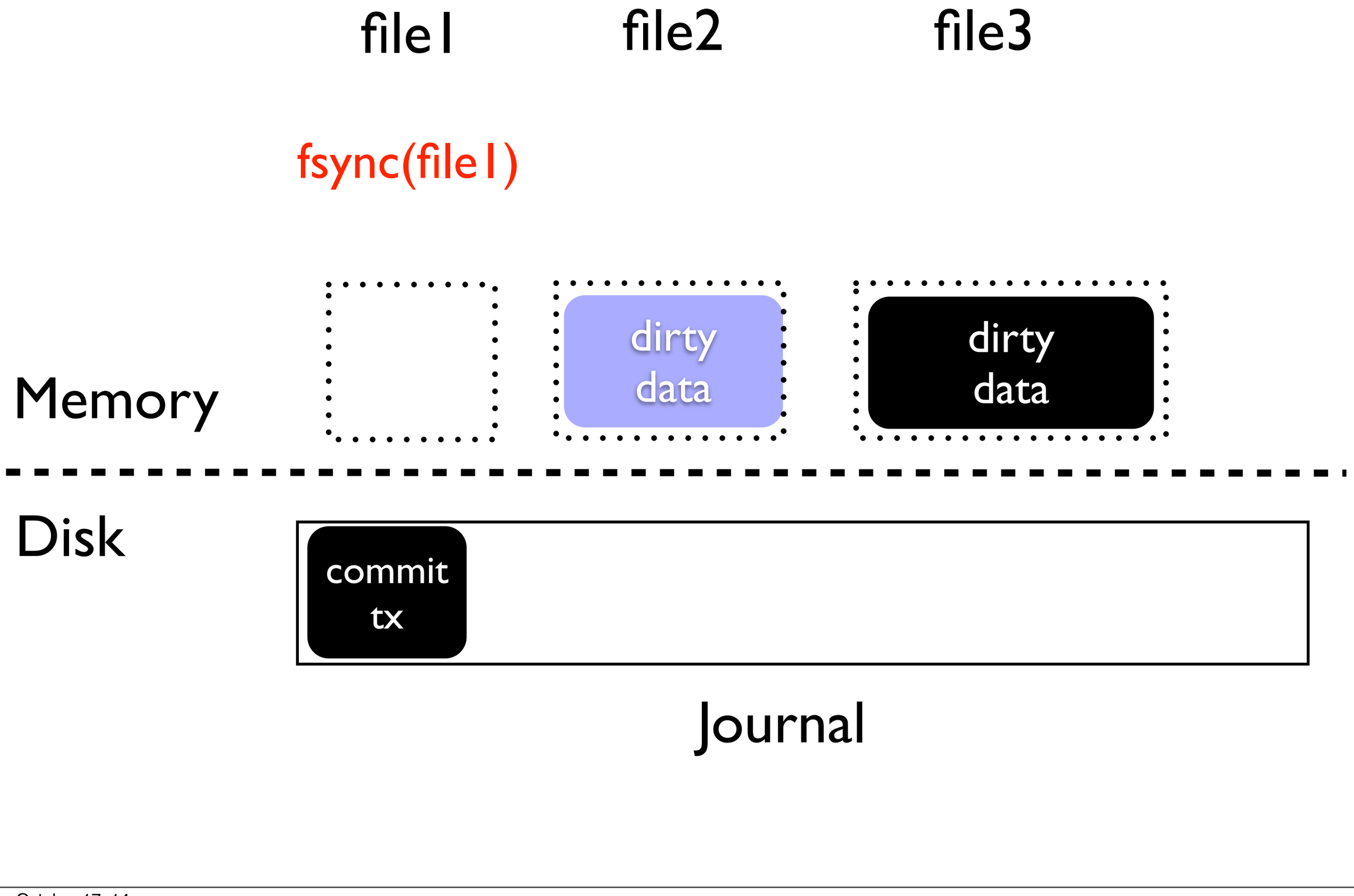
dirty data

Disk

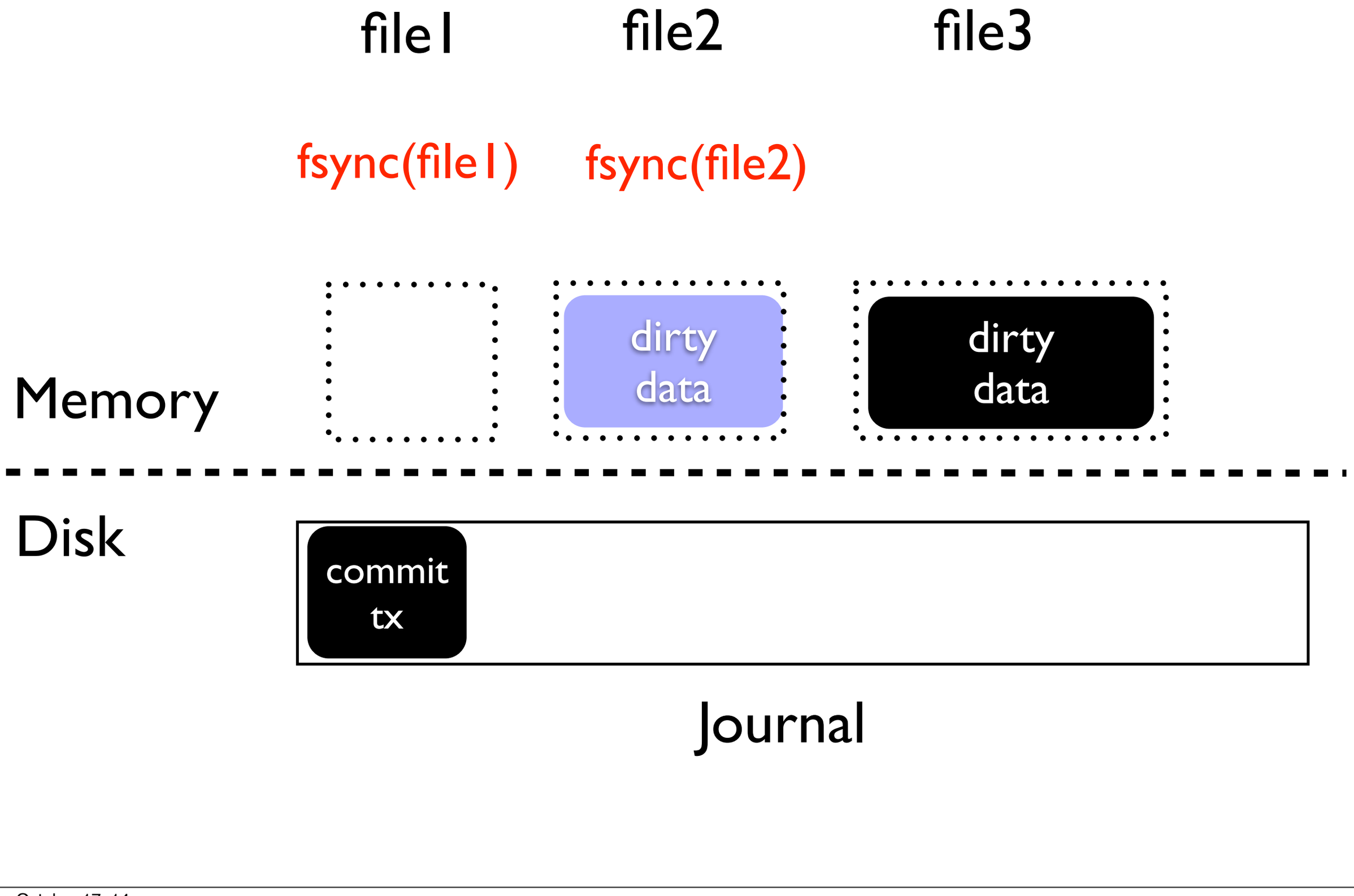
dirty data

Journal

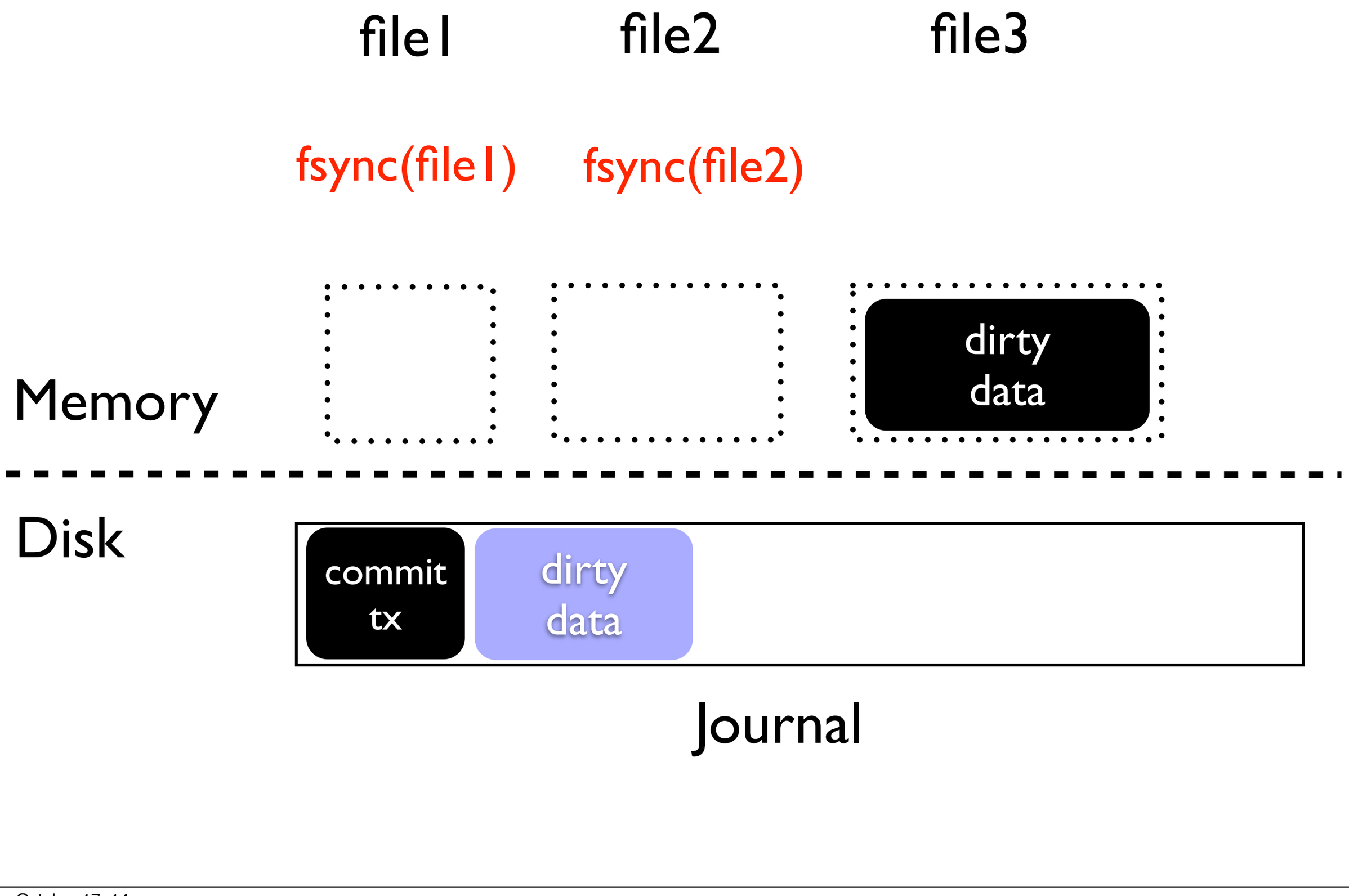
IceFS Transaction Splitting



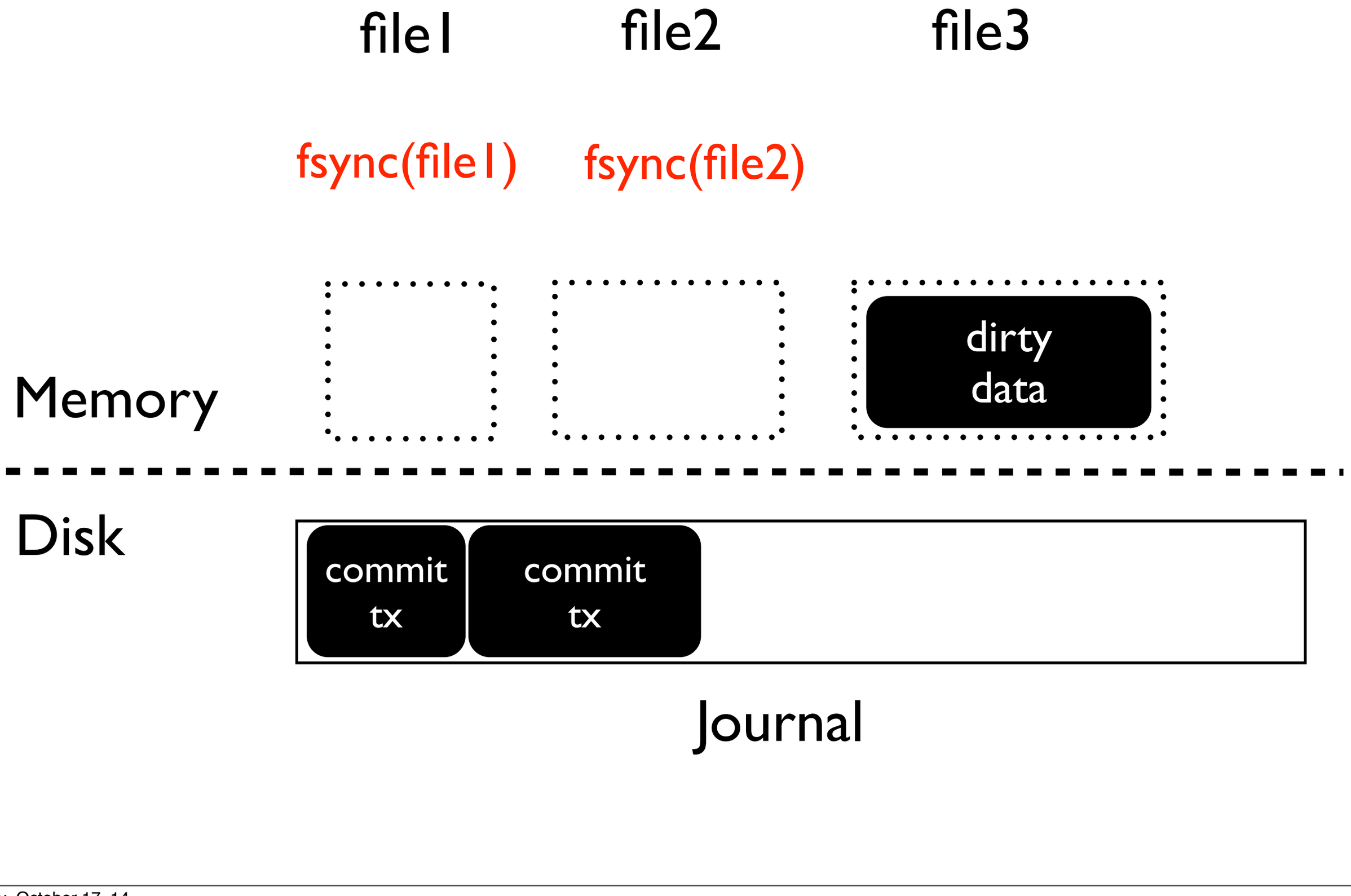
IceFS Transaction Splitting



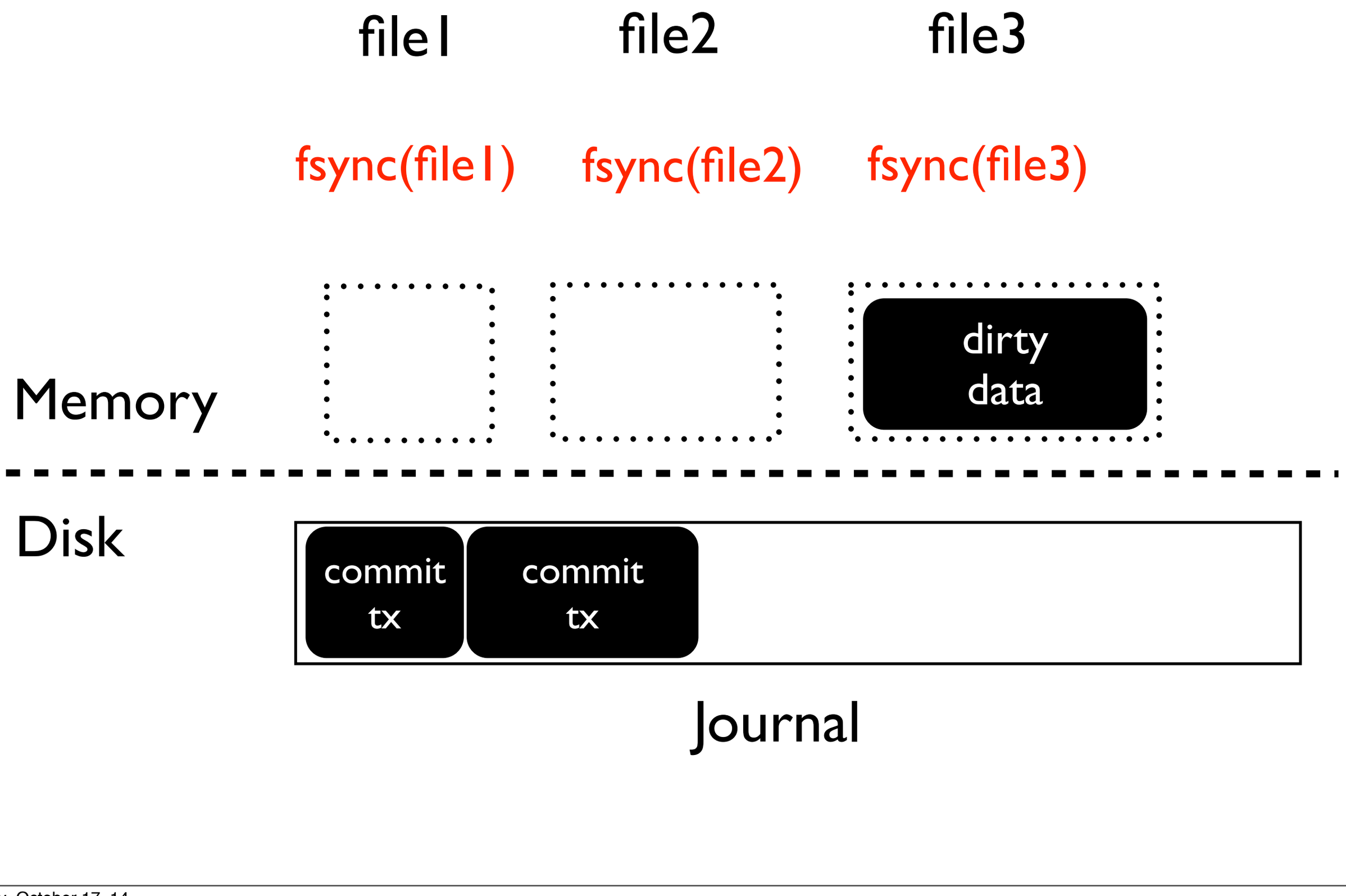
IceFS Transaction Splitting



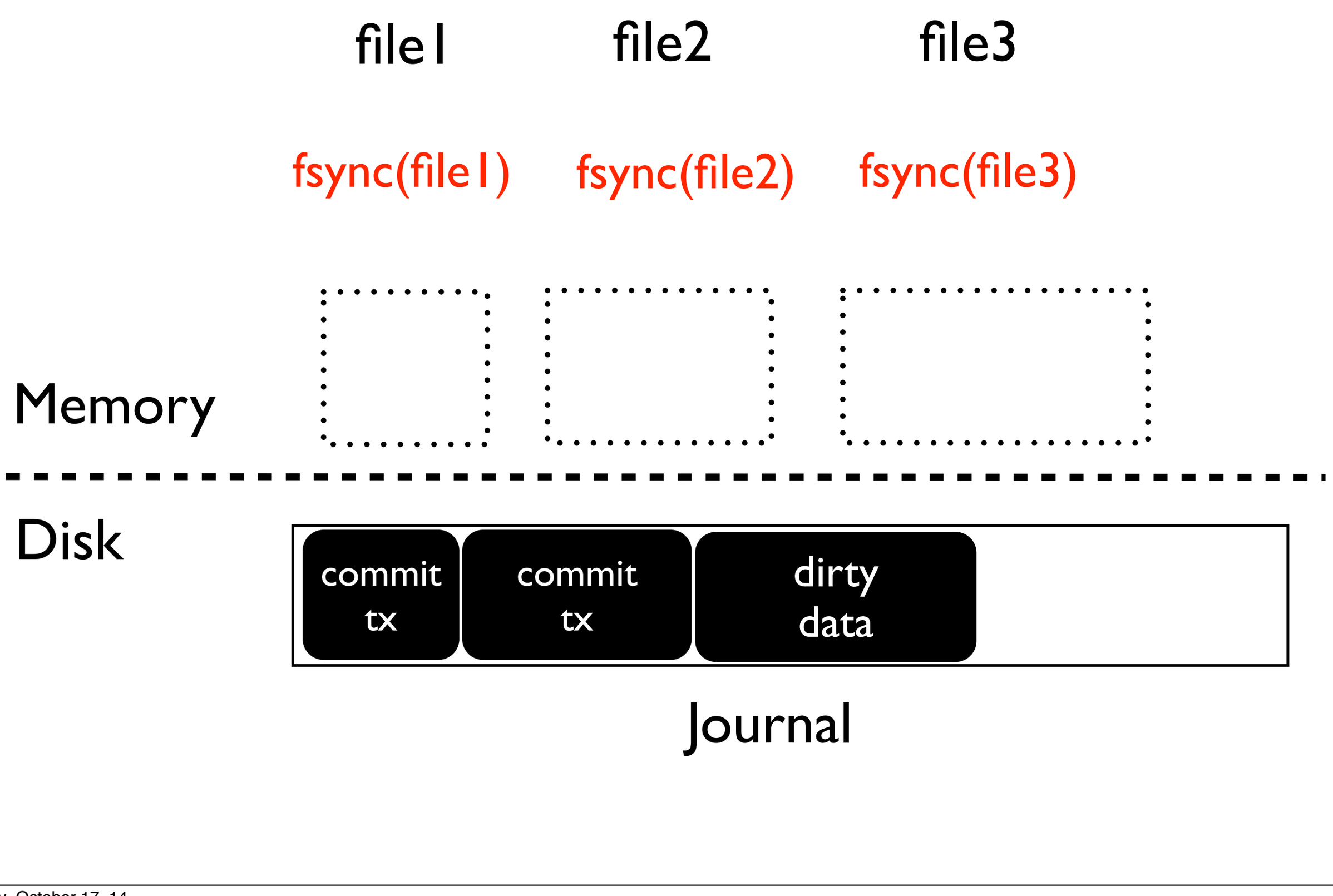
IceFS Transaction Splitting



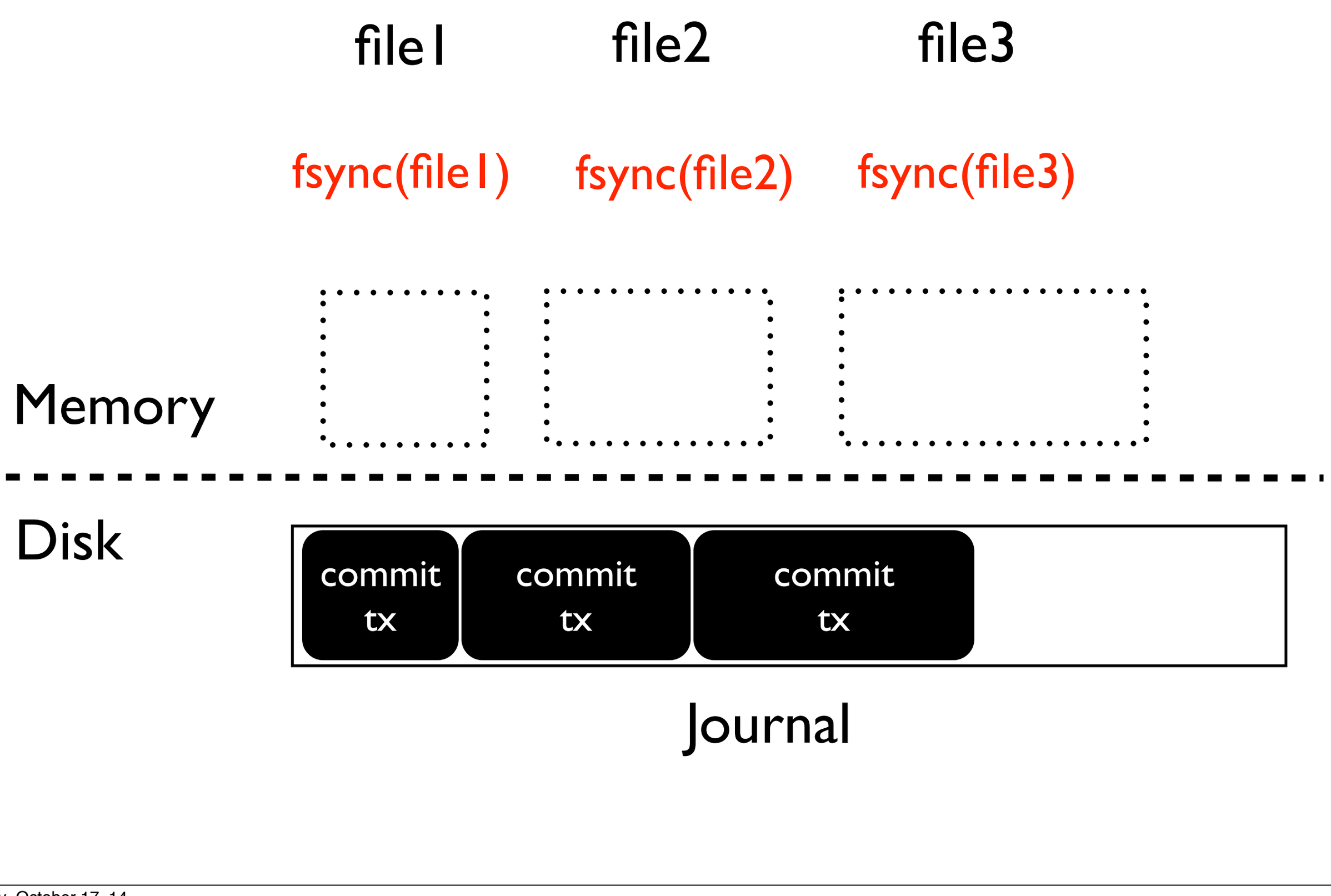
IceFS Transaction Splitting



IceFS Transaction Splitting



IceFS Transaction Splitting



Introduction

File System Disentanglement

The Ice File System

- Design and implementation
- Benefits

Evaluation

Conclusion

Benefits of Disentanglement

Benefits of Disentanglement

Localized reactions to failures

- per-cube read-only and crash
- encourage more runtime checking

Benefits of Disentanglement

Localized reactions to failures

- per-cube read-only and crash
- encourage more runtime checking

Localized recovery

- only check faulty cubes
- offline and online

Benefits of Disentanglement

Localized reactions to failures

- per-cube read-only and crash
- encourage more runtime checking

Localized recovery

- only check faulty cubes
- offline and online

Specialized journaling

- concurrent and independent transactions
- diverse journal modes (e.g., no journal, no fsync)

Introduction

File System Disentanglement

The Ice File System

Evaluation

Conclusion

Evaluation

Evaluation

Does IceFS isolate failures ?

Evaluation

Does IceFS isolate failures ?

- inject around 200 faults
- per-cube failure (read-only or crash) in IceFS

Evaluation

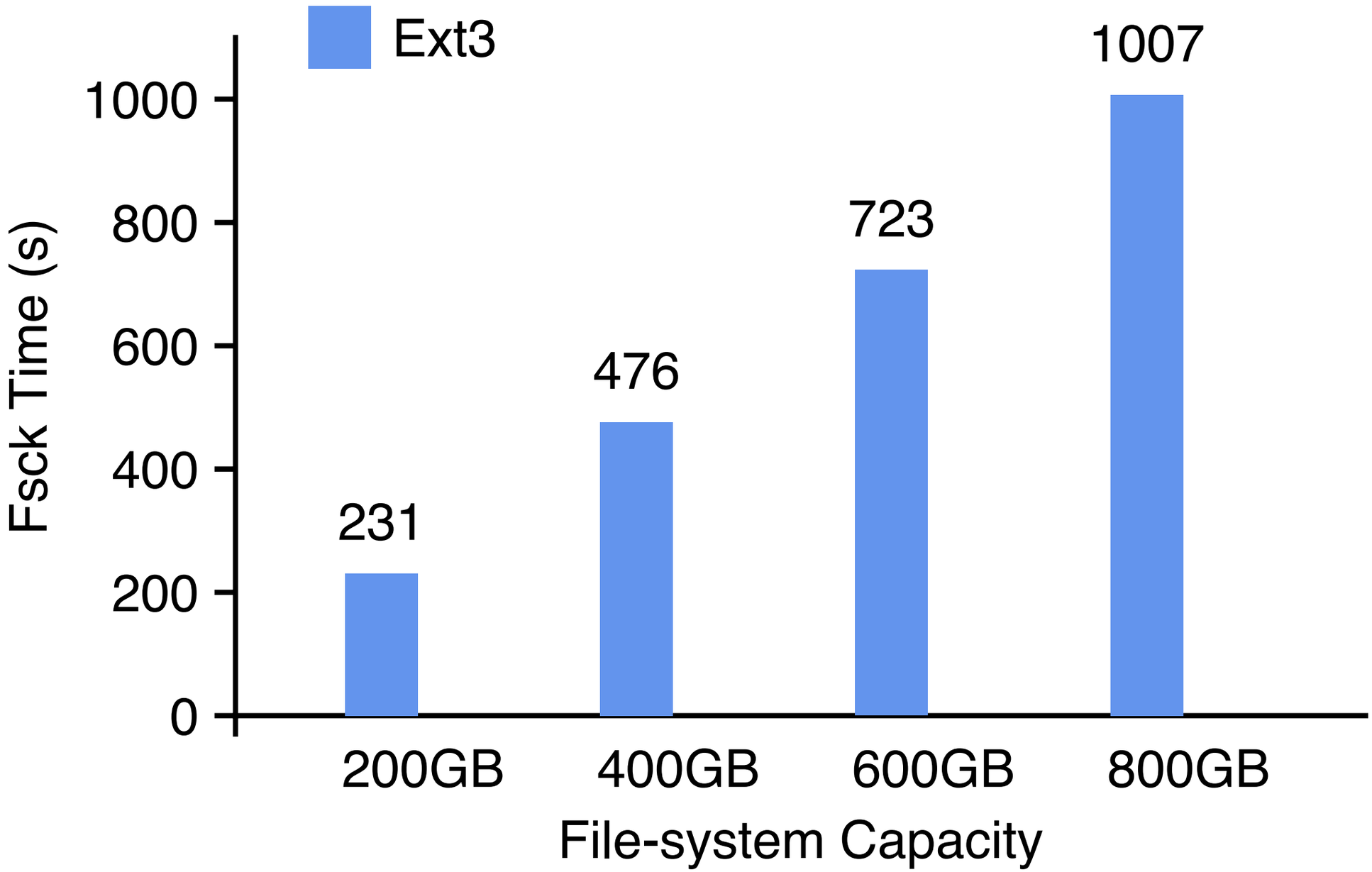
Evaluation

Does IceFS isolate failures ?

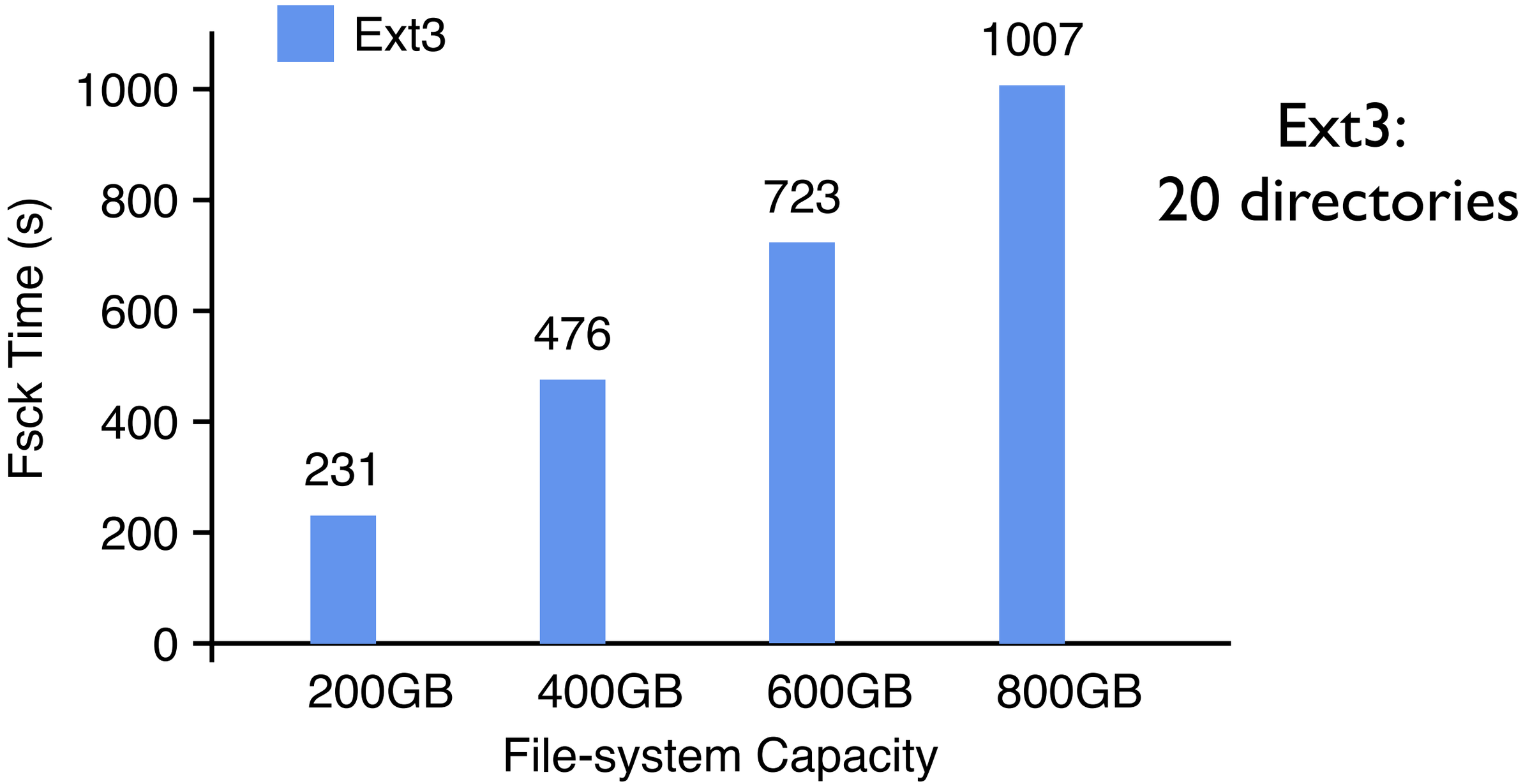
- inject around 200 faults
- per-cube failure (read-only or crash) in IceFS

Does IceFS have faster recovery ?

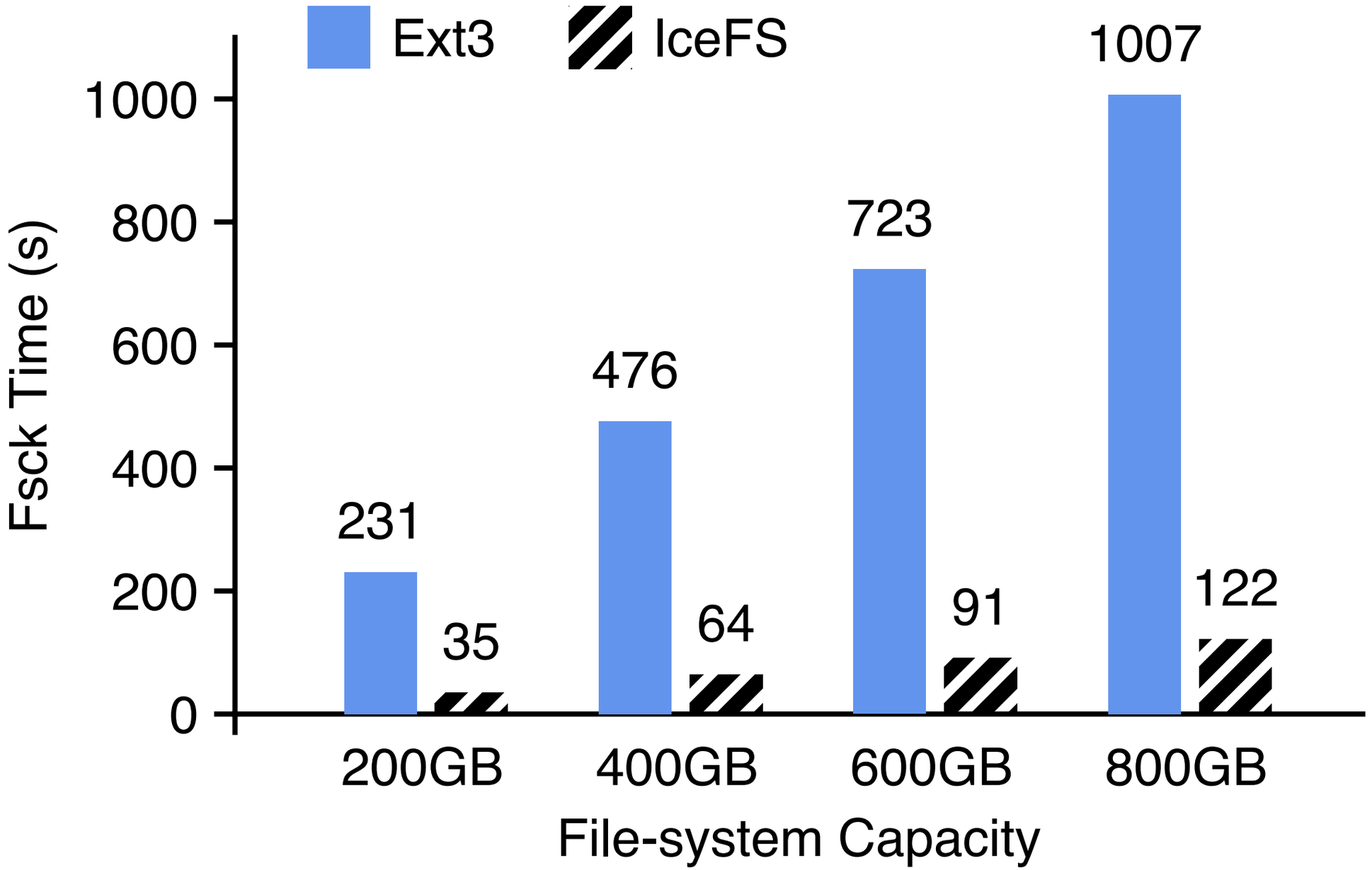
Recovery In Ext3



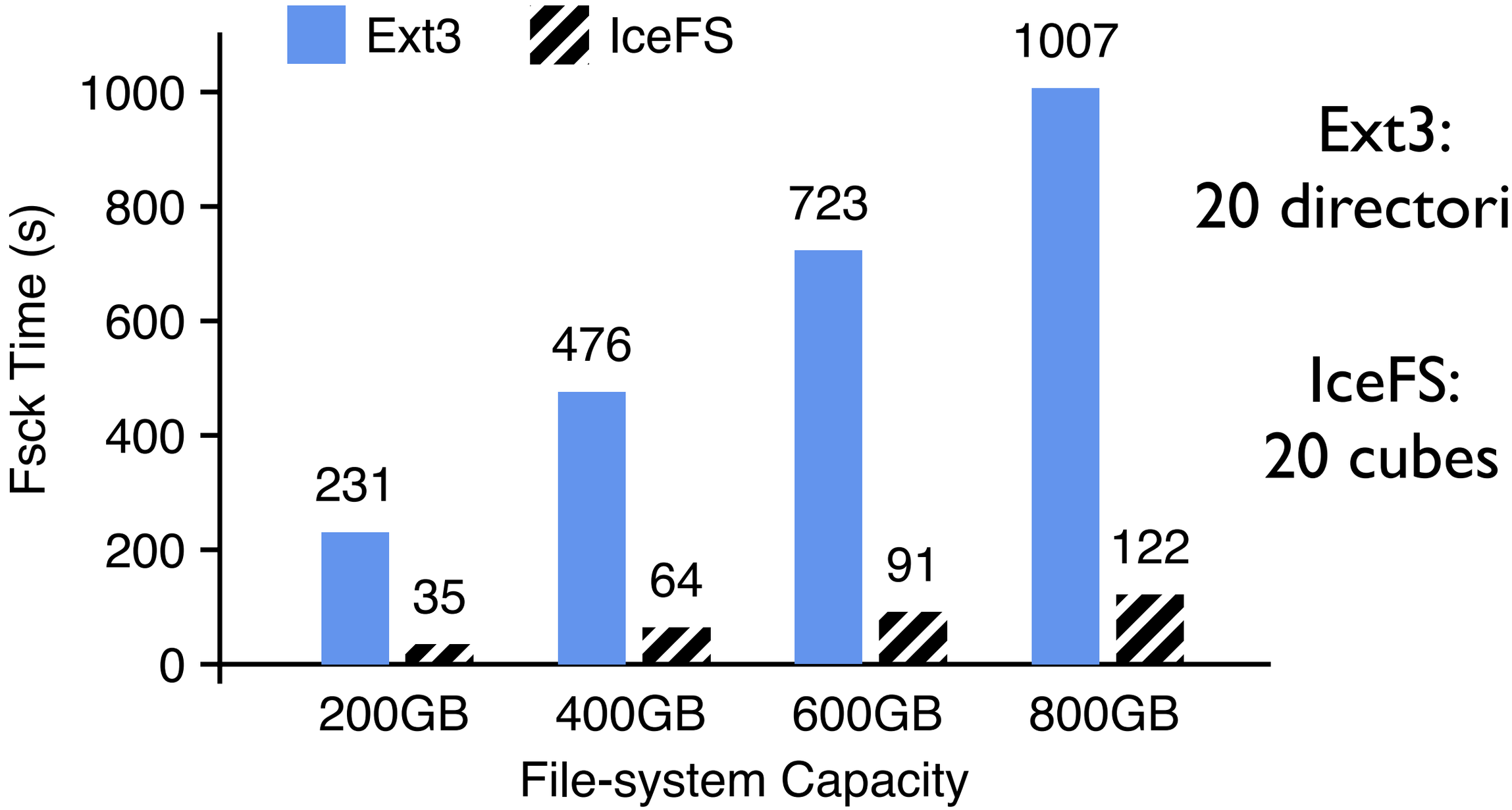
Recovery In Ext3



Fast Recovery In IceFS



Fast Recovery In IceFS



Partial recovery for a cube (up to 8x)

Evaluation

Evaluation

Does IceFS isolate failures ?

- inject around 200 faults
- per-cube failure (read-only or crash) for IceFS

Does IceFS have faster recovery ?

- independent recovery for a cube

Evaluation

Evaluation

Does IceFS isolate failures ?

- inject around 200 faults
- per-cube failure (read-only or crash) for IceFS

Does IceFS have faster recovery ?

- independent recovery for a cube

Does IceFS have better performance ?

Workloads

Workloads

SQLite

- a database application
- sequentially write large key/value pairs
- asynchronous

Workloads

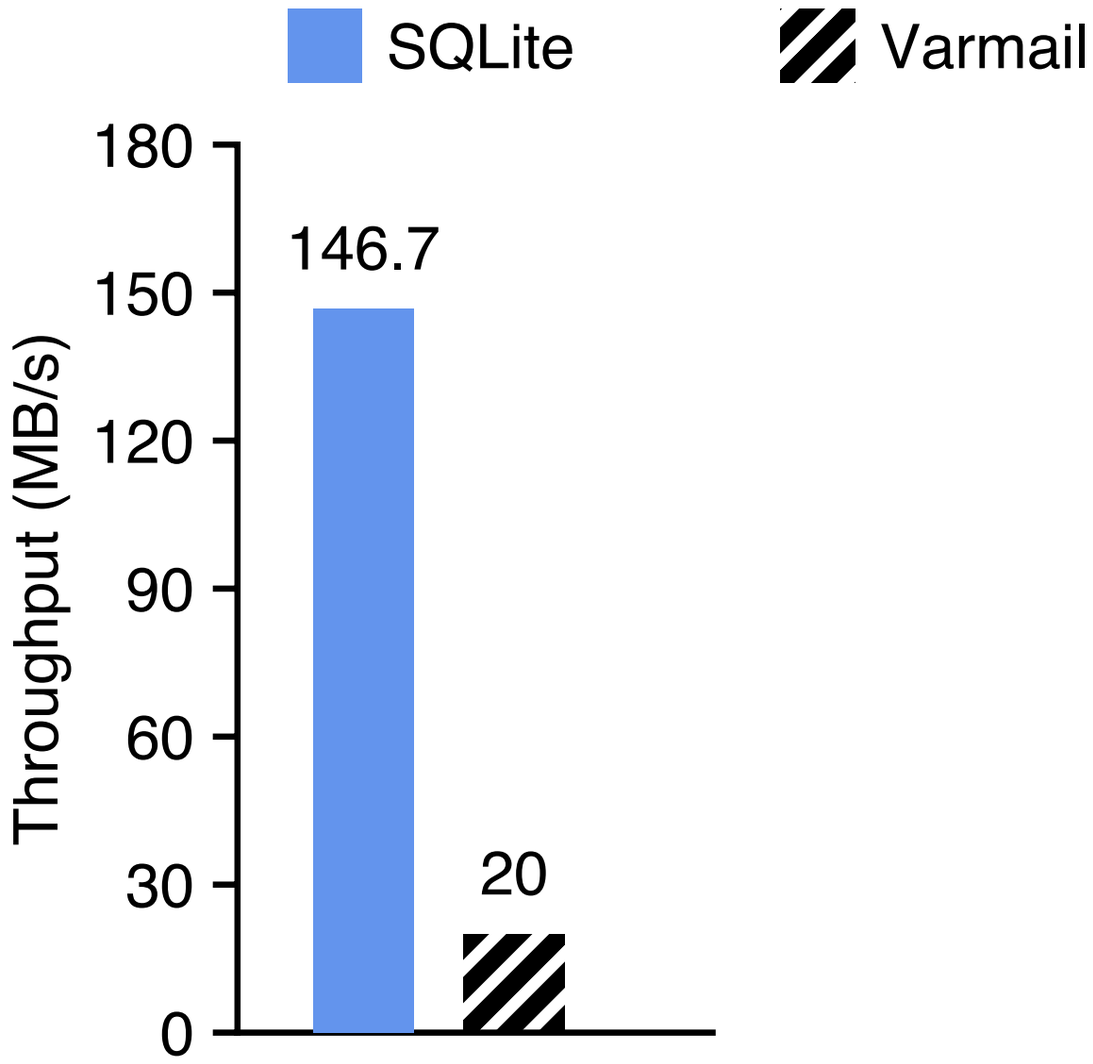
SQLite

- a database application
- sequentially write large key/value pairs
- asynchronous

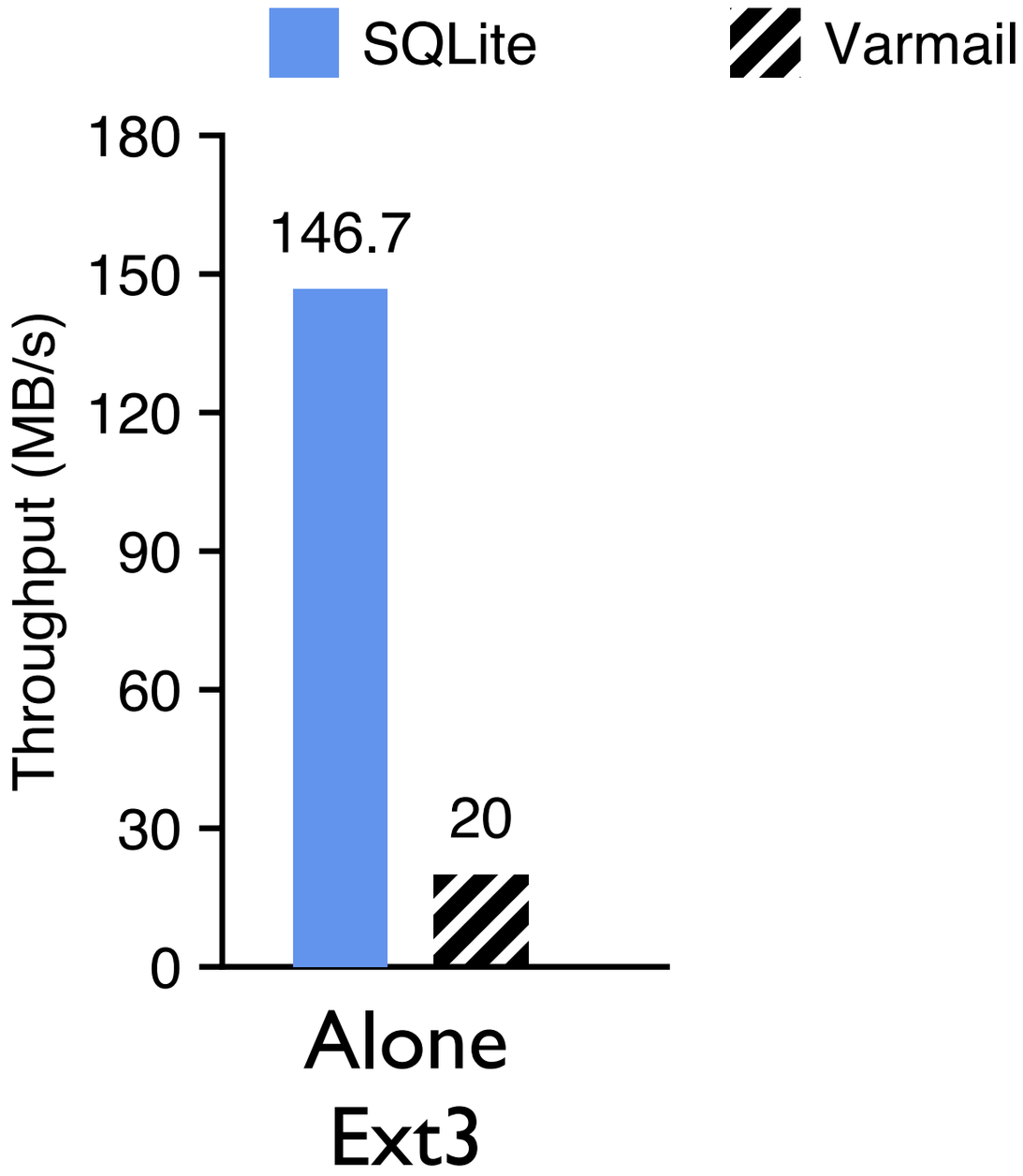
Varmail

- an email server workload
- randomly write small blocks
- fsync after each write

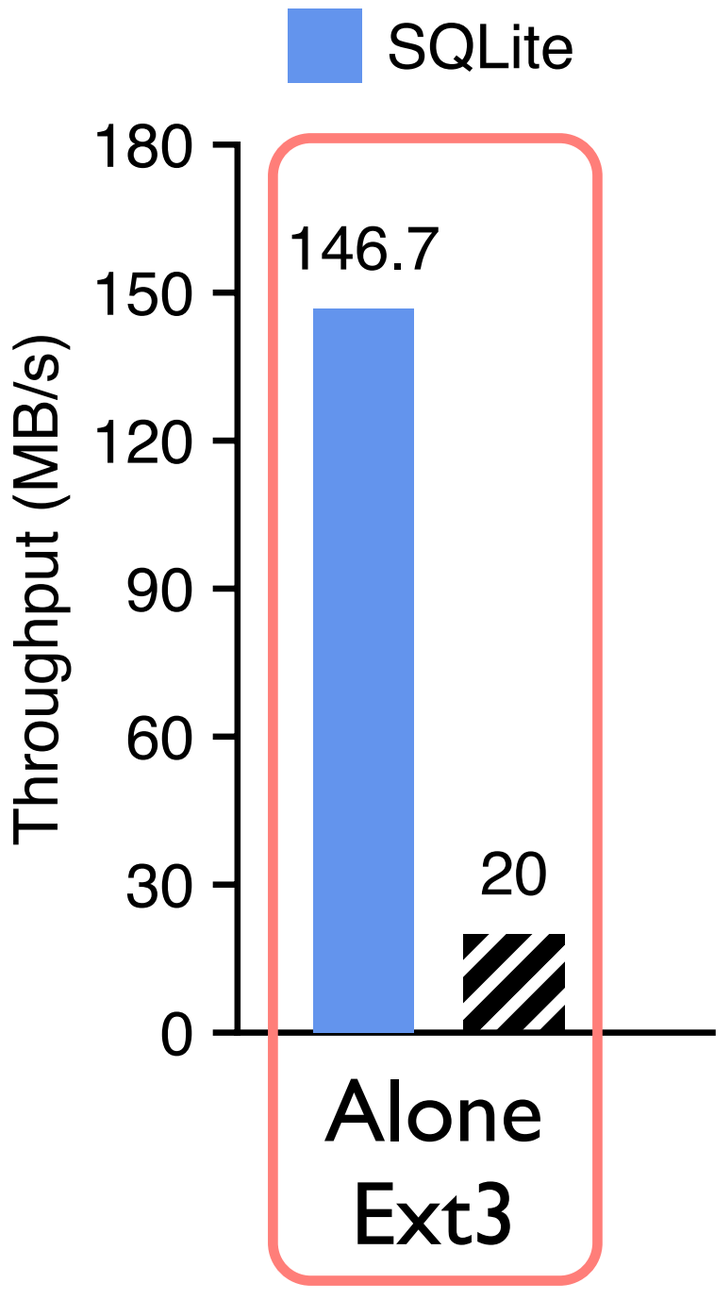
Ext3 Journaling



Ext3 Journaling

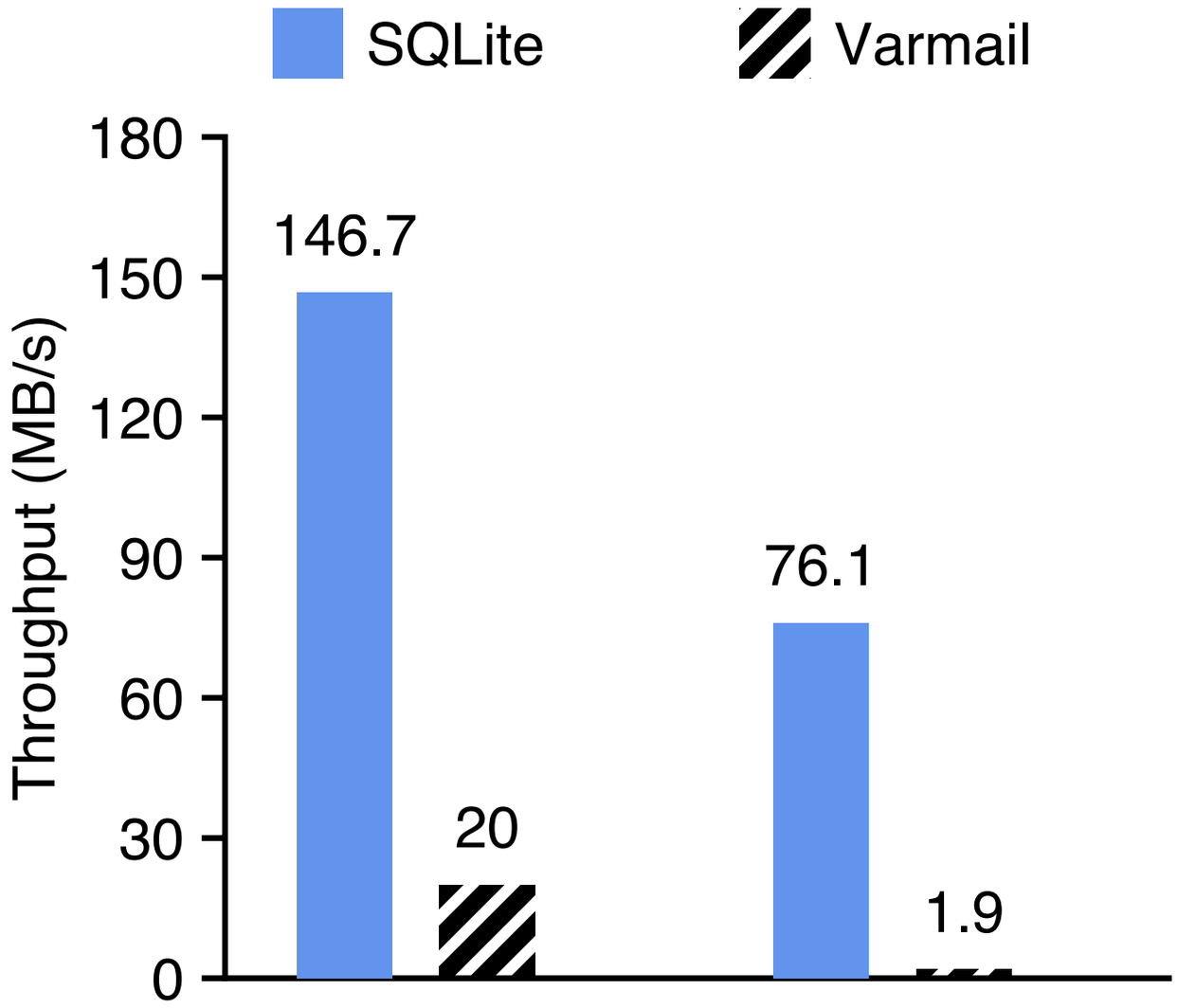


Ext3 Journaling

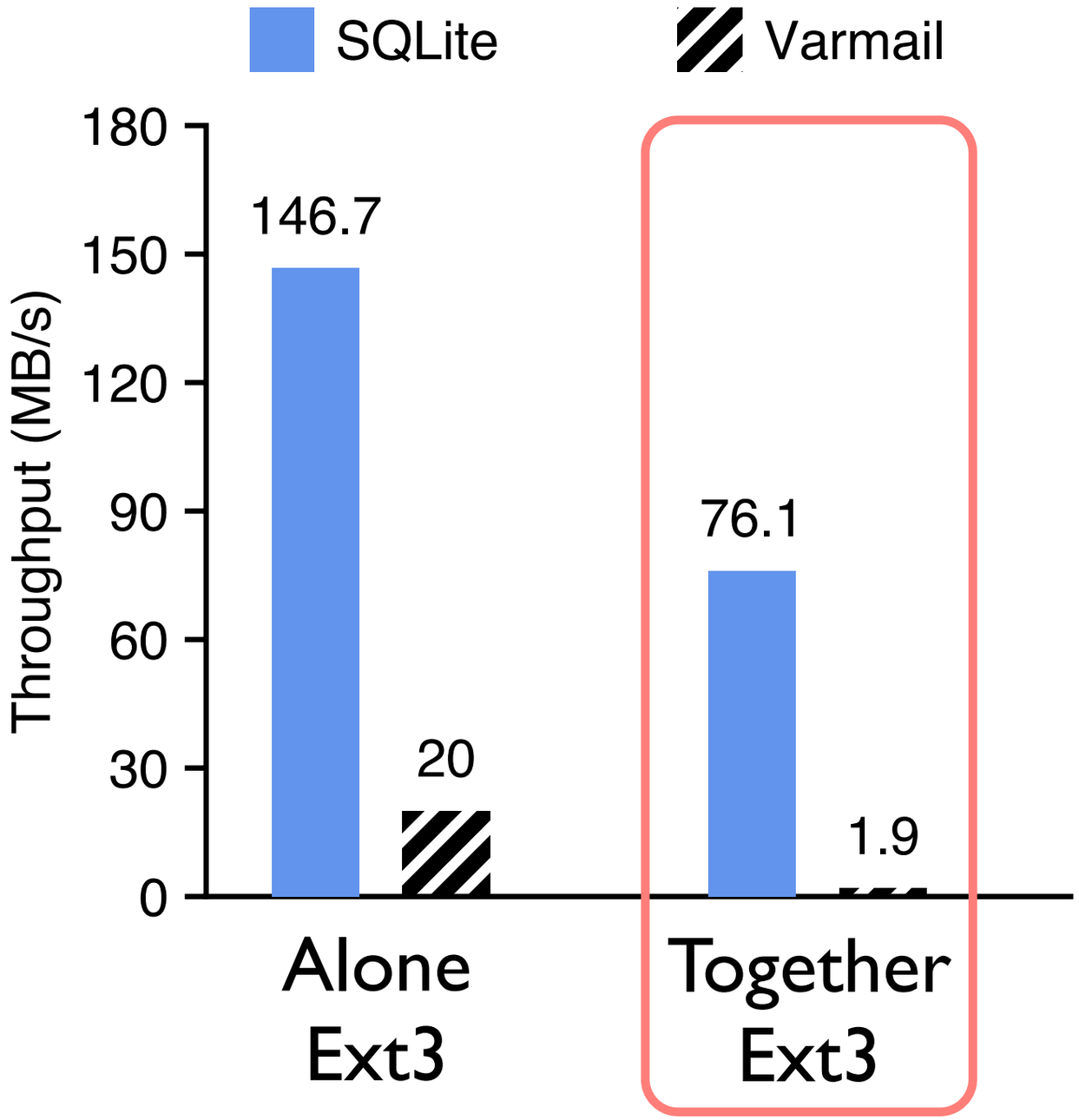


Ext3 runs with 2 directories

Ext3 Journaling

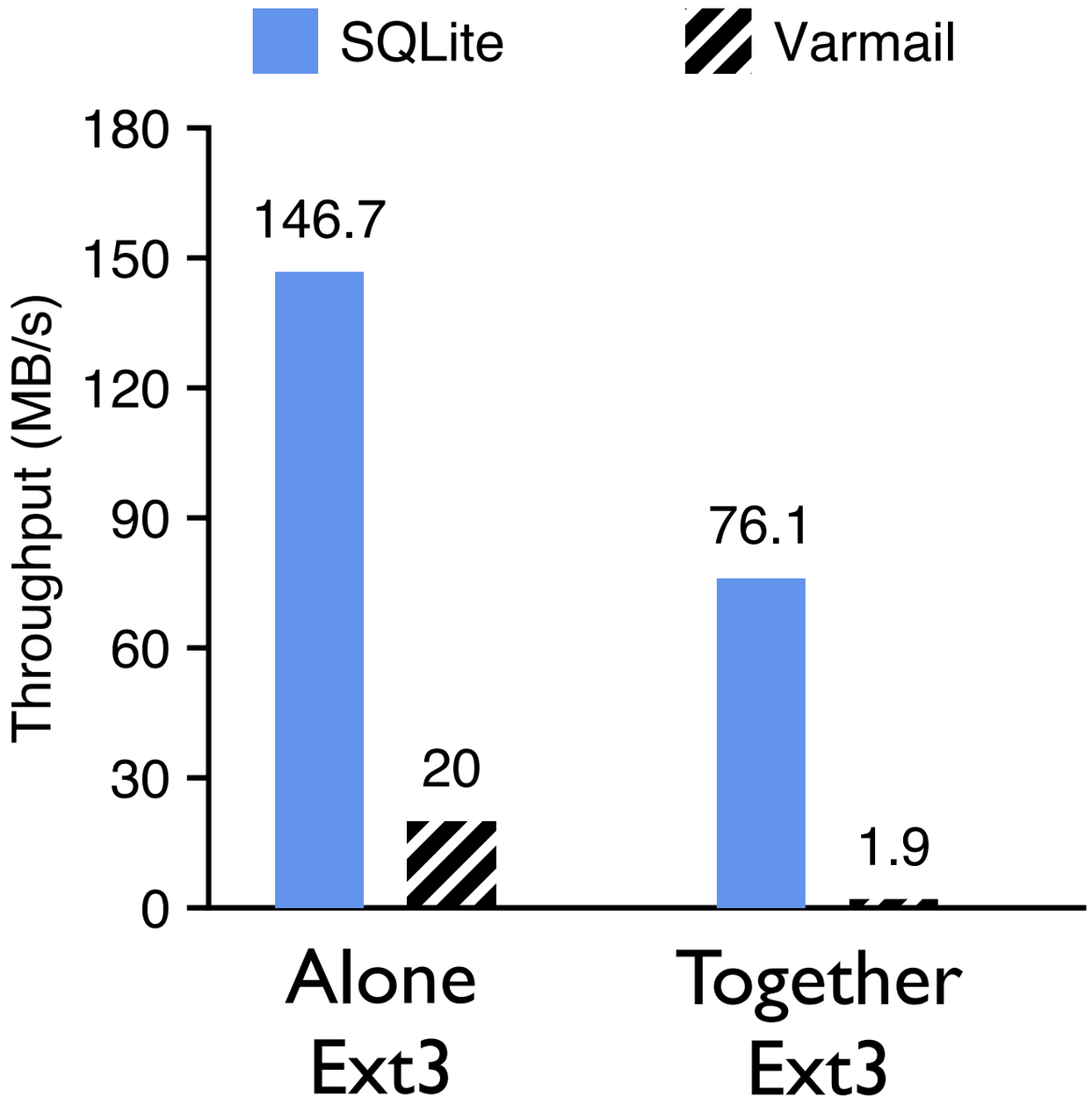


Ext3 Journaling



Ext3 runs with 2 directories

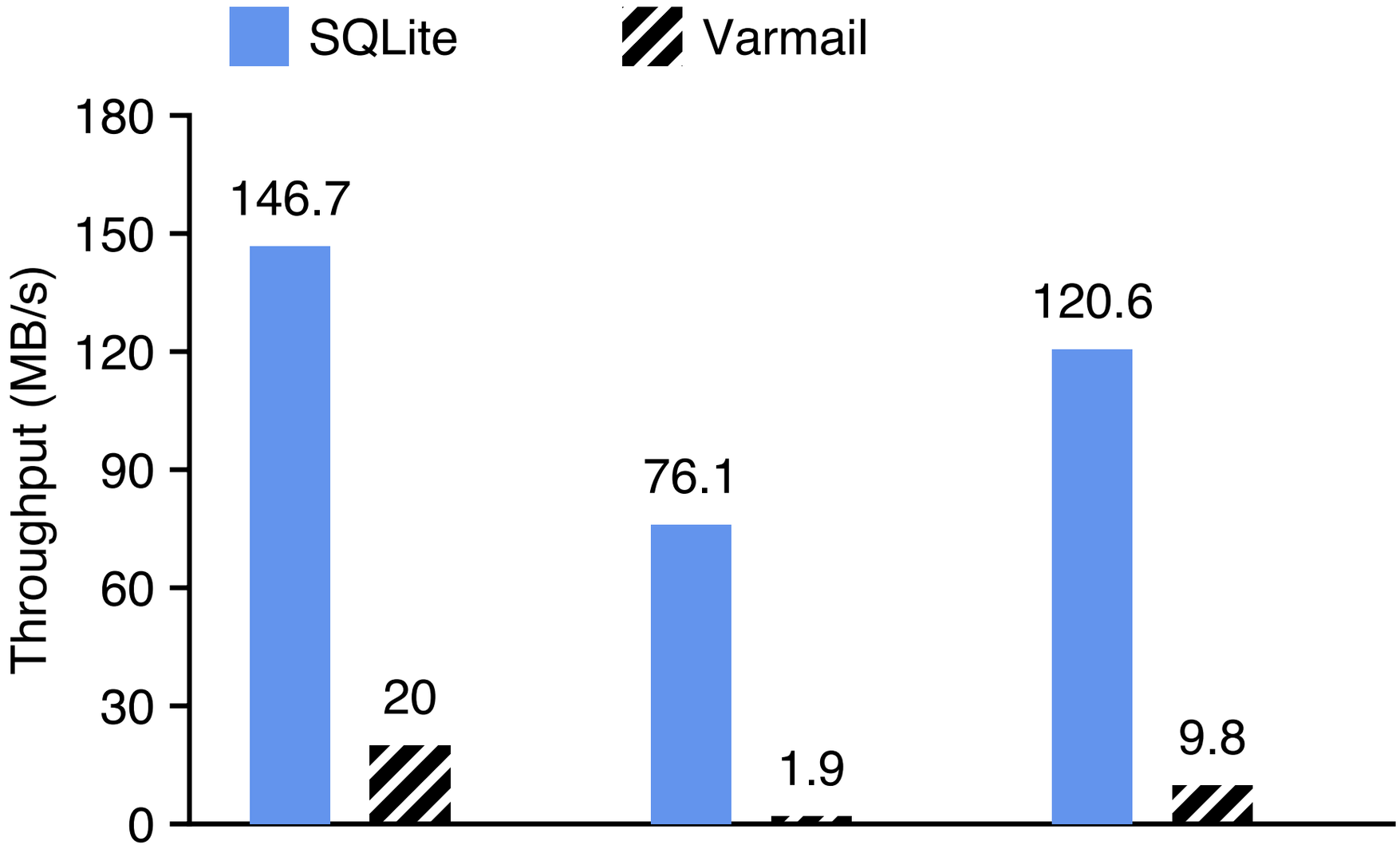
Ext3 Journaling



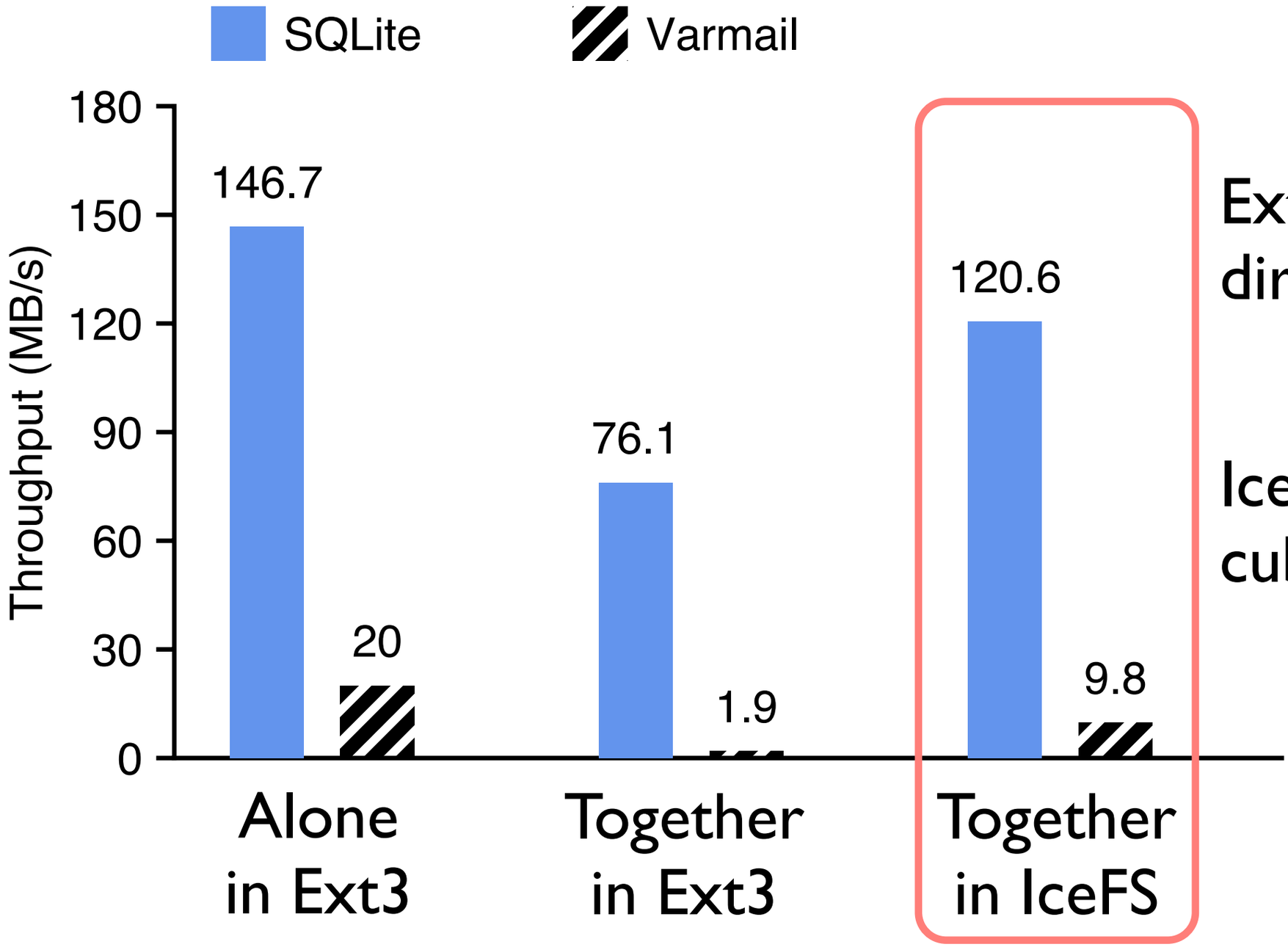
Ext3 runs with 2 directories

Shared transactions hurt performance (over 10x)

Isolated Journaling In IceFS



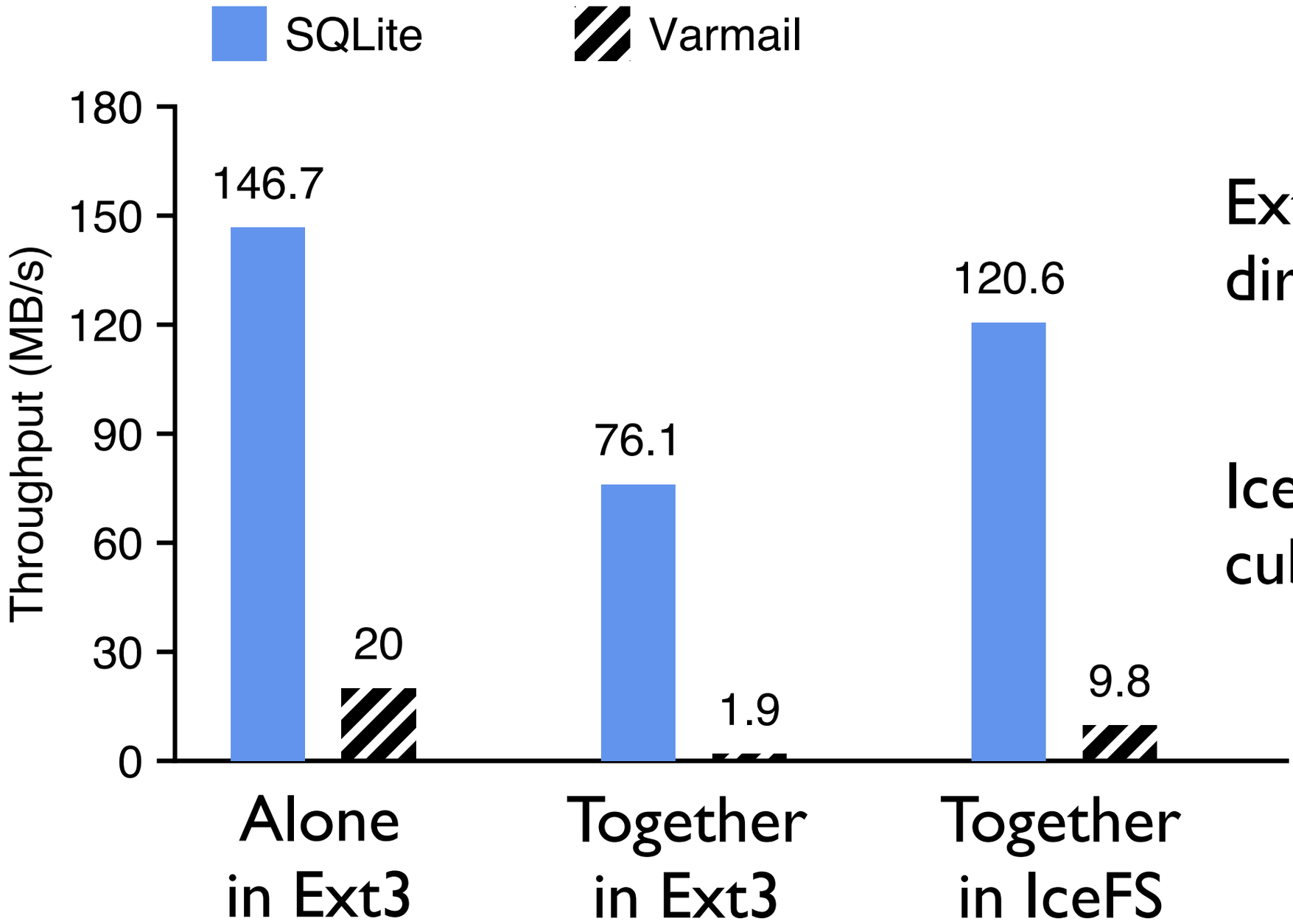
Isolated Journaling In IceFS



Ext3 runs with 2 directories

IceFS runs with 2 cubes

Isolated Journaling In IceFS

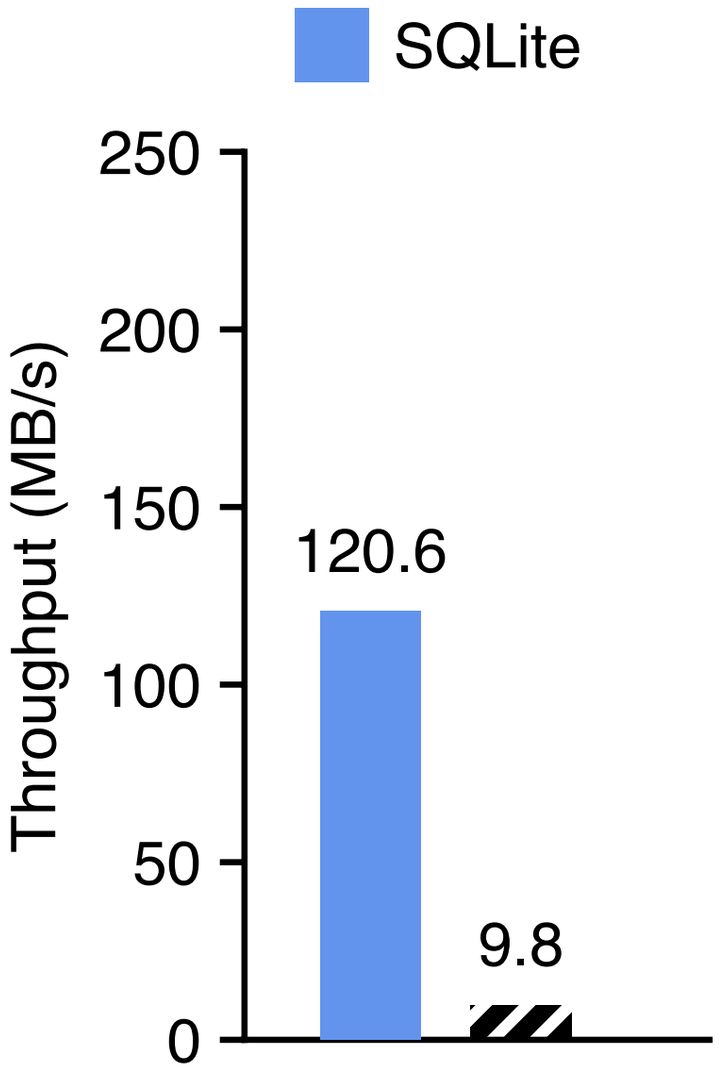


Ext3 runs with 2 directories

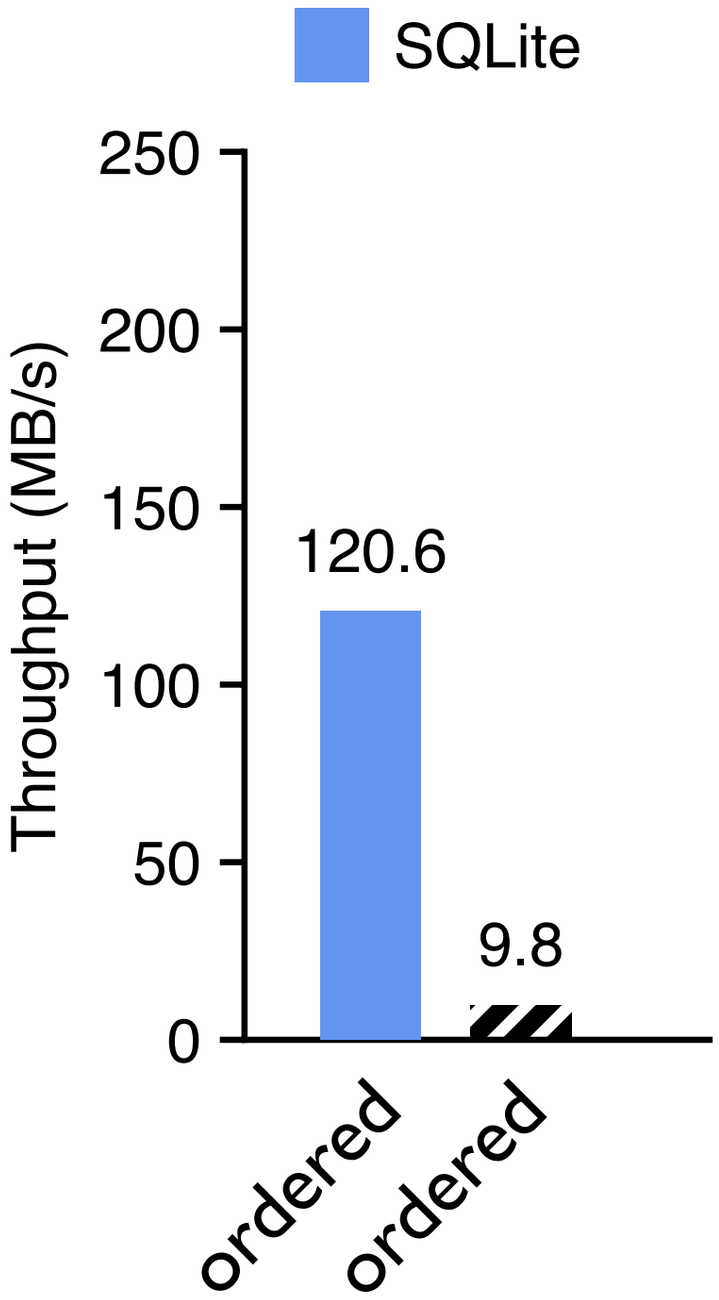
IceFS runs with 2 cubes

Parallel transactions in IceFS provide isolated performance (over 5x)

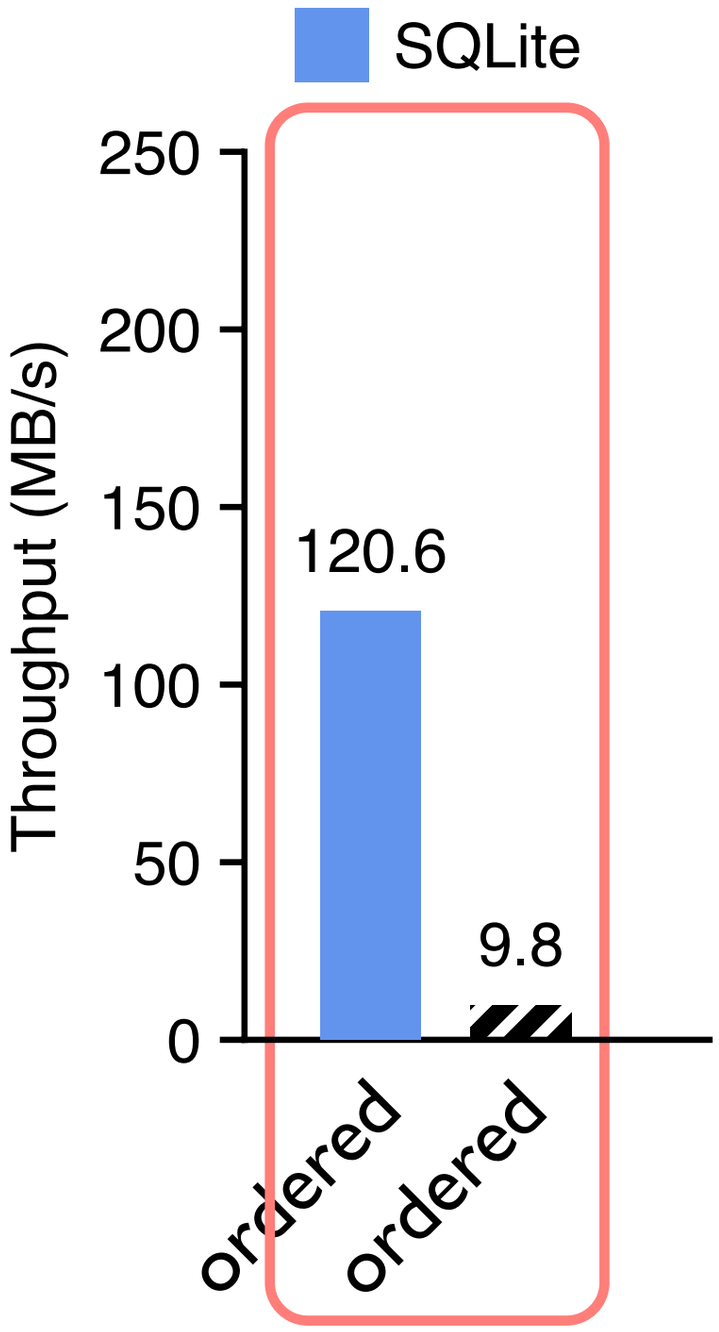
Specialized Journaling In IceFS



Specialized Journaling In IceFS

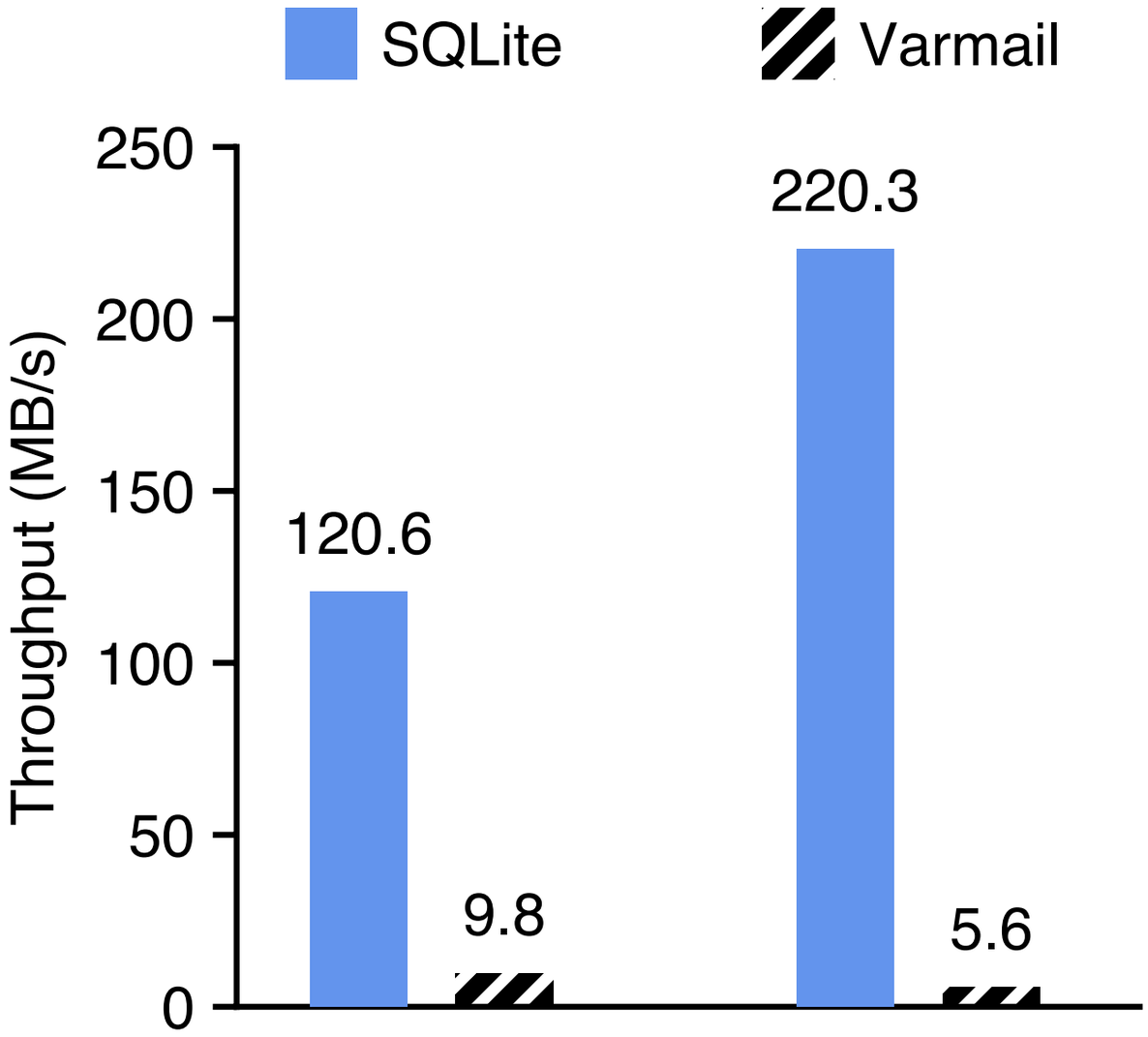


Specialized Journaling In IceFS

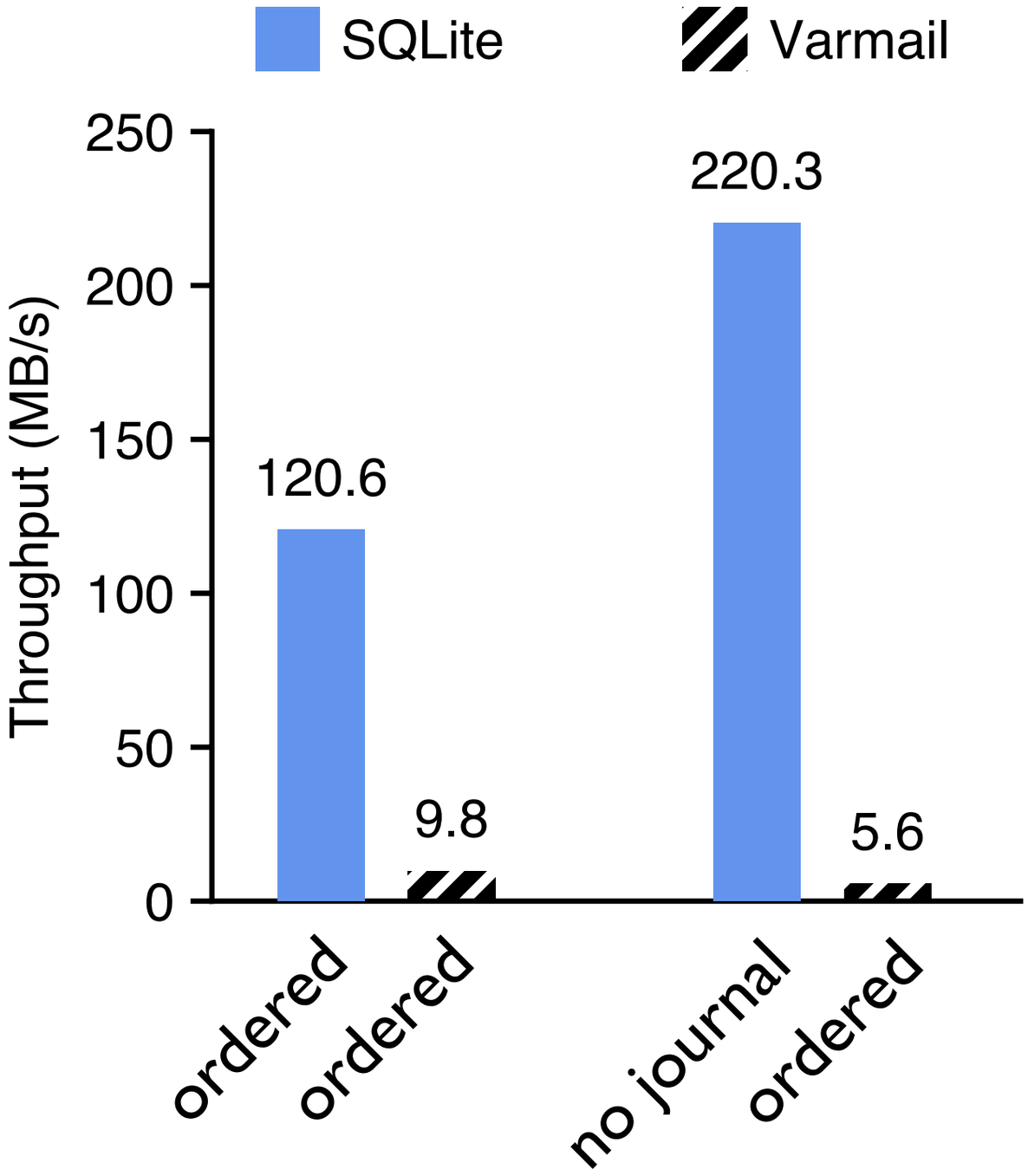


Both cubes use ordered mode

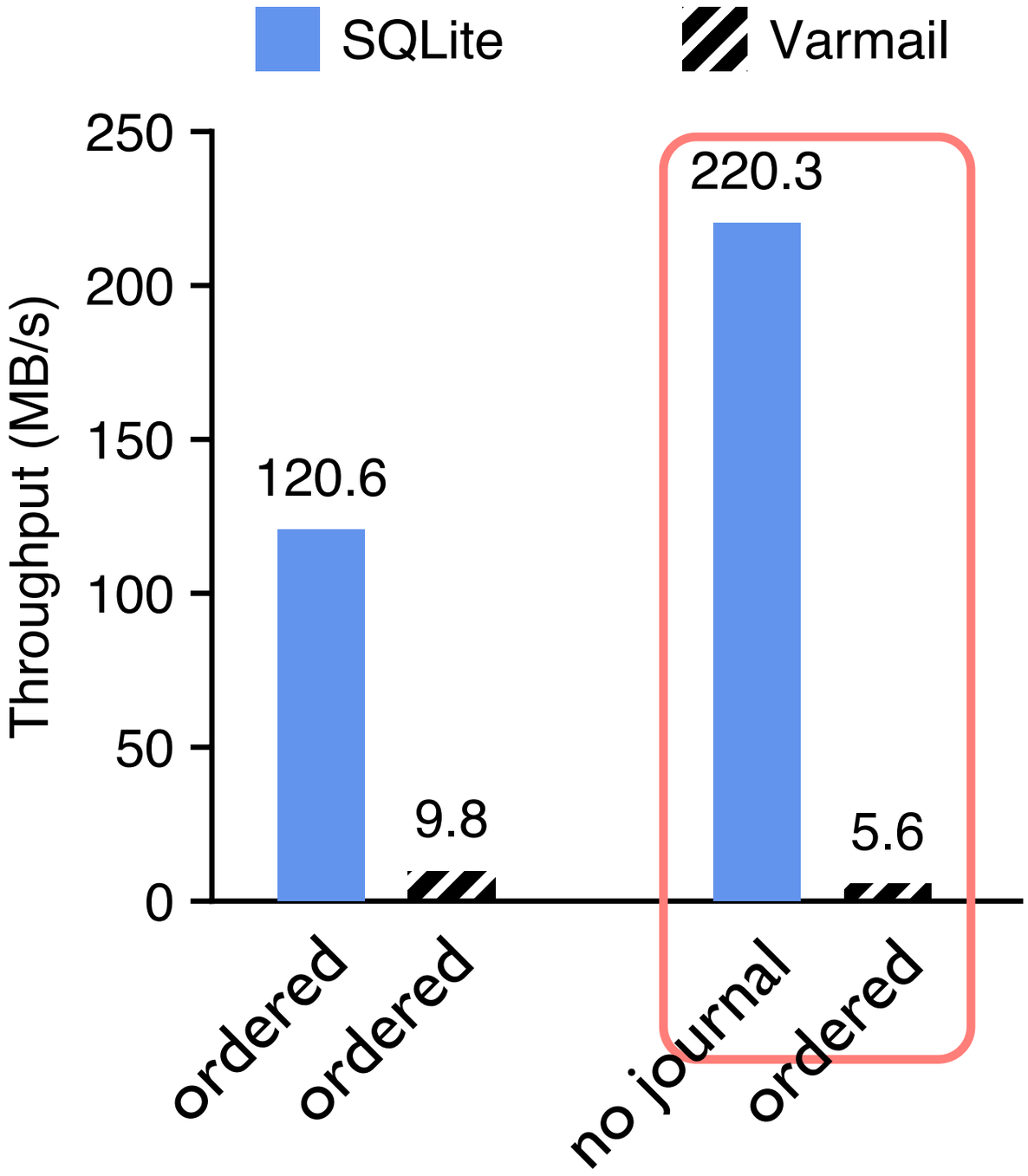
Specialized Journaling In IceFS



Specialized Journaling In IceFS

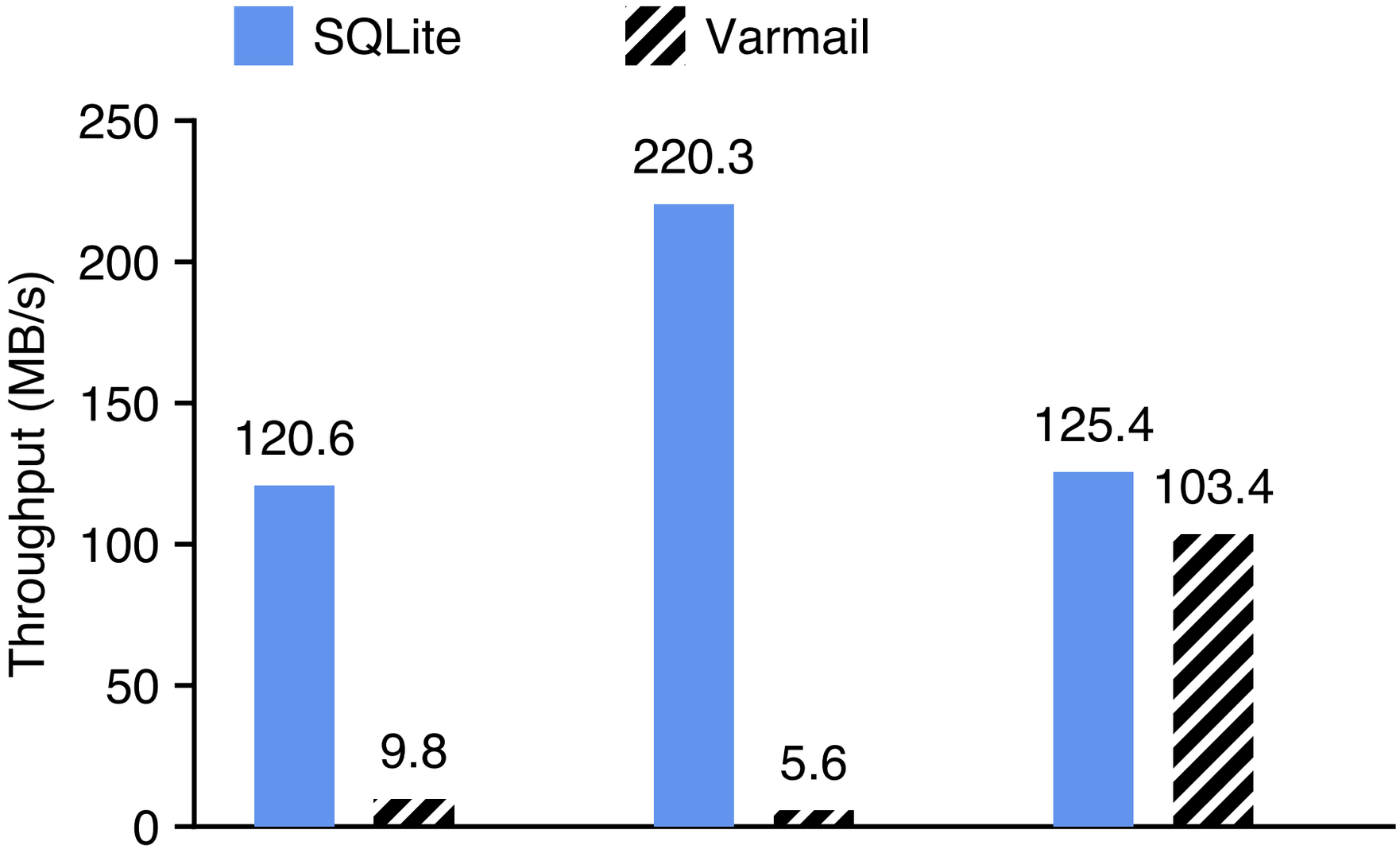


Specialized Journaling In IceFS

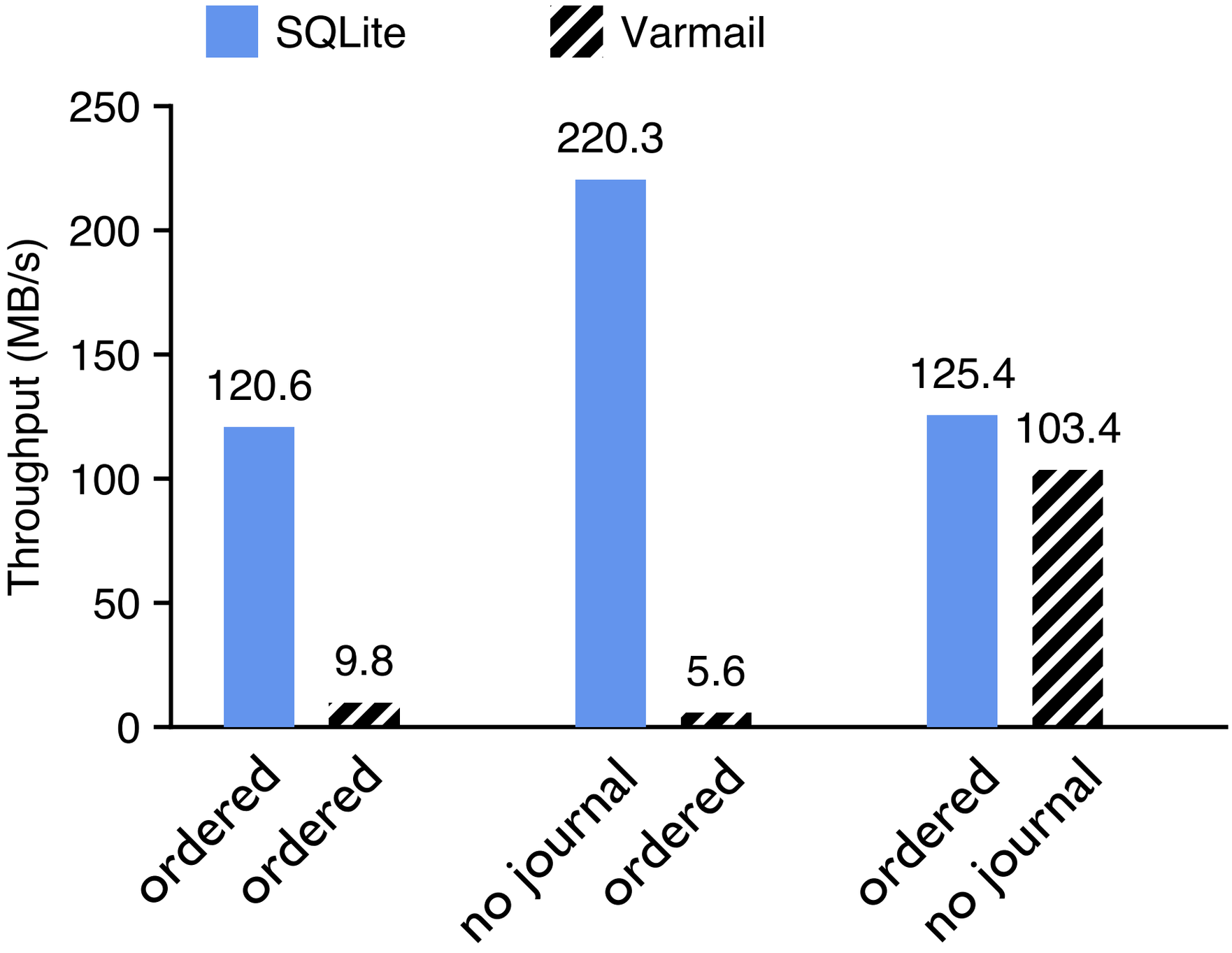


SQLite runs with no journal
Varmail runs with ordered

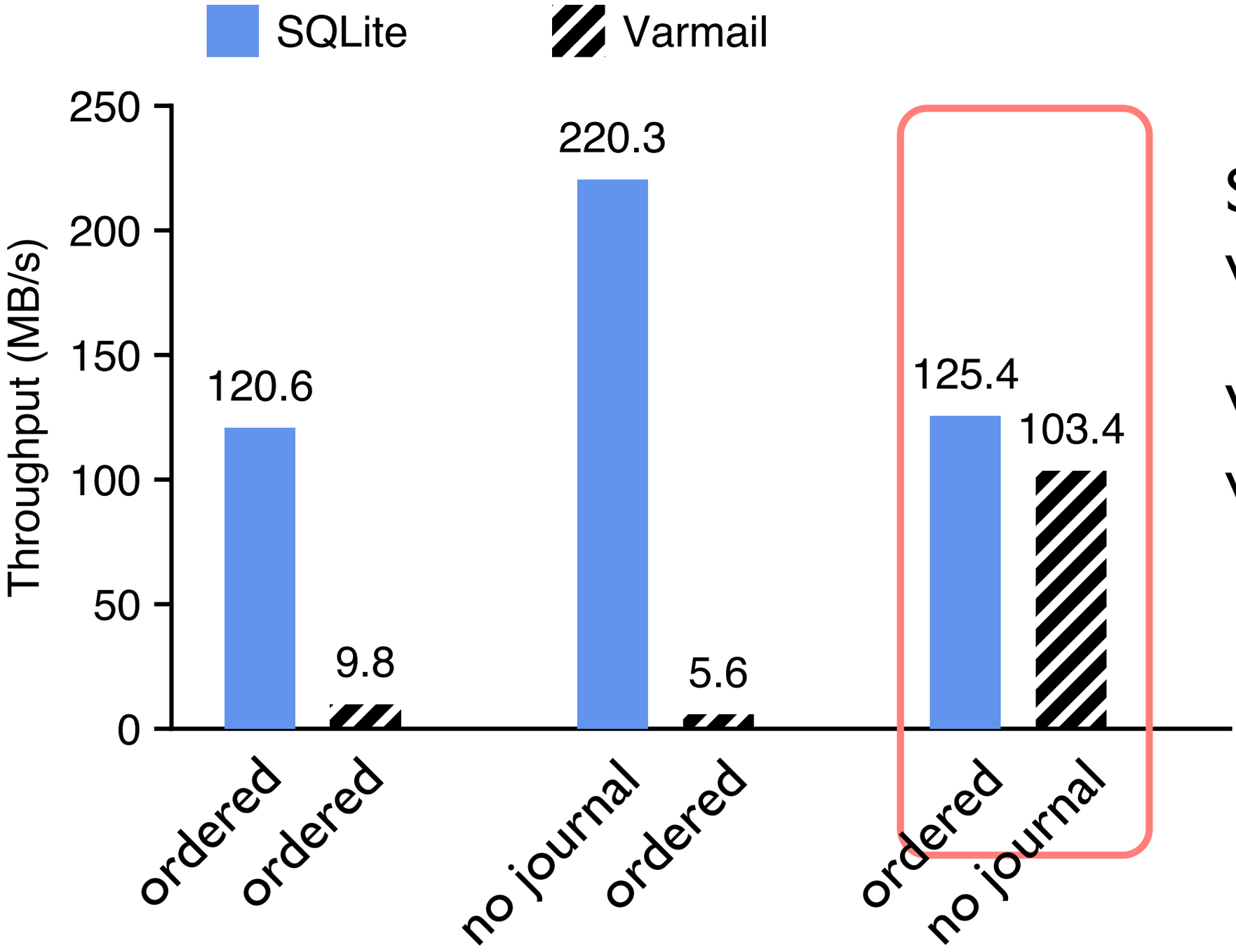
Specialized Journaling In IceFS



Specialized Journaling In IceFS



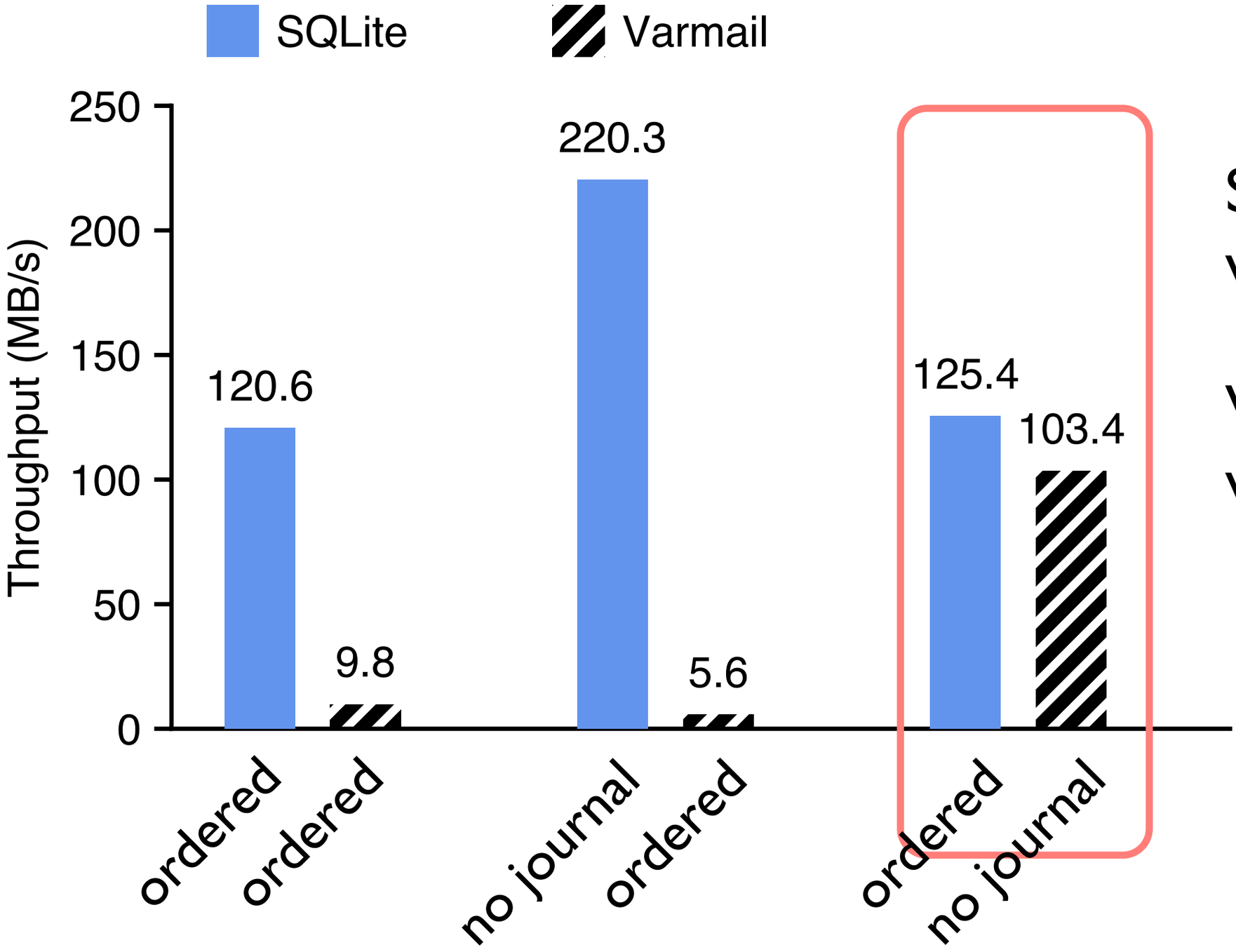
Specialized Journaling In IceFS



SQLite runs with ordered

Varmail runs with no journal

Specialized Journaling In IceFS



SQLite runs with ordered

Varmail runs with no journal

Specialized journaling in IceFS provide flexibility between consistency and performance (over 50x)

Evaluation

Evaluation

Isolate failures ?

- inject around 200 faults
- per-cube failure (read-only or crash) for IceFS

Faster recovery ?

- independent recovery for a cube

Better journaling performance ?

- isolated journaling performance
- flexibility between consistency and performance

Evaluation

Evaluation

Isolate failures ?

- inject around 200 faults
- per-cube failure (read-only or crash) for IceFS

Faster recovery ?

- independent recovery for a cube

Better journaling performance ?

- isolated journaling performance
- flexibility between consistency and performance

Useful for applications ?

Server Virtualization

Server Virtualization



Server Virtualization

vm1 vm2 vm3



Shared file system

Server Virtualization

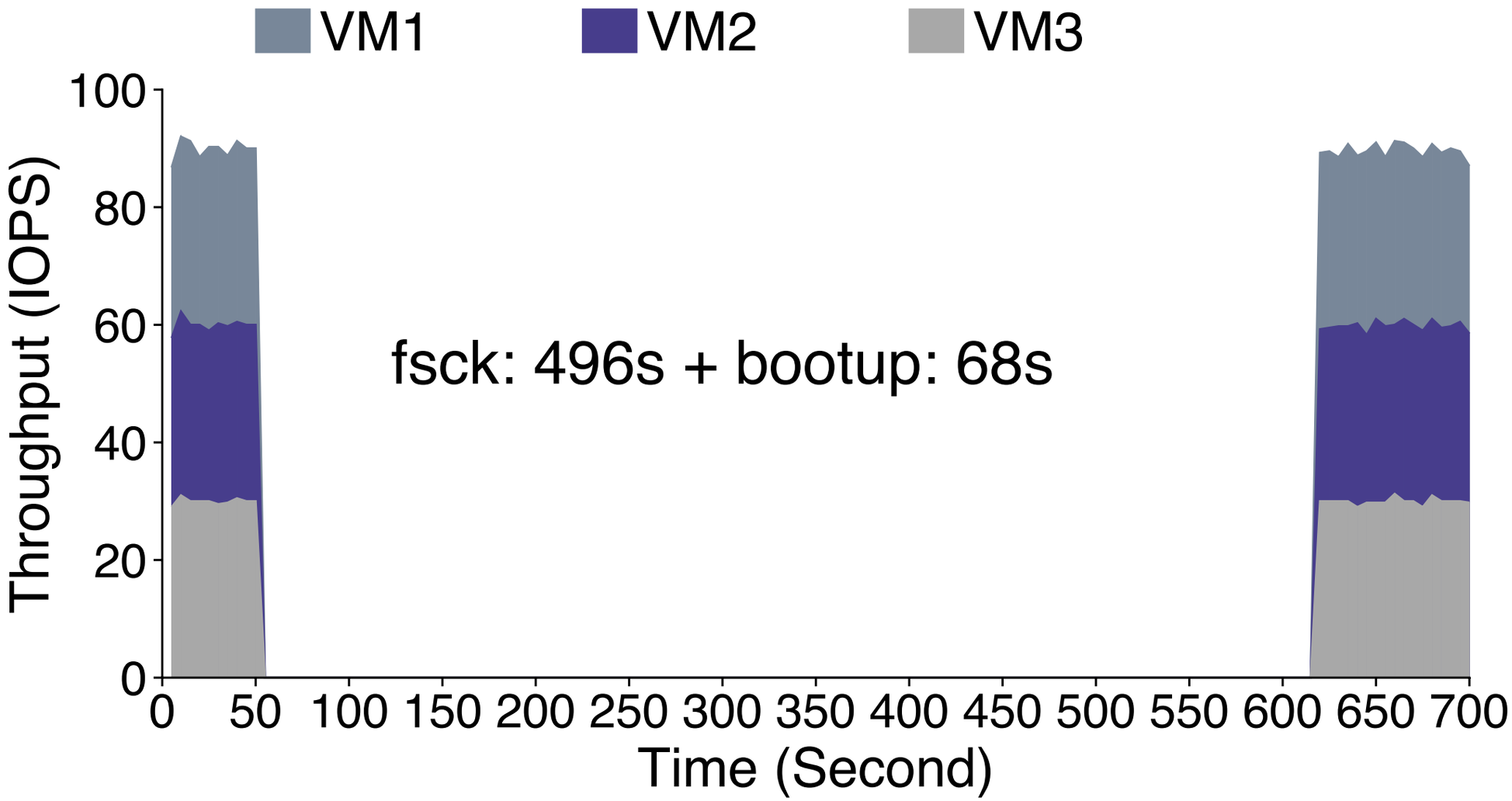
vm1 vm2 vm3



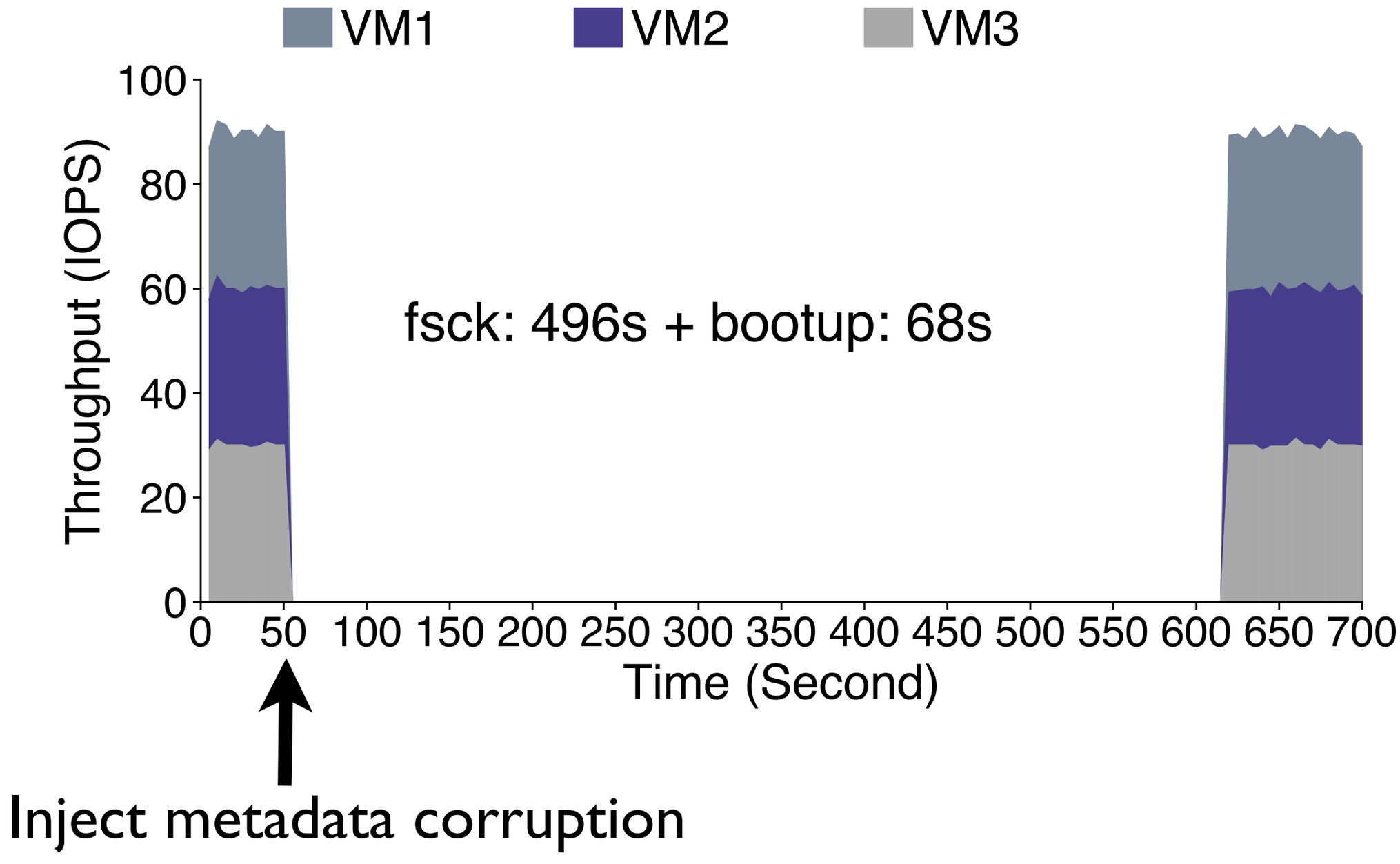
Shared file system

Failures and recovery of the shared file system impact all virtual machines

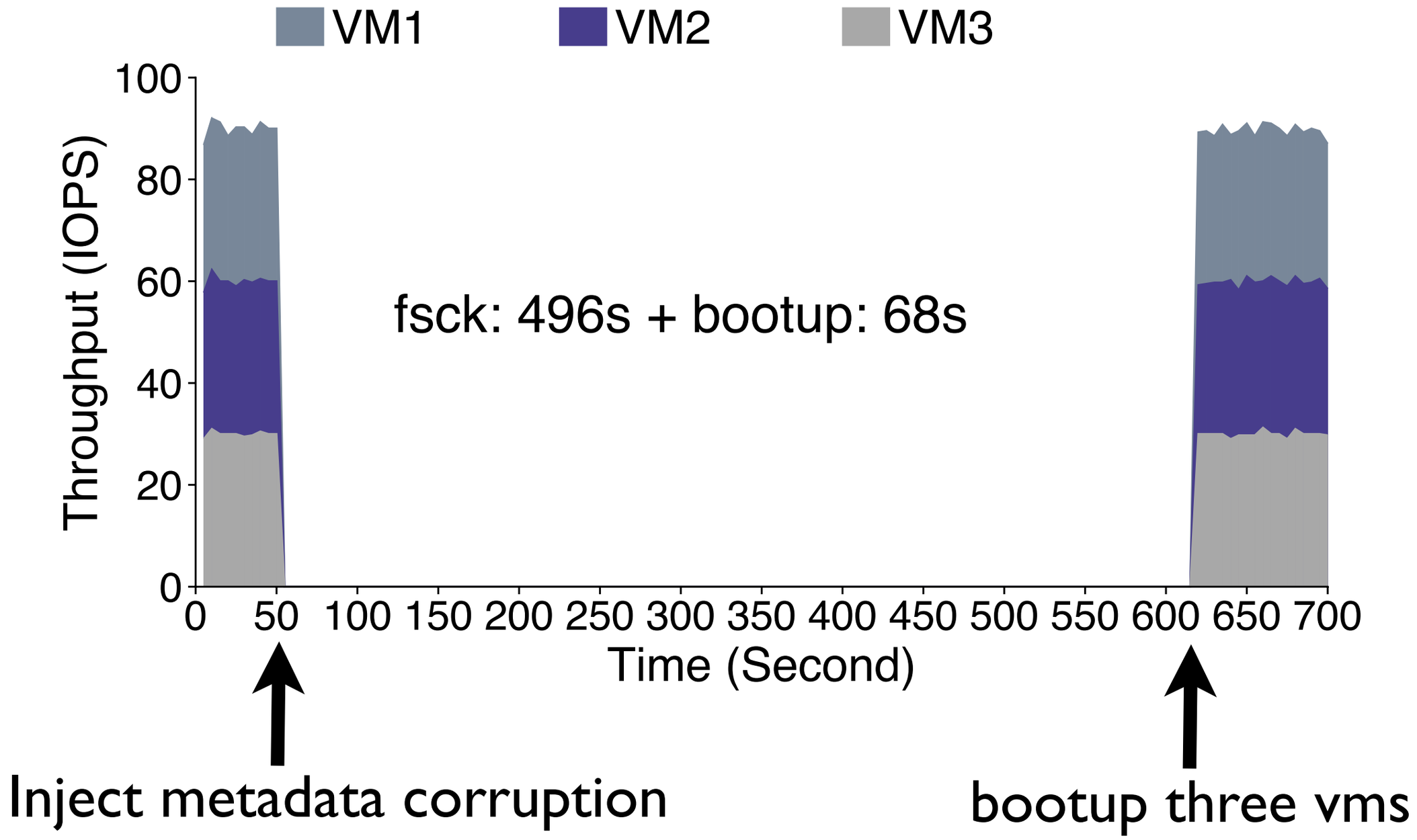
Virtual Machines



Virtual Machines



Virtual Machines



Server Virtualization with IceFS

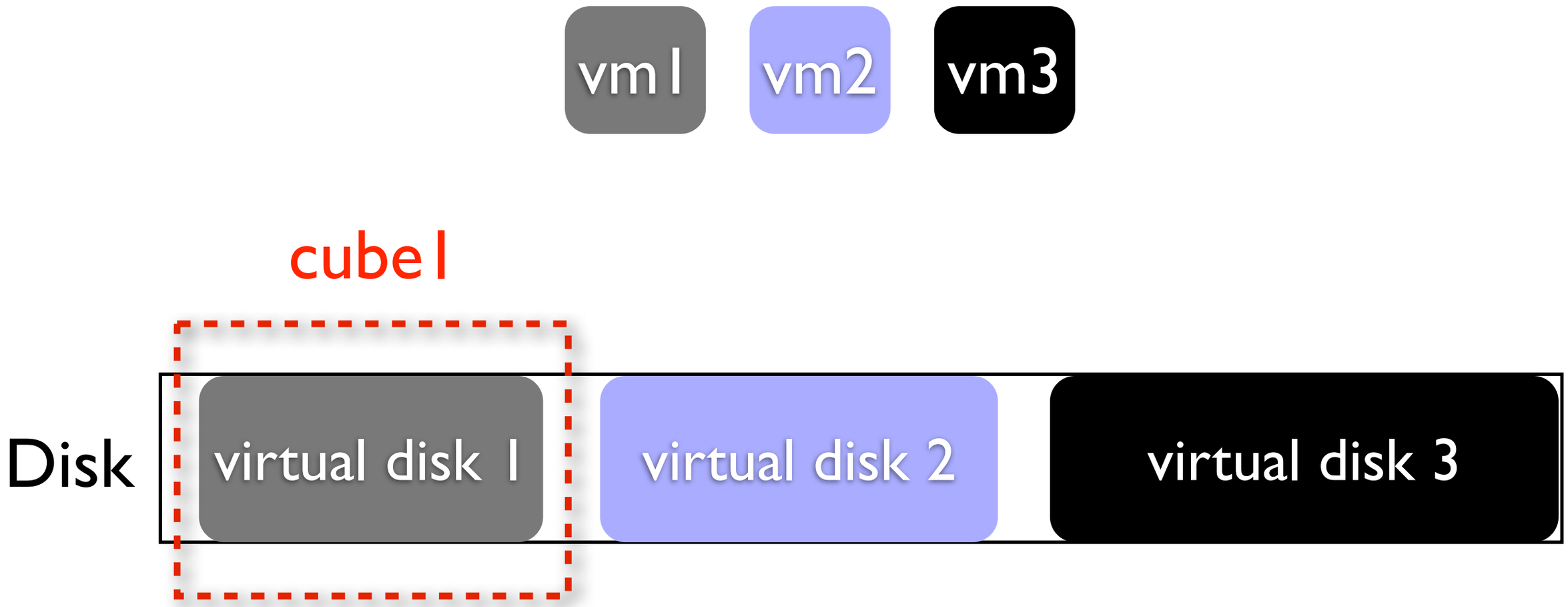
Server Virtualization with IceFS





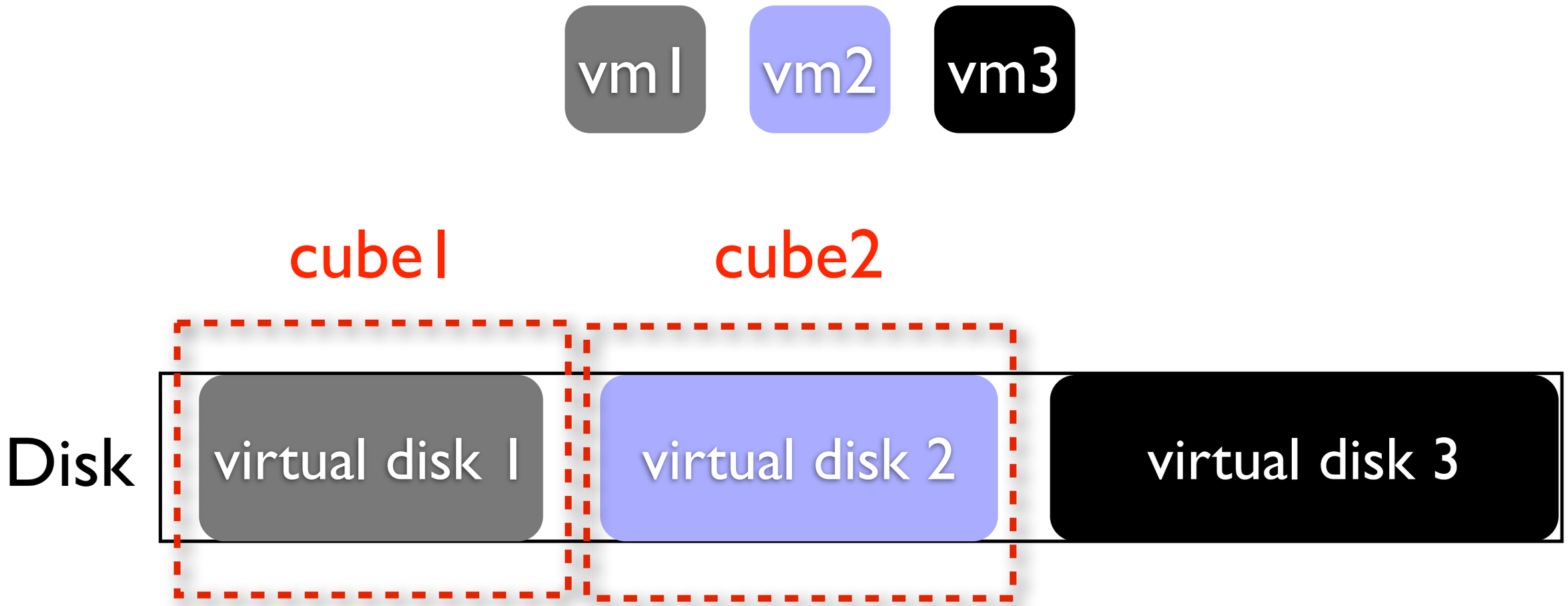
Shared file system with cubes

Server Virtualization with IceFS



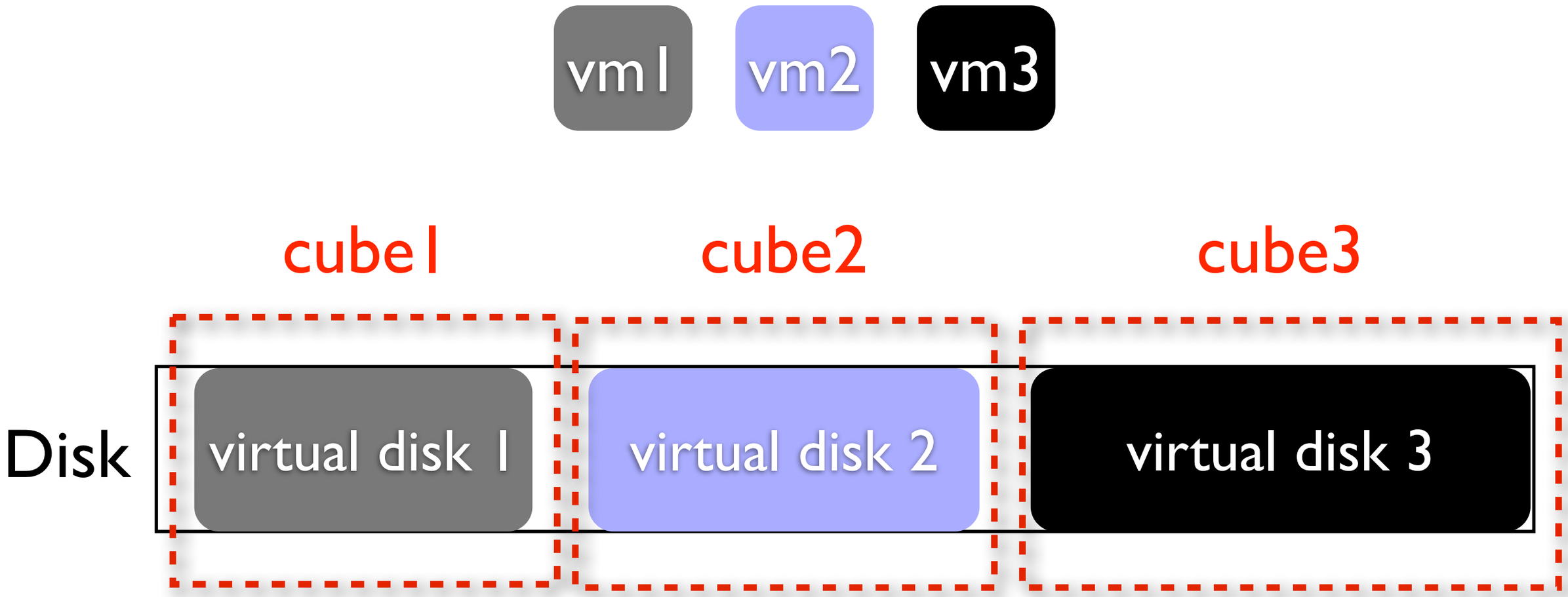
Shared file system with cubes

Server Virtualization with IceFS



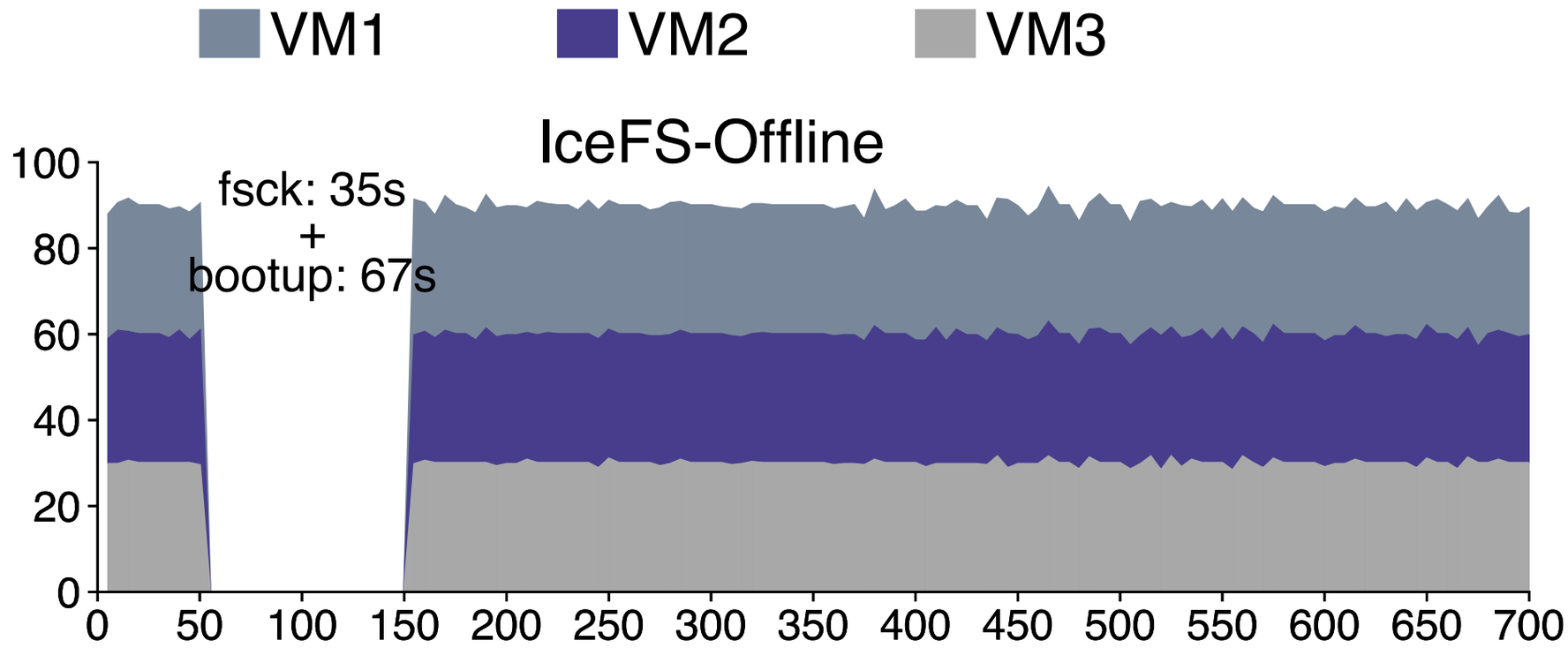
Shared file system with cubes

Server Virtualization with IceFS

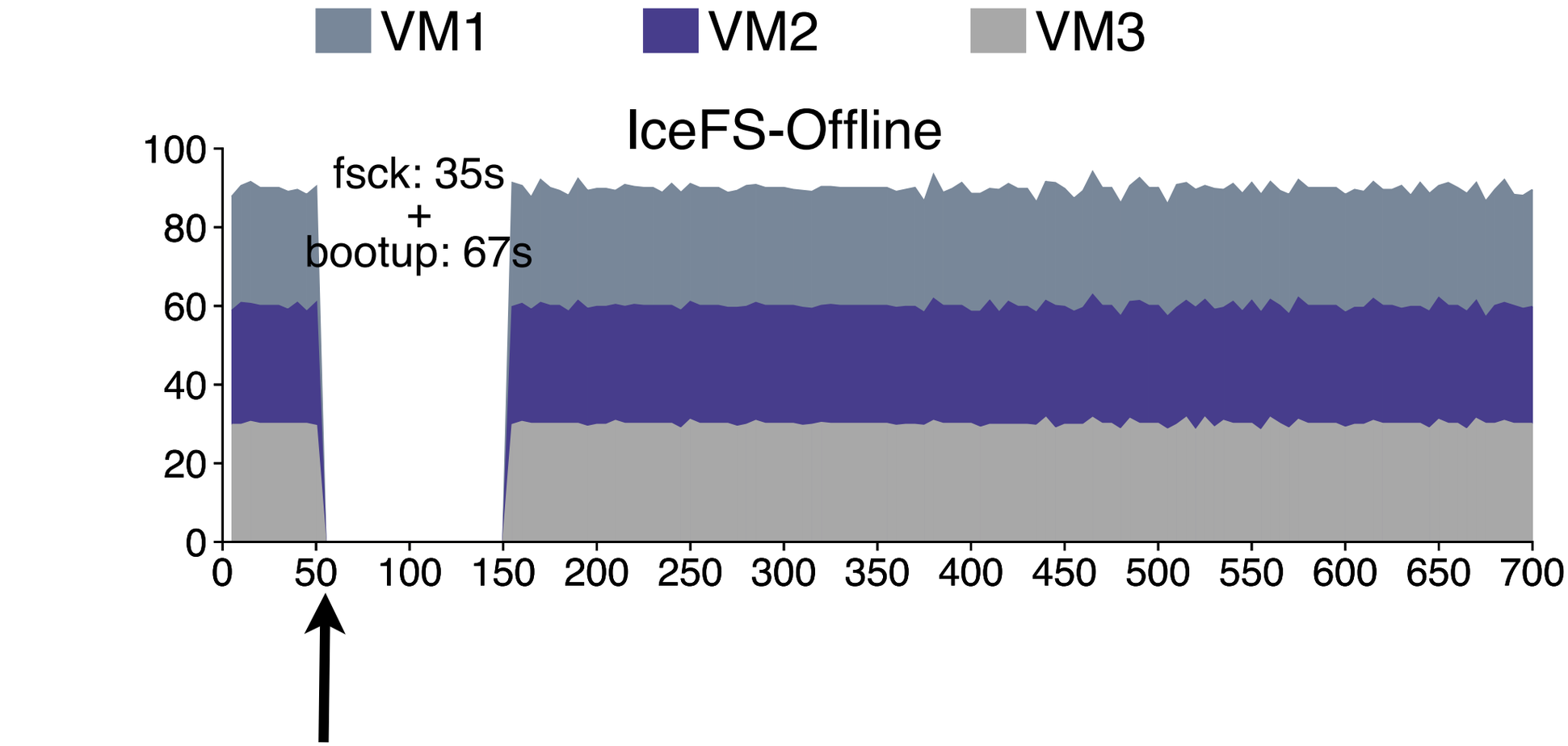


Shared file system with cubes

Server Virtualization with IceFS



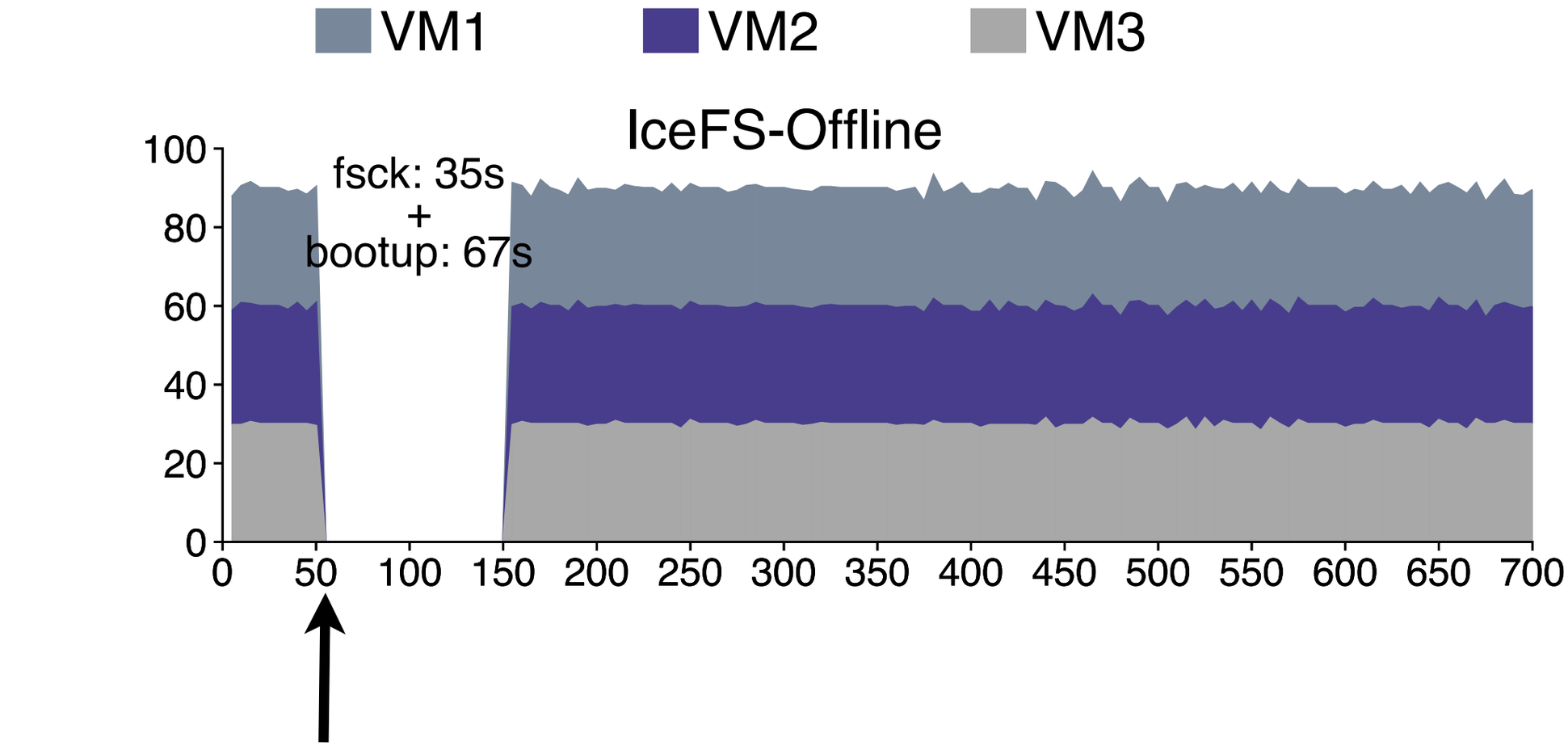
Server Virtualization with IceFS



Inject metadata corruption

Server Virtualization with IceFS

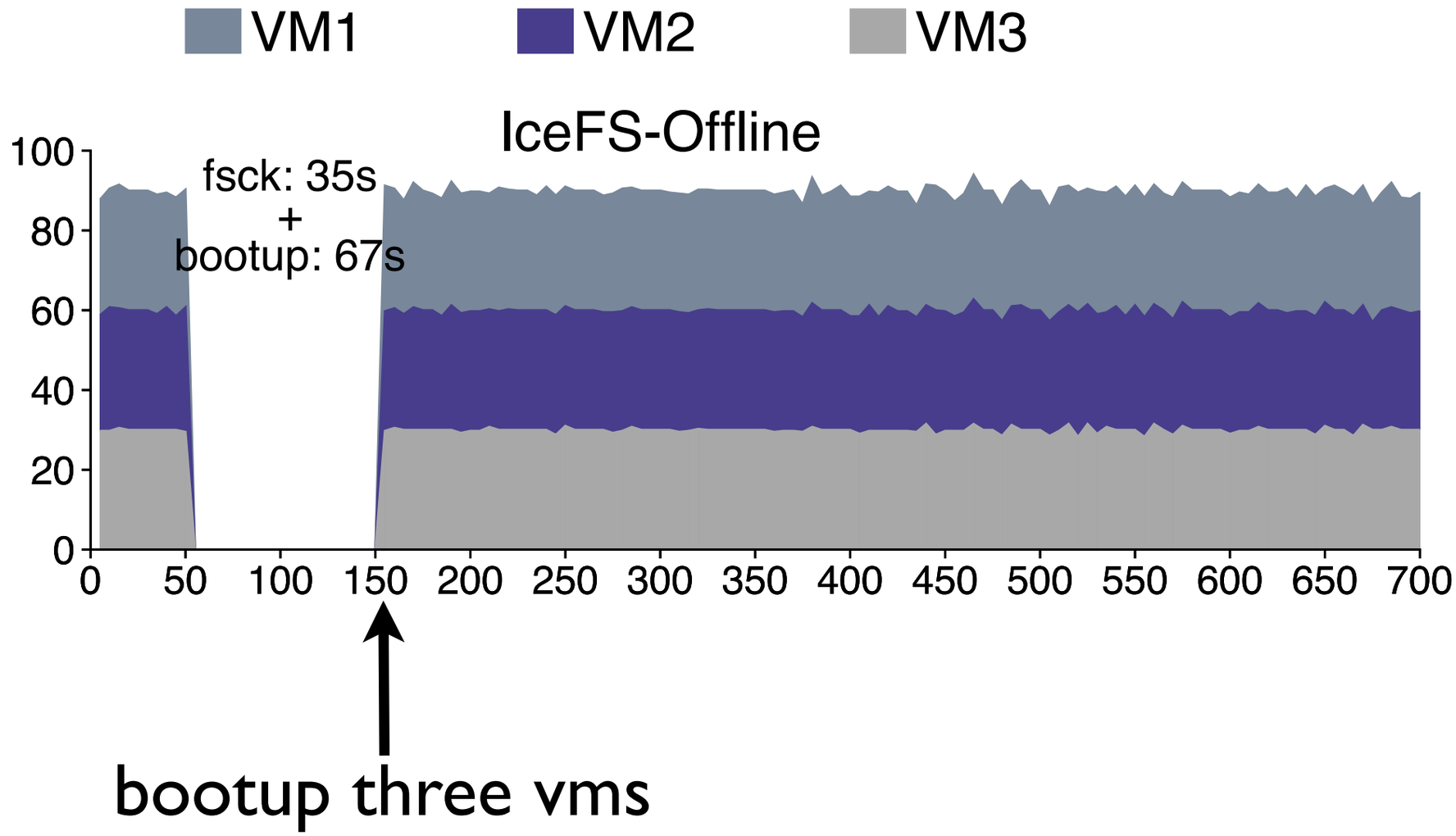
recover
a cube
offline



Inject metadata corruption

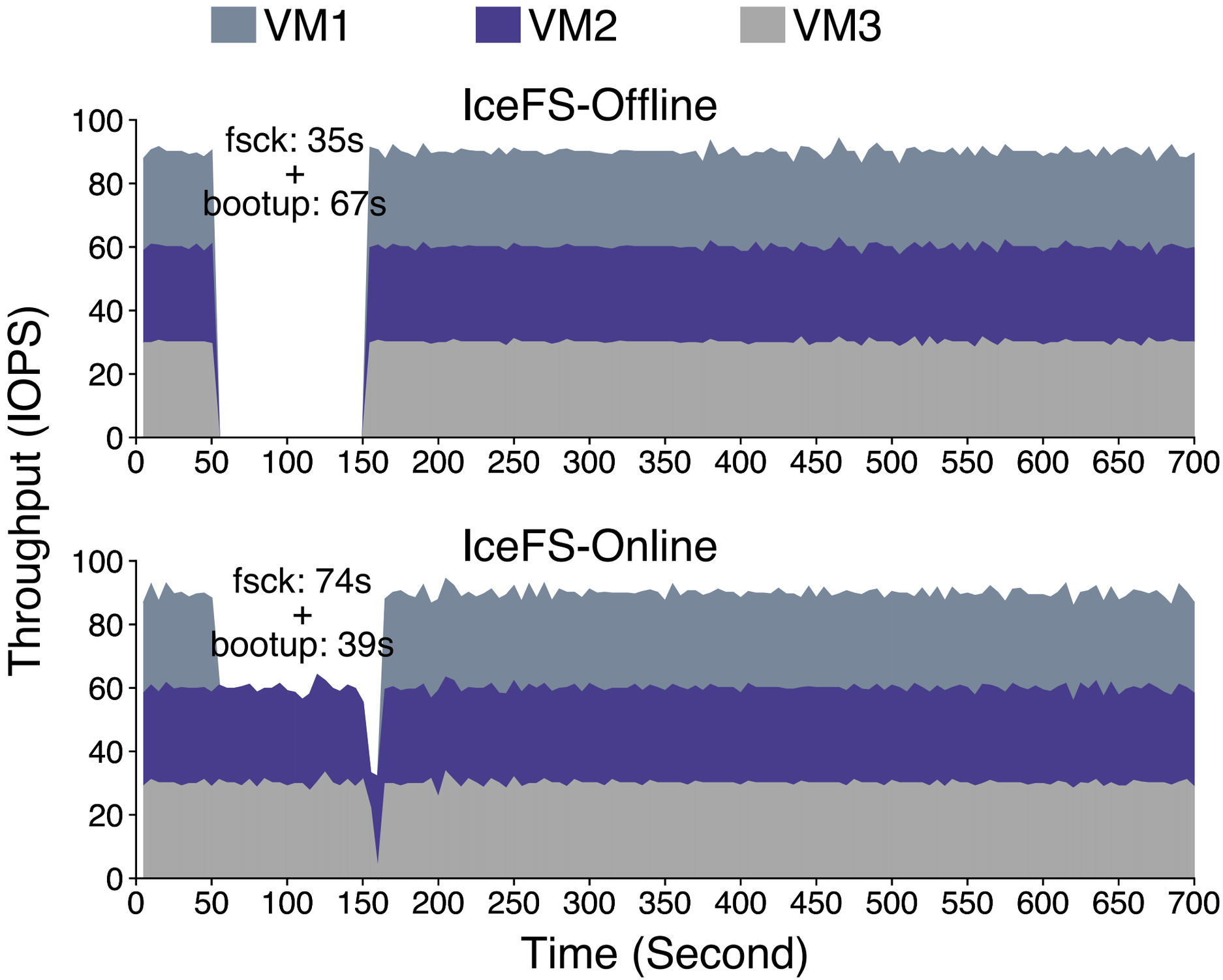
Server Virtualization with IceFS

recover
a cube
offline



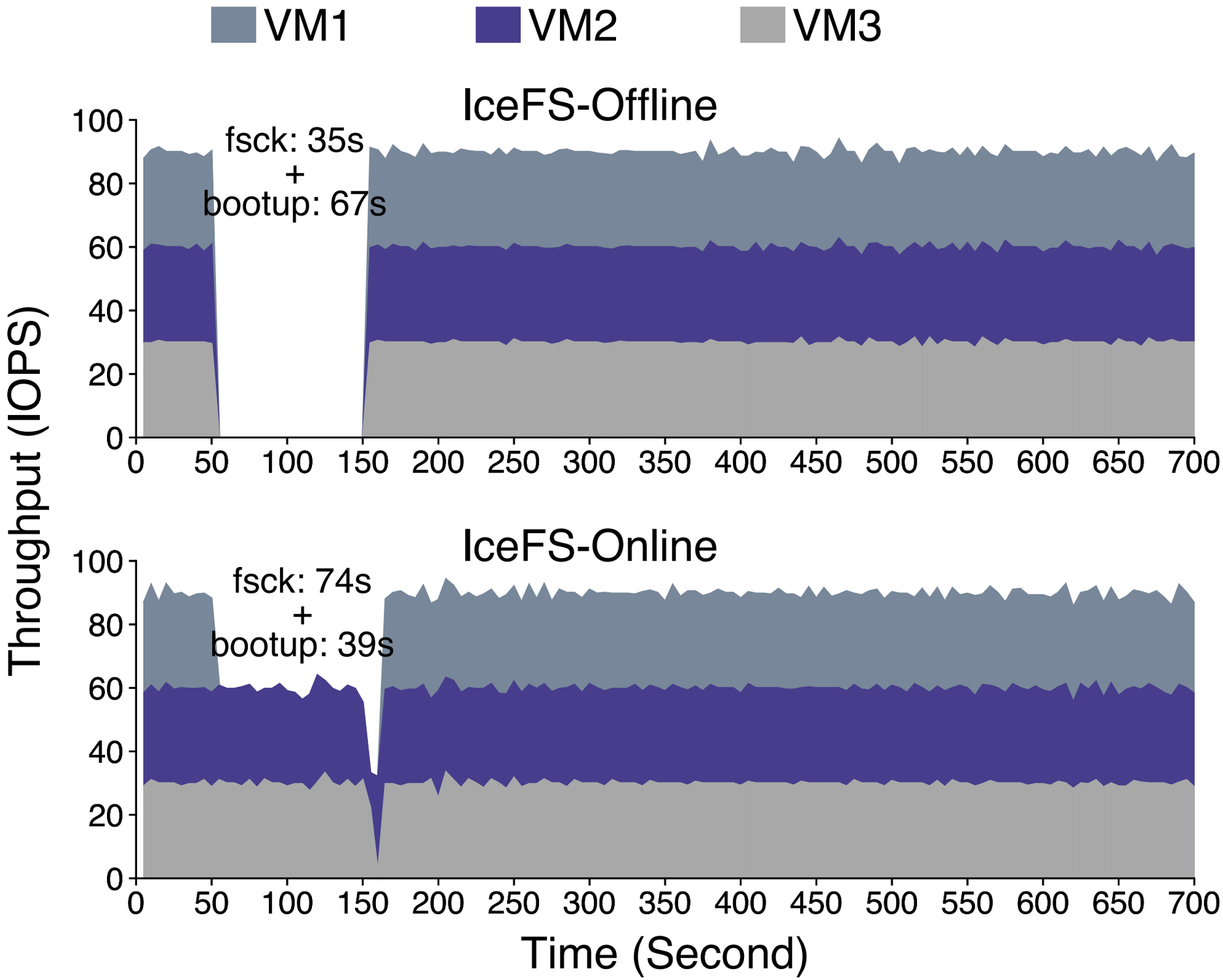
bootup three vms

Server Virtualization with IceFS



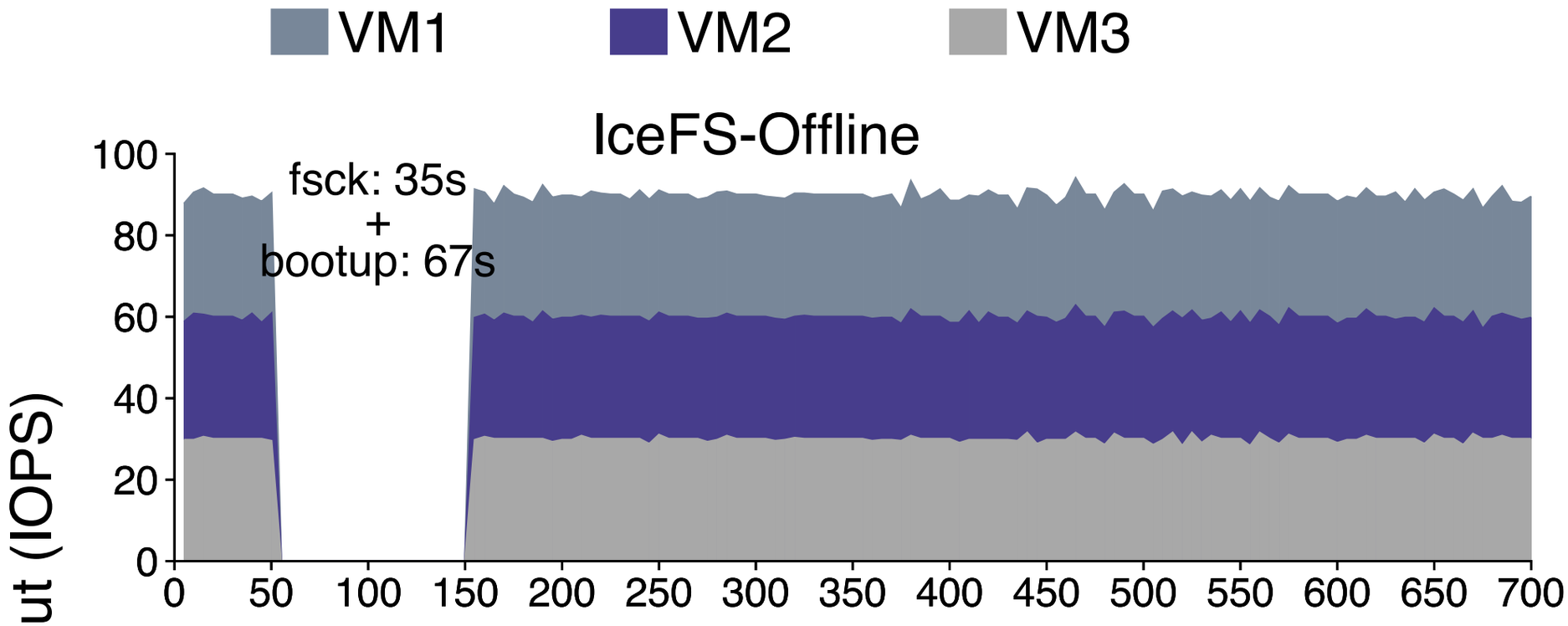
Server Virtualization with IceFS

recover
a cube
offline

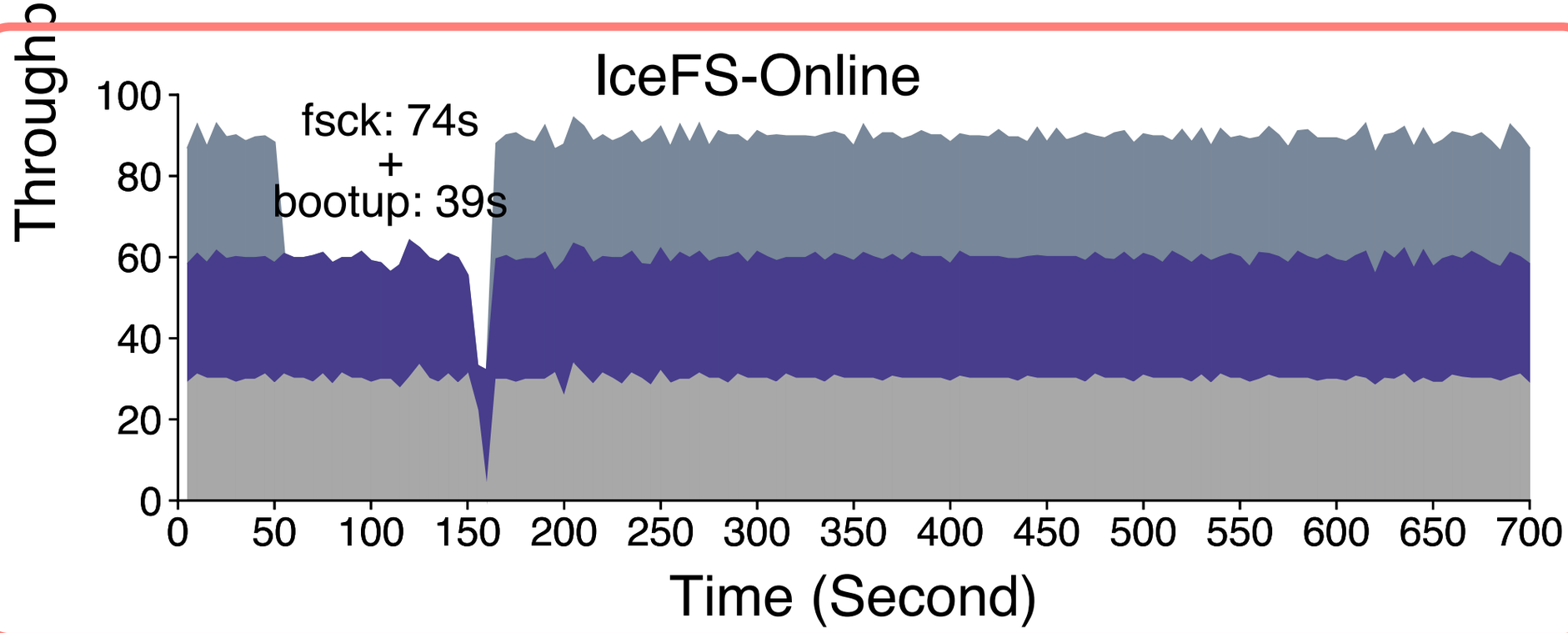


Server Virtualization with IceFS

recover a cube offline



recover a cube online



Evaluation

Evaluation

Isolate failures ?

- inject around 200 faults
- per-cube failure (read-only or crash) for IceFS

Faster recovery ?

- independent recovery for a cube

Better journaling performance ?

- isolated journaling performance for cubes
- flexibility between consistency and performance

Useful for applications ?

- significantly reduce system downtime

Conclusion

Conclusion

Local file systems lack physical isolation

- physical entanglement
- reliability and performance problems

Conclusion

Local file systems lack physical isolation

- physical entanglement
- reliability and performance problems

IceFS provides isolation with data containers

Conclusion

Local file systems lack physical isolation

- physical entanglement
- reliability and performance problems

IceFS provides isolation with data containers

Computing is becoming virtualized, shared, and multi-tenant

Conclusion

Local file systems lack physical isolation

- physical entanglement
- reliability and performance problems

IceFS provides isolation with data containers

Computing is becoming virtualized, shared, and multi-tenant

- isolation is the key

Conclusion

Local file systems lack physical isolation

- physical entanglement
- reliability and performance problems

IceFS provides isolation with data containers

Computing is becoming virtualized, shared, and multi-tenant

- isolation is the key

Systems need to rethink isolation

Conclusion

Local file systems lack physical isolation

- physical entanglement
- reliability and performance problems

IceFS provides isolation with data containers

Computing is becoming virtualized, shared, and multi-tenant

- isolation is the key

Systems need to rethink isolation

- avoid entanglement

Conclusion

Local file systems lack physical isolation

- physical entanglement
- reliability and performance problems

IceFS provides isolation with data containers

Computing is becoming virtualized, shared, and multi-tenant

- isolation is the key

Systems need to rethink isolation

- avoid entanglement
- provide useful abstractions for applications

Source code available soon at :

http://research.cs.wisc.edu/adsl/Software/icefs/

Source code available soon at :

http://research.cs.wisc.edu/adsl/Software/icefs/

Questions ?