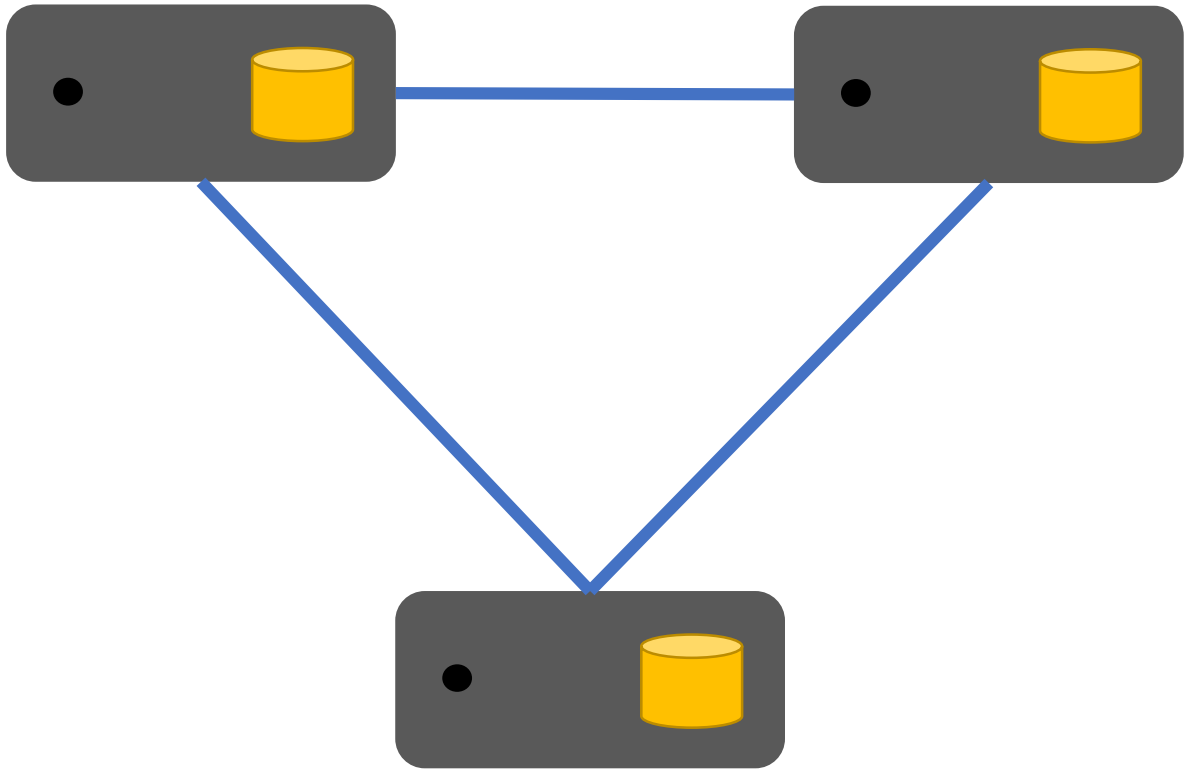
Redundancy Does Not Imply Fault Tolerance:

Analysis of Distributed Storage Reactions to Single Errors and Corruptions

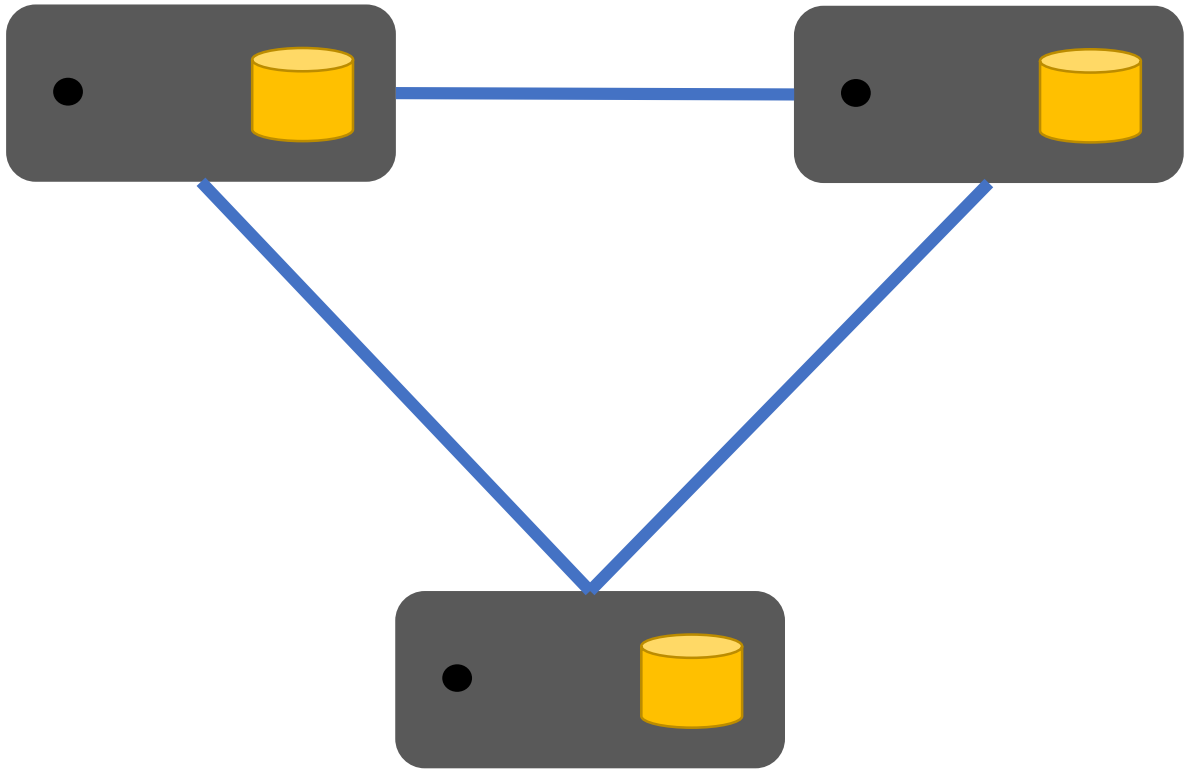
Aishwarya Ganesan, Ramnatthan Alagappan,

Andrea Arpaci-Dusseau, Remzi Arpaci-Dusseau

Redundancy for Fault Tolerance

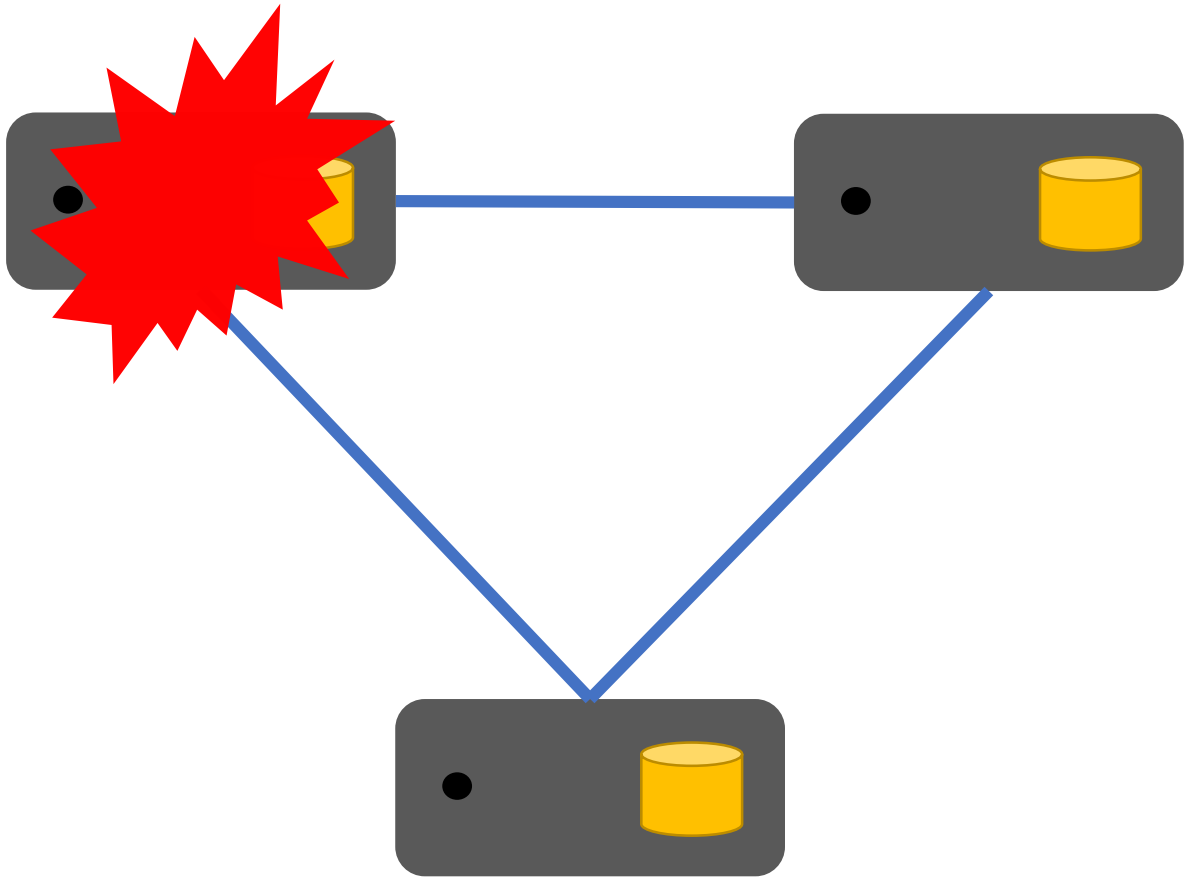


Redundancy for Fault Tolerance



Replication can mask failures

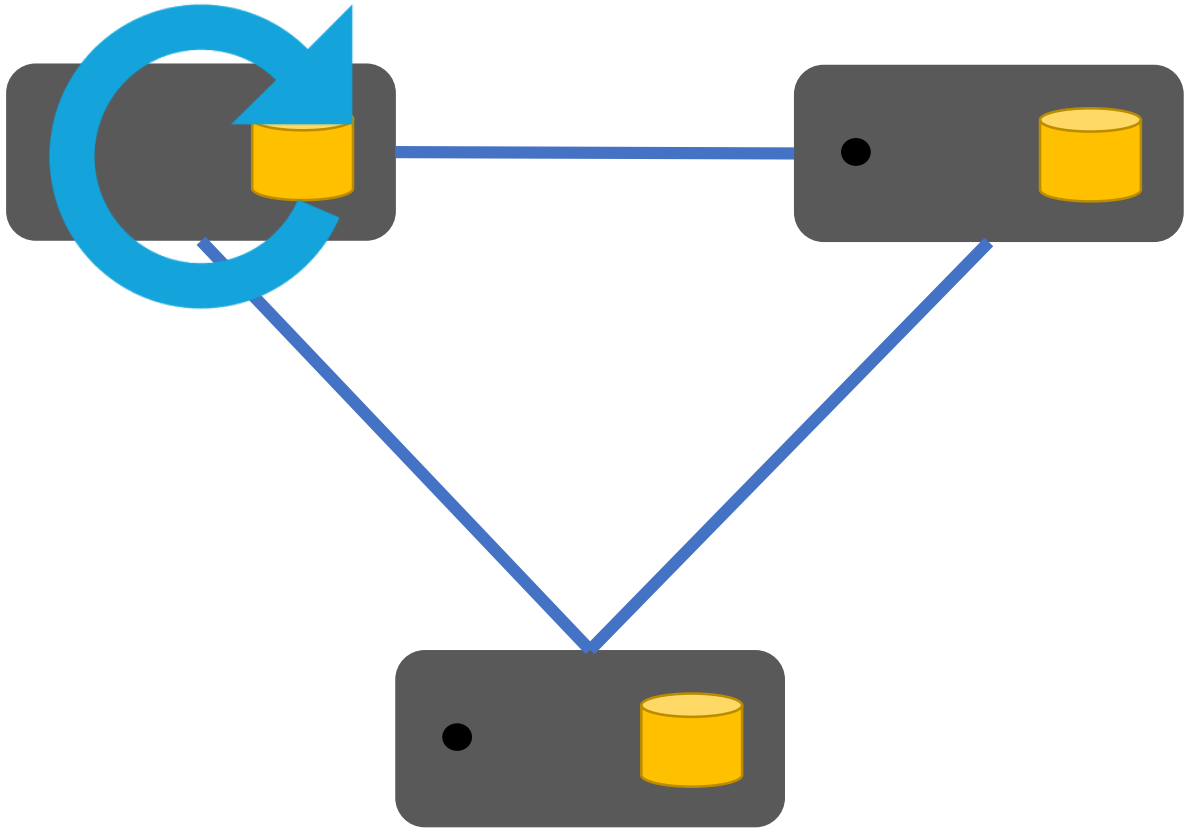
Redundancy for Fault Tolerance



Replication can mask failures

- System crashes

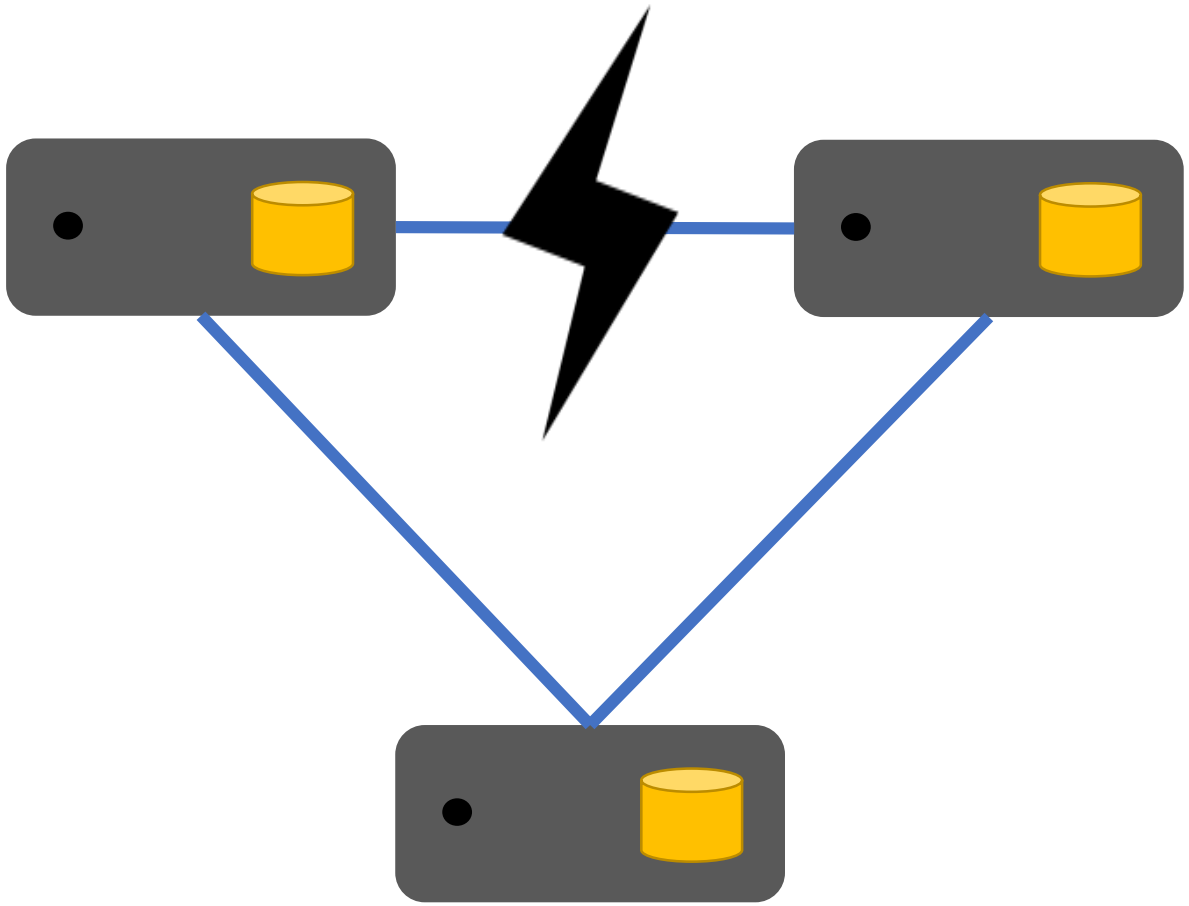
Redundancy for Fault Tolerance



Replication can mask failures

- System crashes
- Machine reboots

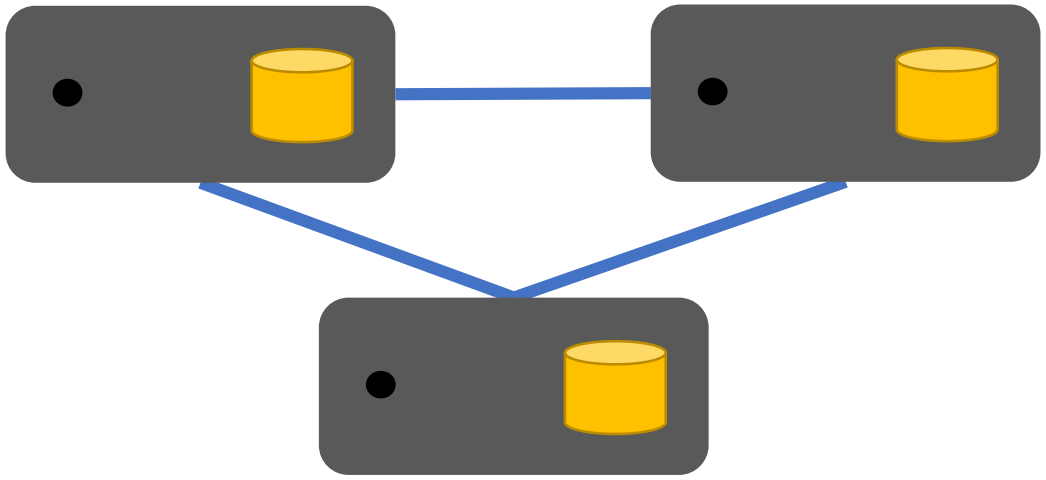
Redundancy for Fault Tolerance



Replication can mask failures

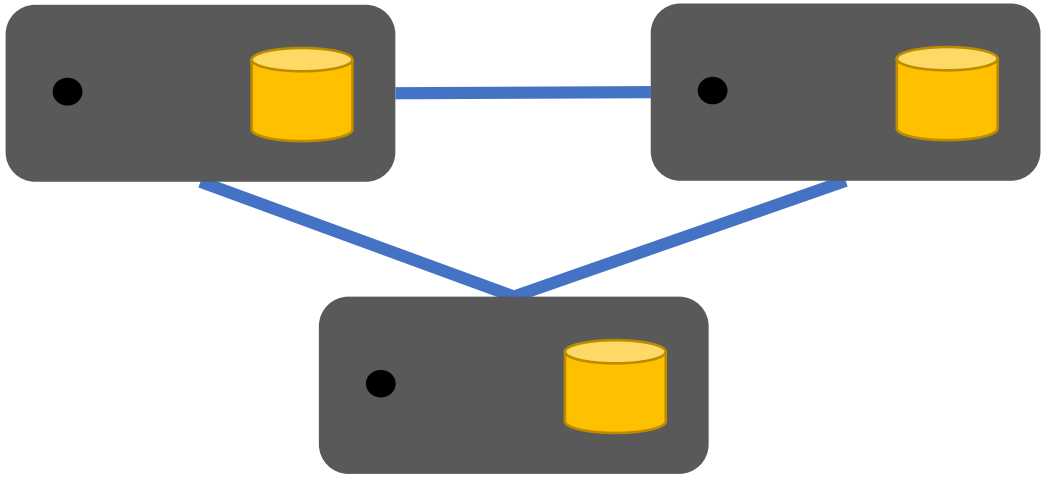
- System crashes
- Machine reboots
- Network failures

Redundancy in Distributed Storage



Redundancy in Distributed Storage

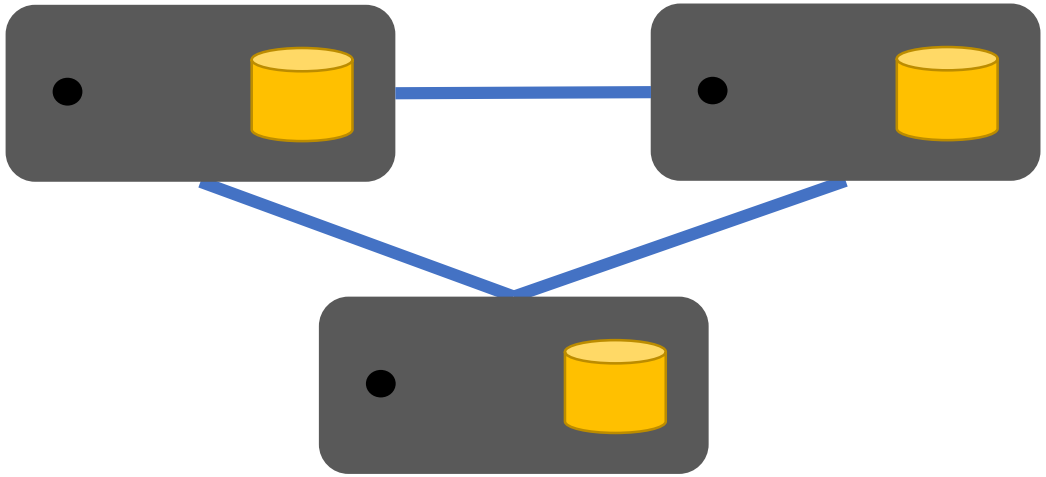
Depend on local file systems to store data



Redundancy in Distributed Storage

Depend on local file systems to store data

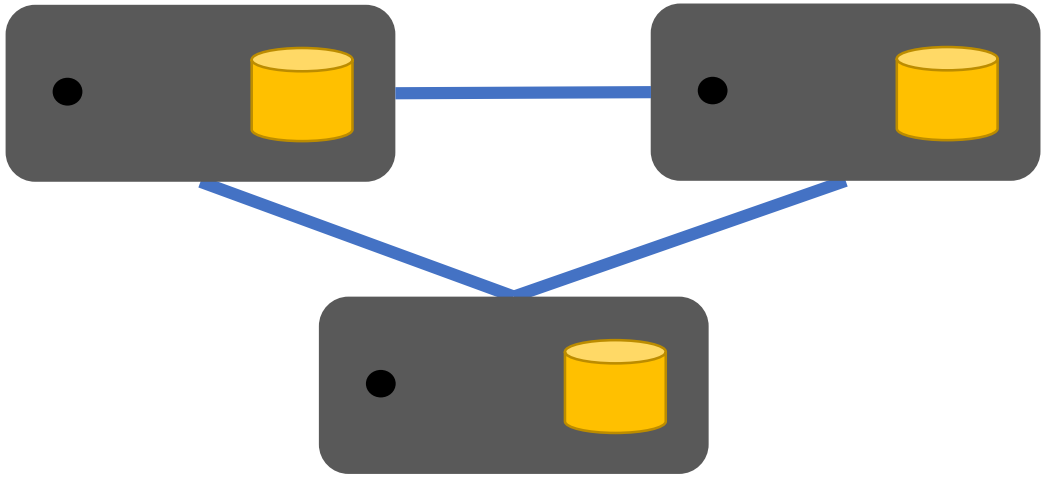
How about partial storage faults?



Redundancy in Distributed Storage

Depend on local file systems to store data

How about partial storage faults?

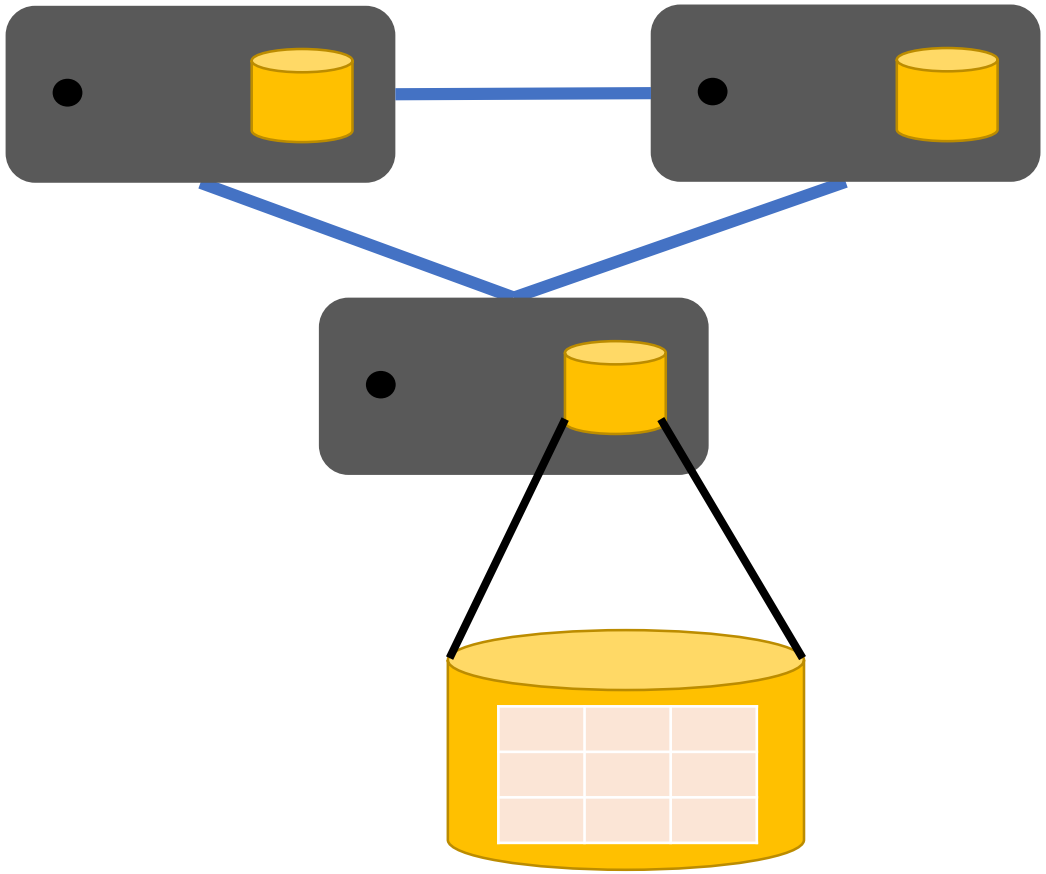


File system may return corrupted data on reads

Redundancy in Distributed Storage

Depend on local file systems to store data

How about partial storage faults?

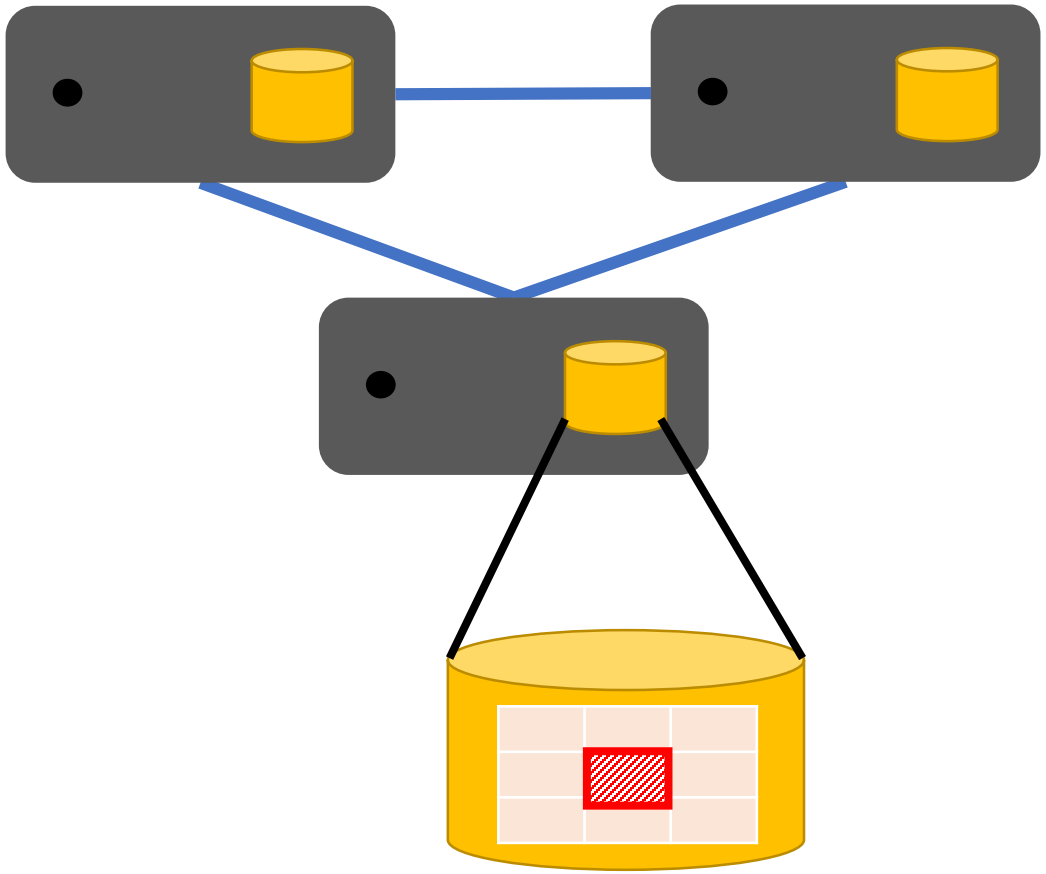


File system may return corrupted data on reads

Redundancy in Distributed Storage

Depend on local file systems to store data

How about partial storage faults?



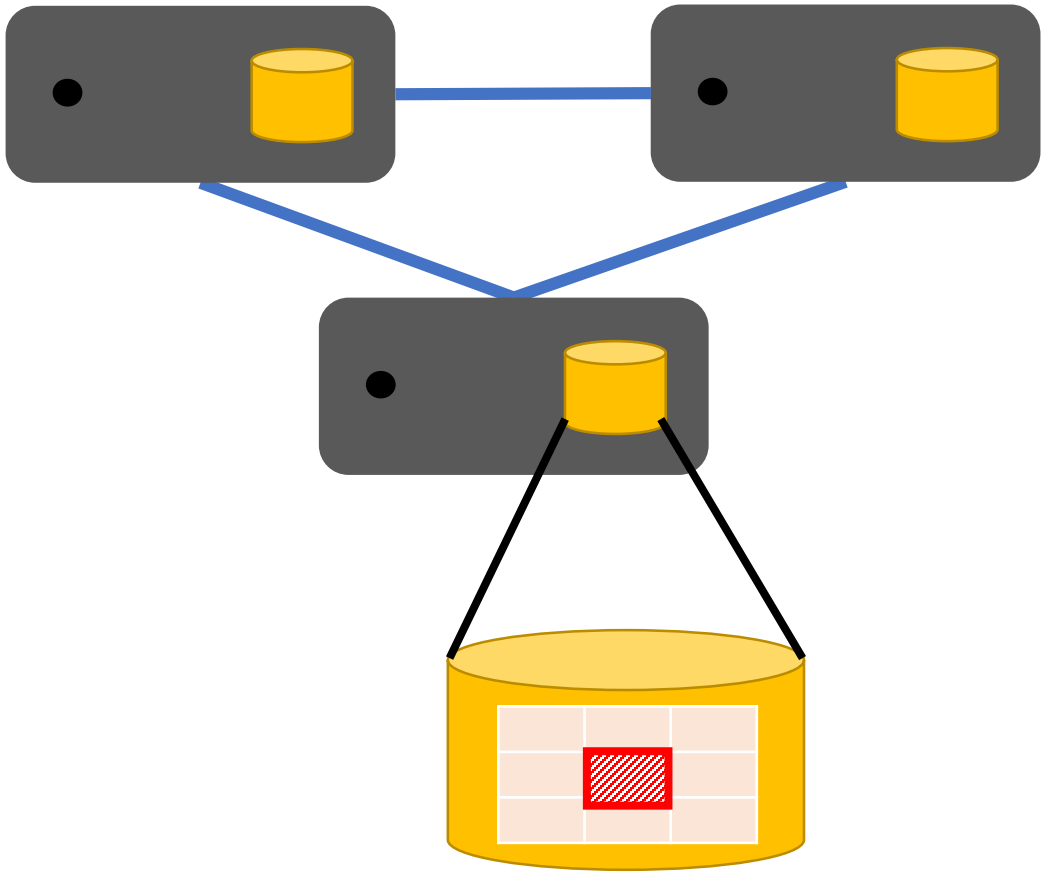
File system may return corrupted data on reads

- disk block corruption

Redundancy in Distributed Storage

Depend on local file systems to store data

How about partial storage faults?



File system may return corrupted data on reads

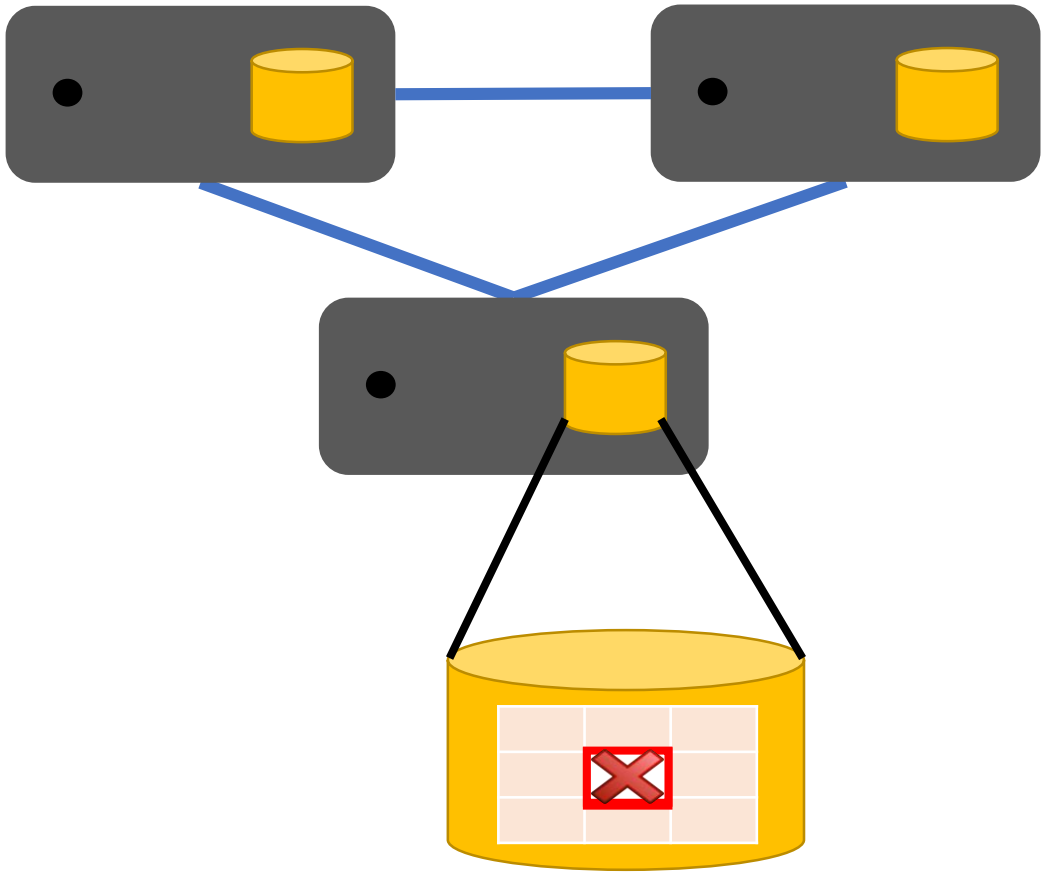
- disk block corruption

File system may return I/O error on a read

Redundancy in Distributed Storage

Depend on local file systems to store data

How about partial storage faults?



File system may return corrupted data on reads

- disk block corruption

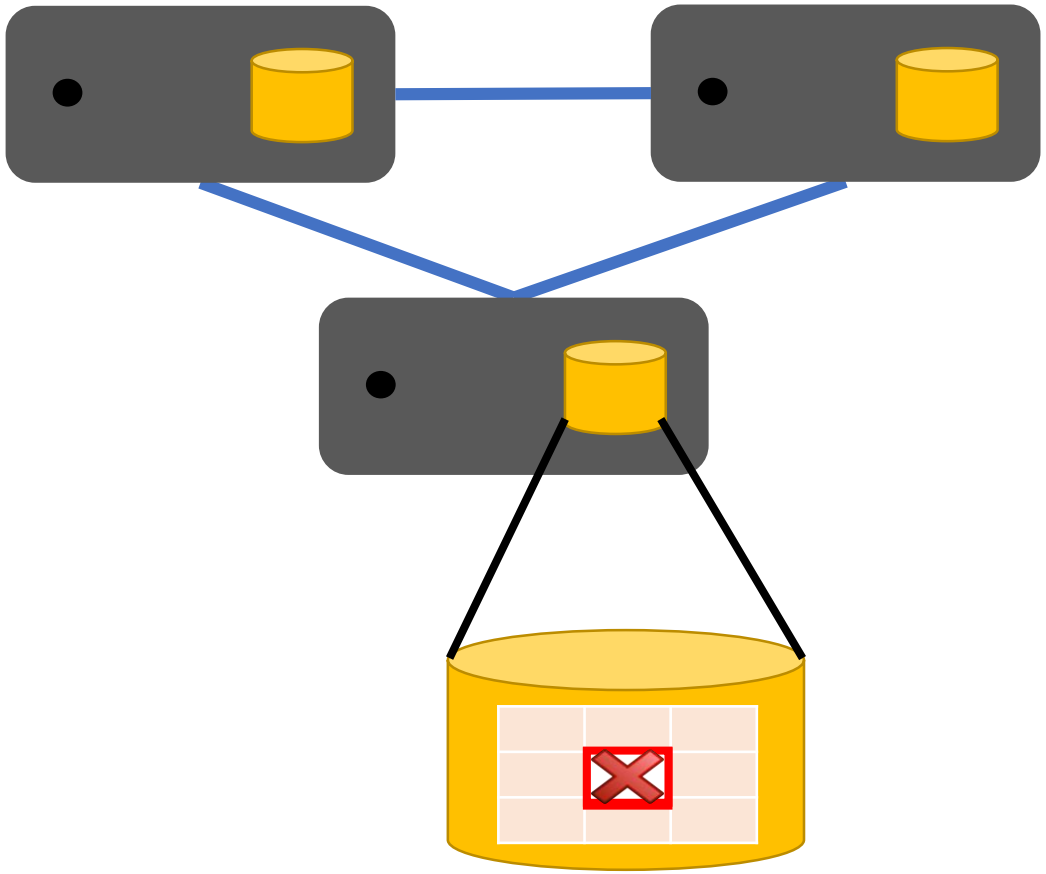
File system may return I/O error on a read

- latent sector error

Redundancy in Distributed Storage

Depend on local file systems to store data

How about partial storage faults?



File system may return corrupted data on reads

- disk block corruption

File system may return I/O error on a read

- latent sector error

We call these file-system faults

Redundancy in Distributed Storage

Depend on local file systems to store data

How about partial storage faults?

File system may return corrupted data

Do distributed storage systems use *redundancy* to *recover* from local file-system faults?

read

- latent sector error

We call these file-system faults

Our Study

Our Study

Behavior of eight distributed systems in response to file-system faults

Our Study

Behavior of eight distributed systems in response to file-system faults

Broad spectrum of replication and consensus protocols

Our Study

Behavior of eight distributed systems in response to file-system faults

Broad spectrum of replication and consensus protocols

Replicated state machines

- ZooKeeper (uses ZAB for consensus)
- LogCabin, CockroachDB, and RethinkDB (uses RAFT for consensus)

Our Study

Behavior of eight distributed systems in response to file-system faults

Broad spectrum of replication and consensus protocols

Replicated state machines

- ZooKeeper (uses ZAB for consensus)
- LogCabin, CockroachDB, and RethinkDB (uses RAFT for consensus)

Primary backup replication

- MongoDB
- Redis
- Kafka (in-sync replicas for leader election)

Our Study

Behavior of eight distributed systems in response to file-system faults

Broad spectrum of replication and consensus protocols

Replicated state machines

- ZooKeeper (uses ZAB for consensus)
- LogCabin, CockroachDB, and RethinkDB (uses RAFT for consensus)

Primary backup replication

- MongoDB
- Redis
- Kafka (in-sync replicas for leader election)

Dynamo-style quorum

- Cassandra (decentralized, no leader/follower)

Fault model

Fault model

A single fault to a single file-system block in a single node

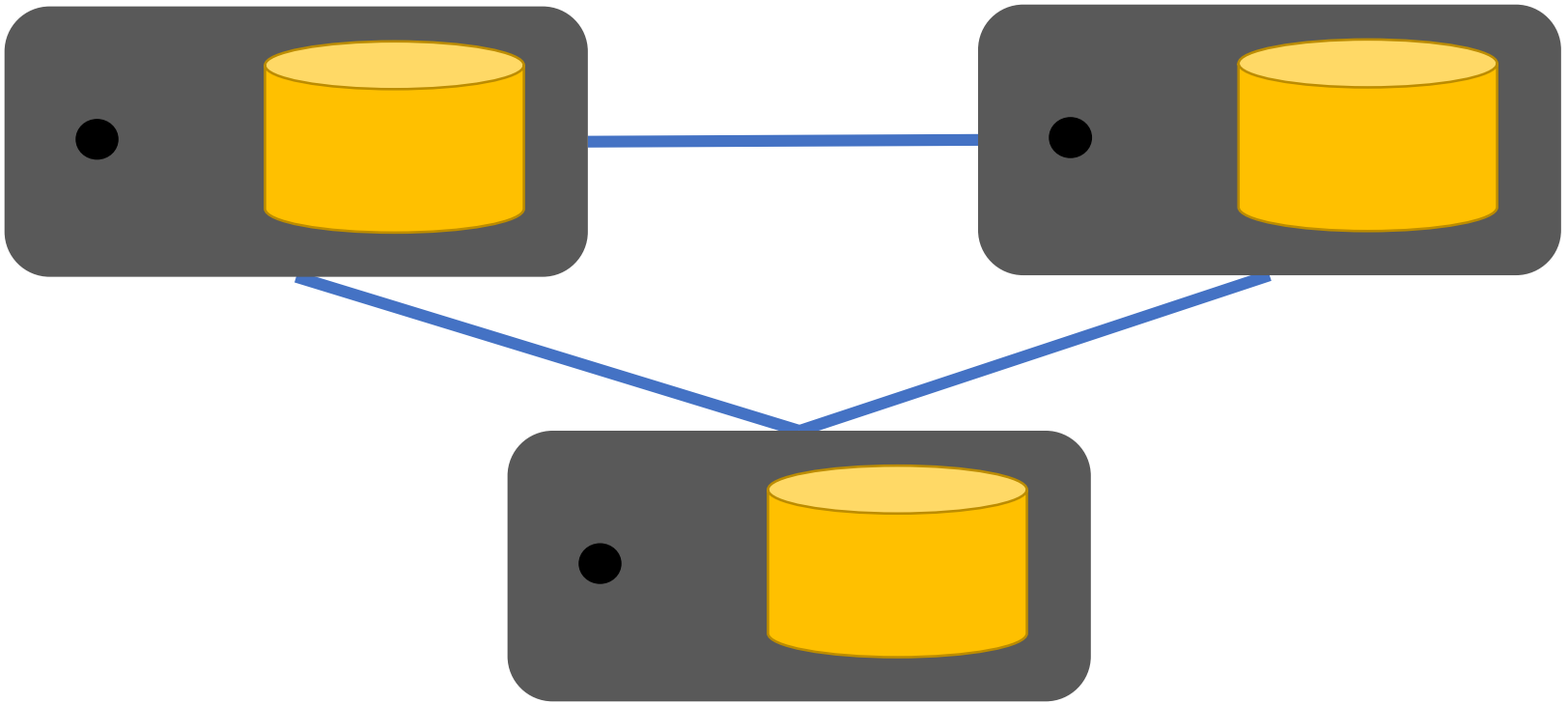
Fault model

A single fault to a single file-system block in a single node

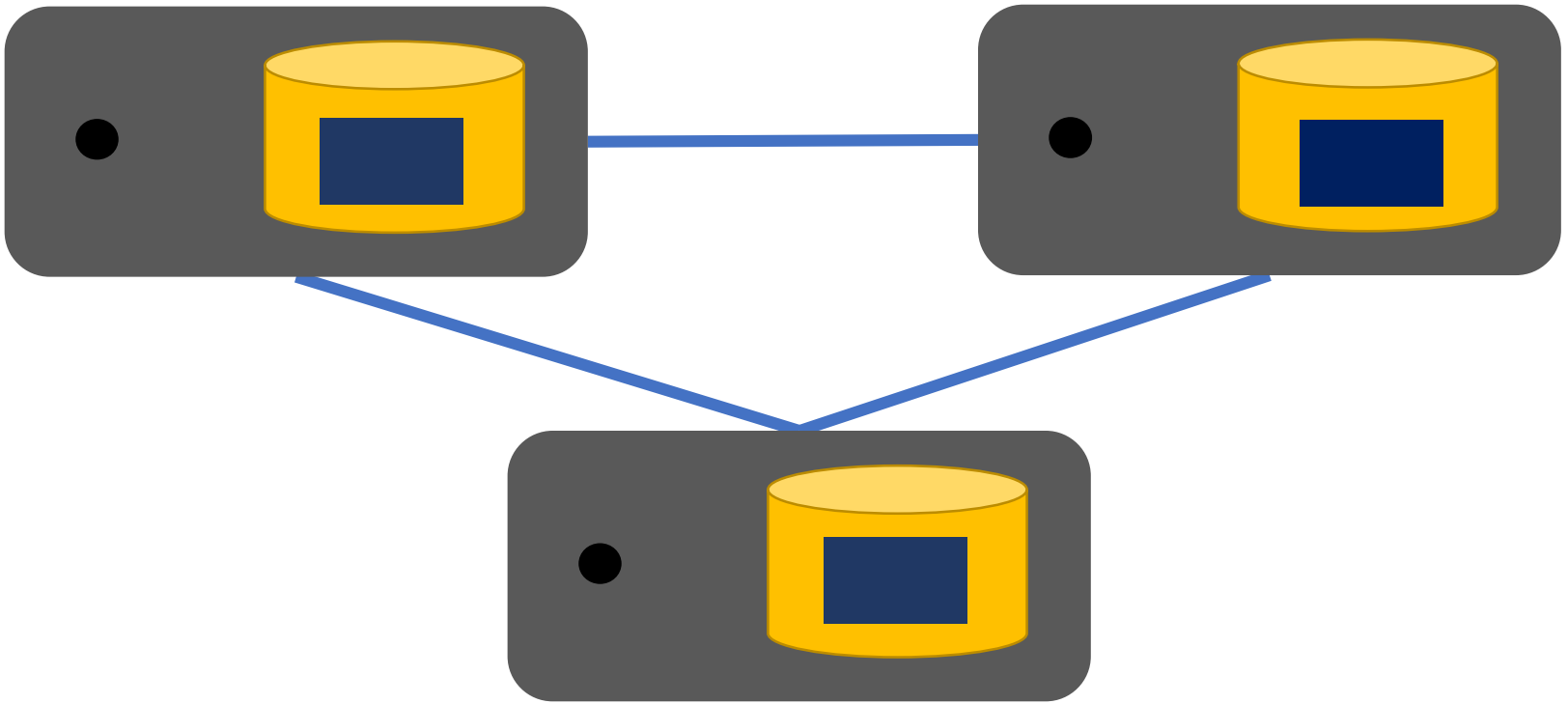
Type of faults:

- corruptions
- read errors
- write errors

Common Expectation

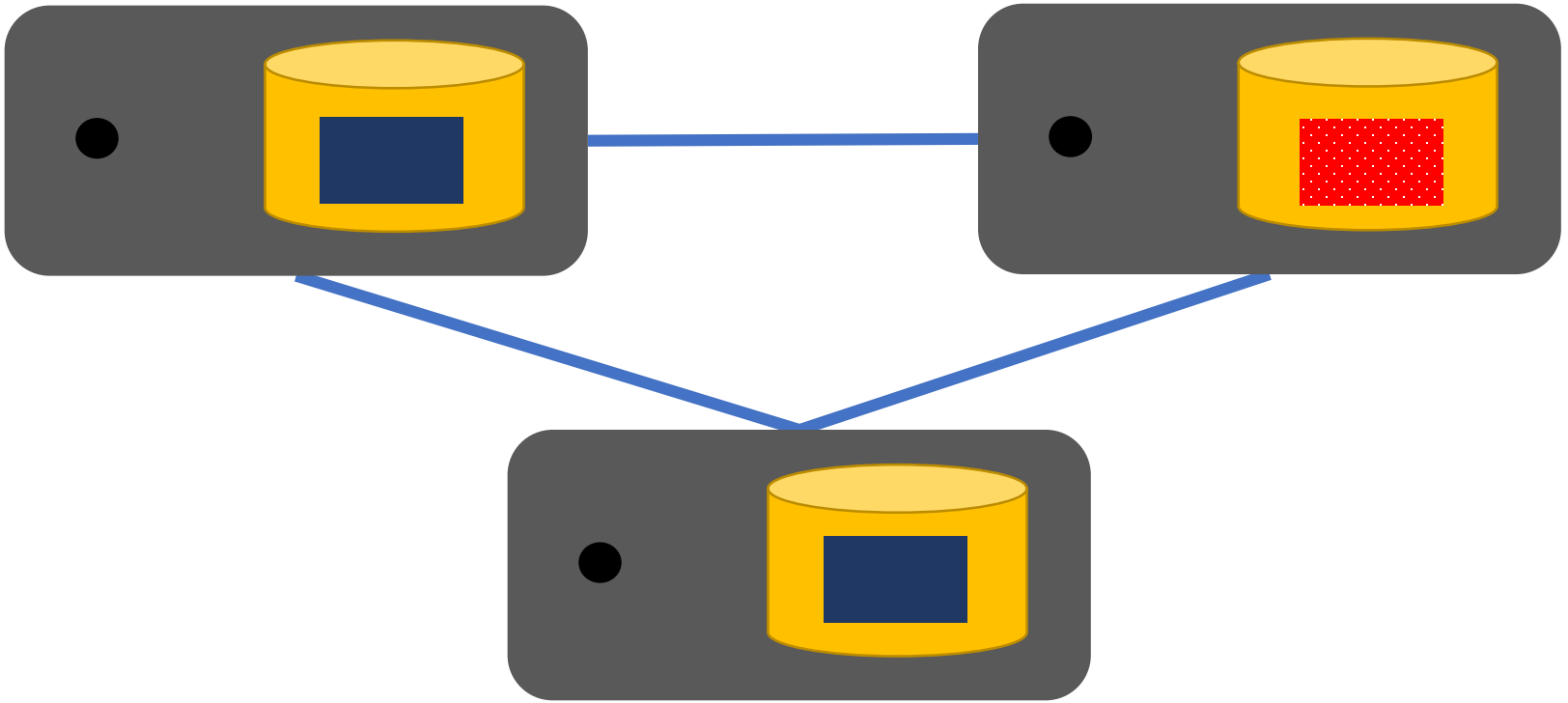


Common Expectation



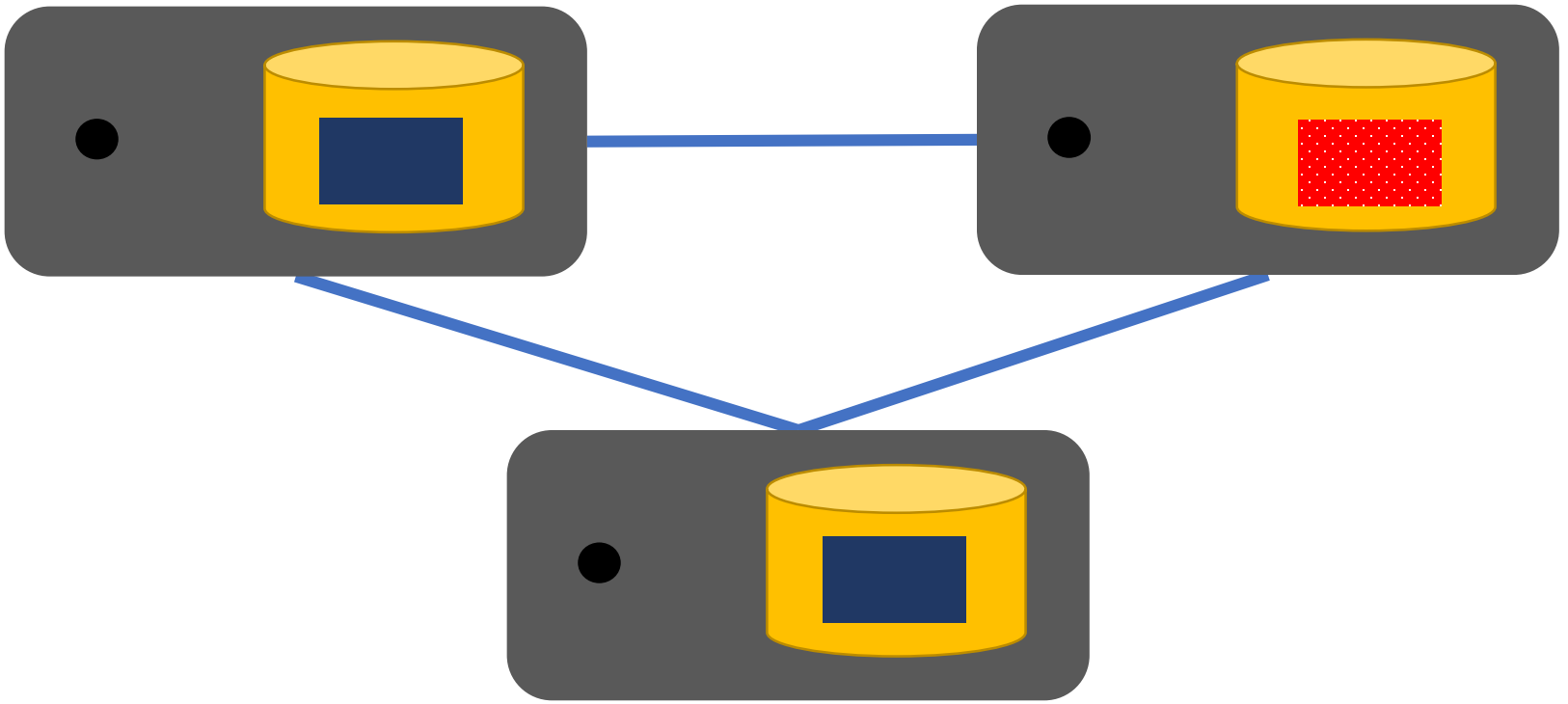
Common Expectation

Fault Model: A single fault to one block in only one replica



Common Expectation

Fault Model: A single fault to one block in only one replica



Redundancy would enable recovery from local file-system faults

Redundancy Does Not Imply Fault Tolerance

Redundancy Does Not Imply Fault Tolerance

A single fault in one node can cause catastrophic outcomes

Redundancy Does Not Imply Fault Tolerance

A single fault in one node can cause catastrophic outcomes

- data loss, corruption, unavailability, and spread of corruption to other intact replicas

Redundancy Does Not Imply Fault Tolerance

A single fault in one node can cause catastrophic outcomes

- data loss, corruption, unavailability, and spread of corruption to other intact replicas

	Silent corruption	Unavailability	Data loss	Reduced redundancy	Query failures
Redis					
ZooKeeper					
Cassandra					
Kafka					
RethinkDB					
MongoDB					
LogCabin					
CockroachDB					

Redundancy Does Not Imply Fault Tolerance

A single fault in one node can cause catastrophic outcomes

- data loss, corruption, unavailability, and spread of corruption to other intact replicas

	Silent corruption	Unavailability	Data loss	Reduced redundancy	Query failures
Redis	■	■	■	■	■
ZooKeeper		■	■	■	
Cassandra	■	■	■	■	
Kafka		■		■	■
RethinkDB	■			■	
MongoDB				■	
LogCabin				■	
CockroachDB		■	■	■	■

Why does Redundancy Not Imply Fault Tolerance?

Why does Redundancy Not Imply Fault Tolerance?

Some fundamental problems across multiple systems – not just implementation bugs!

Why does Redundancy Not Imply Fault Tolerance?

Some fundamental problems across multiple systems – not just implementation bugs!

Faults are often undetected locally – leads to harmful global effects

Why does Redundancy Not Imply Fault Tolerance?

Some fundamental problems across multiple systems – not just implementation bugs!

Faults are often undetected locally – leads to harmful global effects

On detection, crashing is the common action – redundancy underutilized

Why does Redundancy Not Imply Fault Tolerance?

Some fundamental problems across multiple systems – not just implementation bugs!

Faults are often undetected locally – leads to harmful global effects

On detection, crashing is the common action – redundancy underutilized

Crash and corruption handling are entangled – loss of committed data

Why does Redundancy Not Imply Fault Tolerance?

Some fundamental problems across multiple systems – not just implementation bugs!

Faults are often undetected locally – leads to harmful global effects

On detection, crashing is the common action – redundancy underutilized

Crash and corruption handling are entangled – loss of committed data

Unsafe interaction between local behavior and global distributed protocols can spread corruption or data loss

Fundamental Problems: Summary

Fundamental Problems: Summary

	Locally undetected faults?	Crashing - common local action?	Crash & corruption handling entangled?	Unsafe interaction of local & global protocols?	Redundancy underutilized for recovery?
Redis					
ZooKeeper					
Cassandra					
Kafka					
RethinkDB					
MongoDB					
LogCabin					
CockroachDB					

Fundamental Problems: Summary

	Locally undetected faults?	Crashing - common local action?	Crash & corruption handling entangled?	Unsafe interaction of local & global protocols?	Redundancy underutilized for recovery?
Redis	■				
ZooKeeper	■				
Cassandra	■				
Kafka					
RethinkDB	■				
MongoDB					
LogCabin					
CockroachDB	■				

Fundamental Problems: Summary

	Locally undetected faults?	Crashing - common local action?	Crash & corruption handling entangled?	Unsafe interaction of local & global protocols?	Redundancy underutilized for recovery?
Redis	■	■			
ZooKeeper	■	■			
Cassandra	■	■			
Kafka		■			
RethinkDB	■	■			
MongoDB		■			
LogCabin		■			
CockroachDB	■	■			

Fundamental Problems: Summary

	Locally undetected faults?	Crashing - common local action?	Crash & corruption handling entangled?	Unsafe interaction of local & global protocols?	Redundancy underutilized for recovery?
Redis	■	■			
ZooKeeper	■	■	■		
Cassandra	■	■			
Kafka		■	■		
RethinkDB	■	■	■		
MongoDB		■	■		
LogCabin		■	■		
CockroachDB	■	■			

Fundamental Problems: Summary

	Locally undetected faults?	Crashing - common local action?	Crash & corruption handling entangled?	Unsafe interaction of local & global protocols?	Redundancy underutilized for recovery?
Redis	■	■		■	
ZooKeeper	■	■	■		
Cassandra	■	■		■	
Kafka		■	■	■	
RethinkDB	■	■	■		
MongoDB		■	■		
LogCabin		■	■		
CockroachDB	■	■			

Fundamental Problems: Summary

	Locally undetected faults?	Crashing - common local action?	Crash & corruption handling entangled?	Unsafe interaction of local & global protocols?	Redundancy underutilized for recovery?
Redis	■	■		■	■
ZooKeeper	■	■	■		■
Cassandra	■	■		■	■
Kafka		■	■	■	■
RethinkDB	■	■	■		■
MongoDB		■	■		■
LogCabin		■	■		■
CockroachDB	■	■			■

Outline

Introduction

Fault Injection

System Behavior Analysis

Major Results

Observations Across Systems

Conclusion

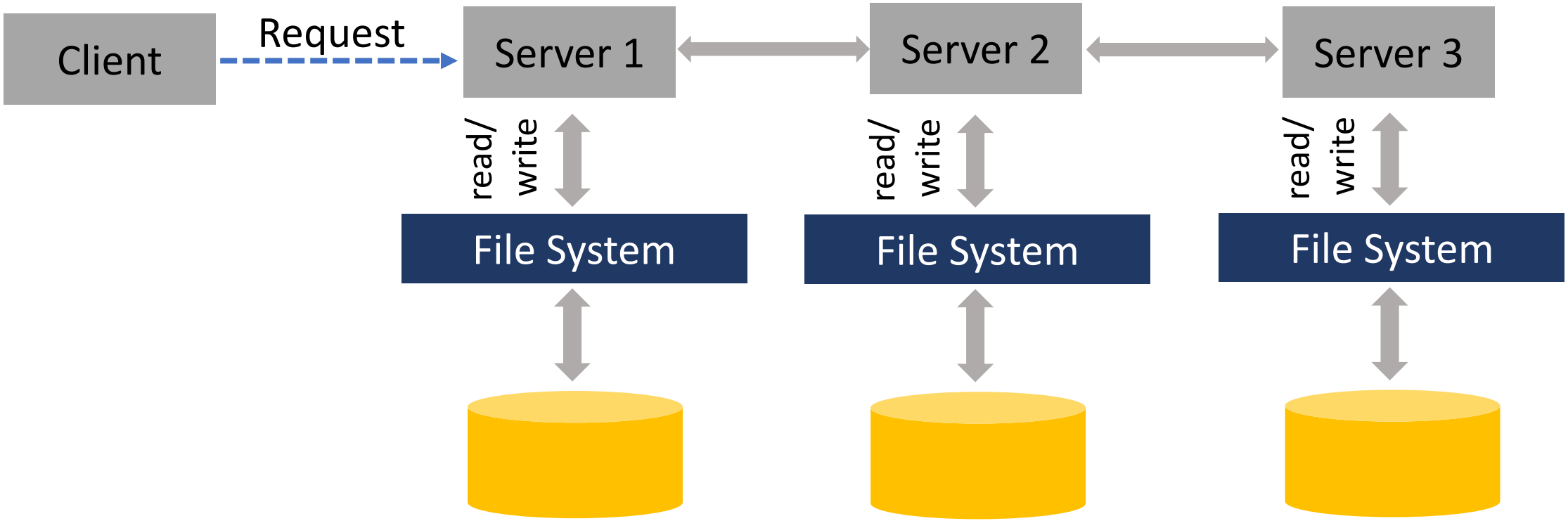
Fault Model

Fault Model

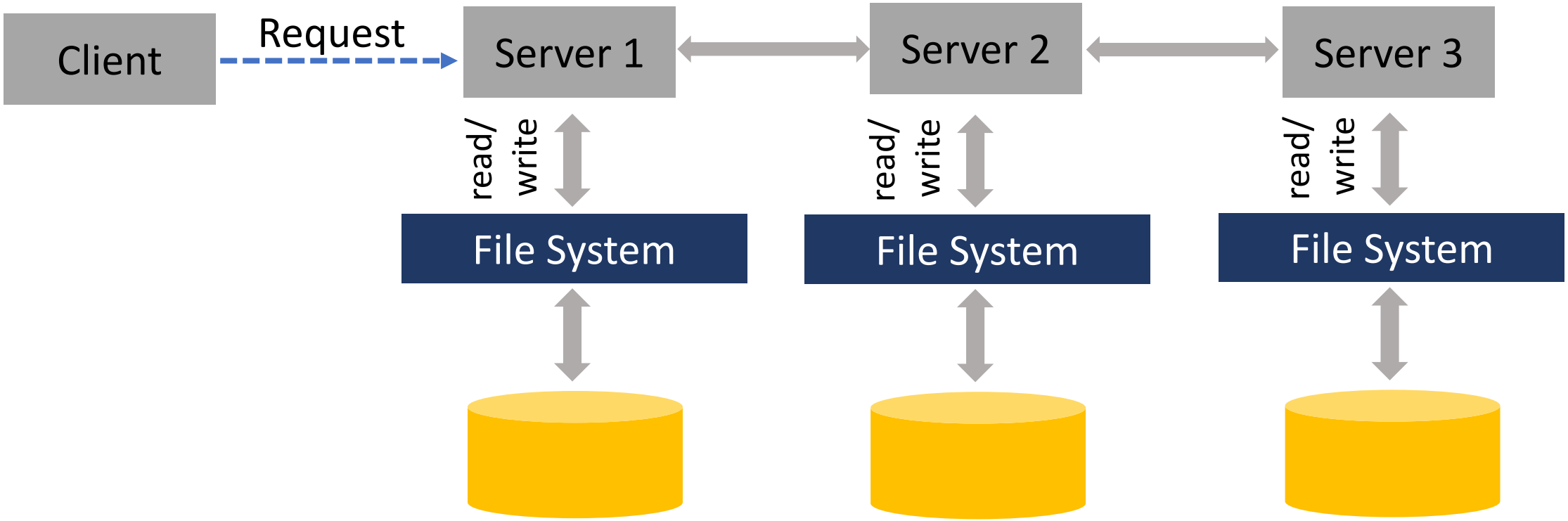
Fault Model

Fault Model

Fault Model

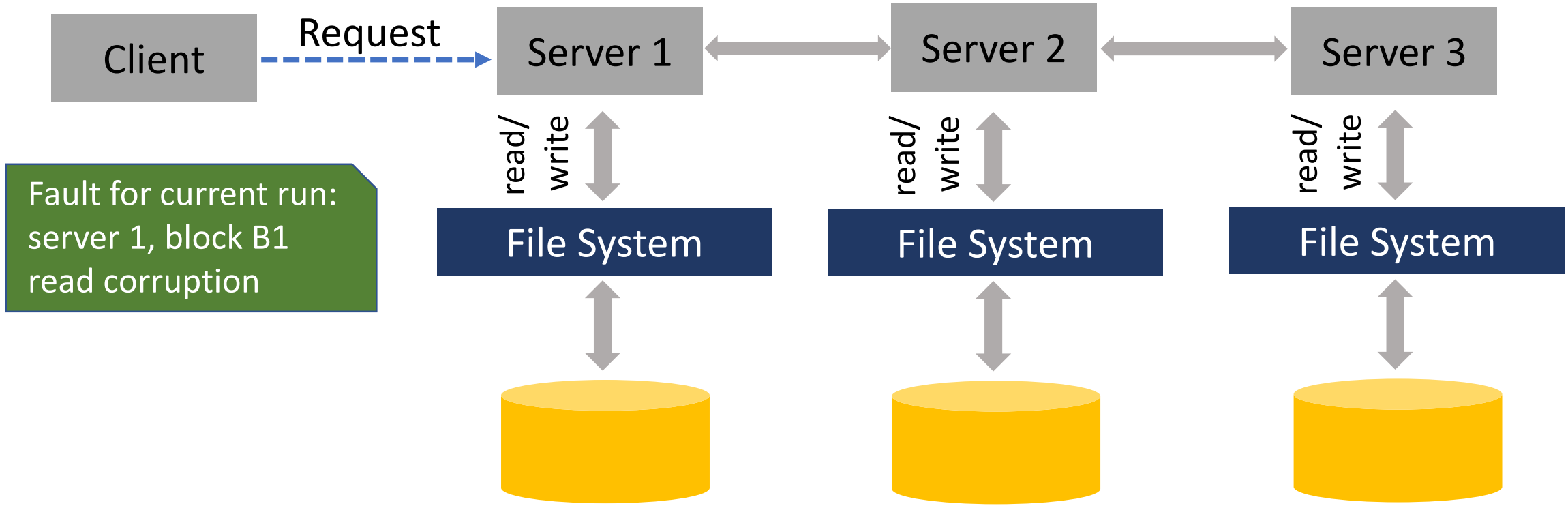

Client
Request
Server 1
Server 2
Server 3
read/
write
read/
write
read/
write
File System
File System
File System

Fault Model



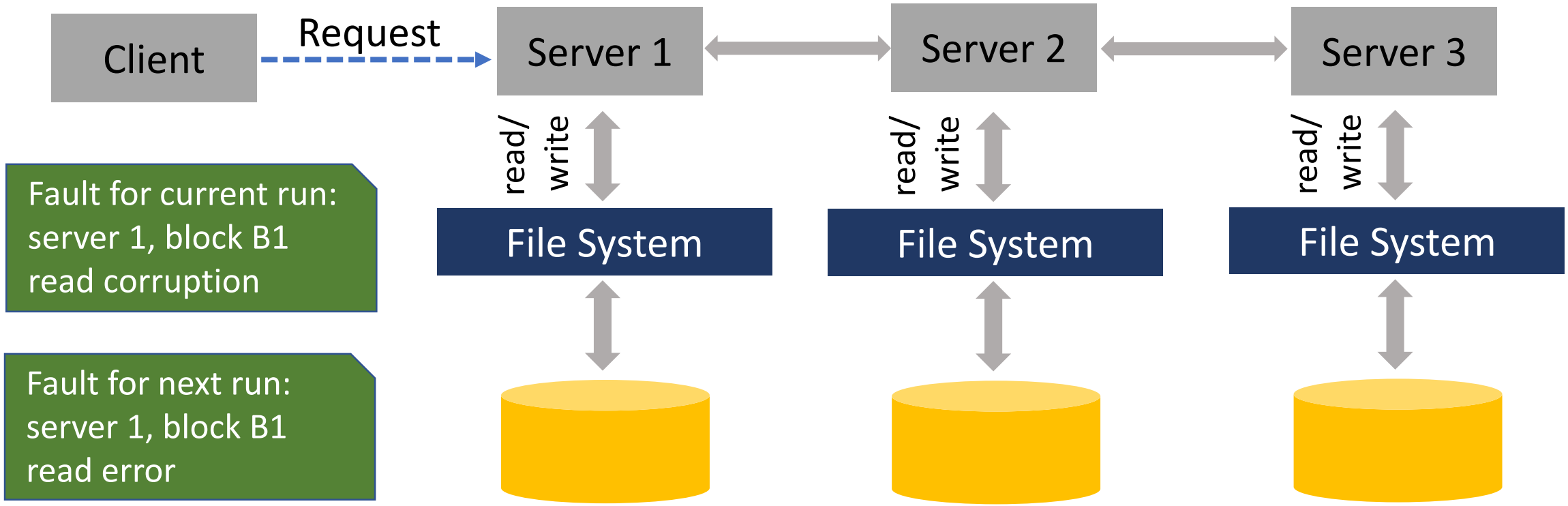
A single fault to a single file-system block in a single node

Fault Model



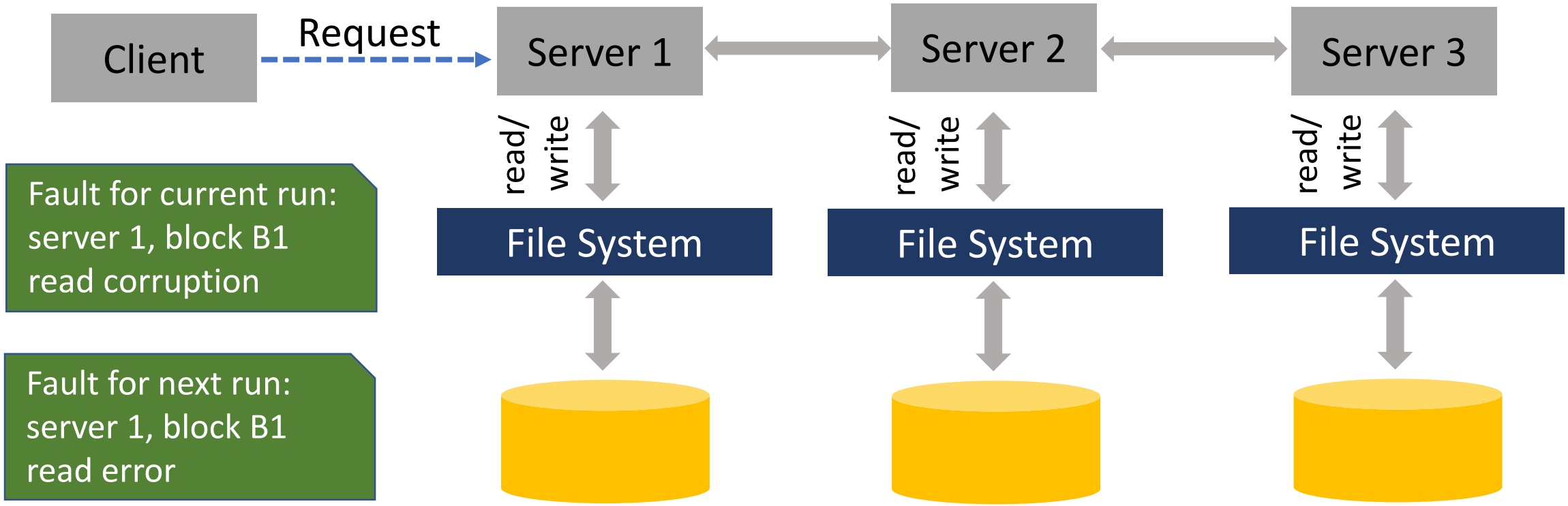
A single fault to a single file-system block in a single node

Fault Model



A single fault to a single file-system block in a single node

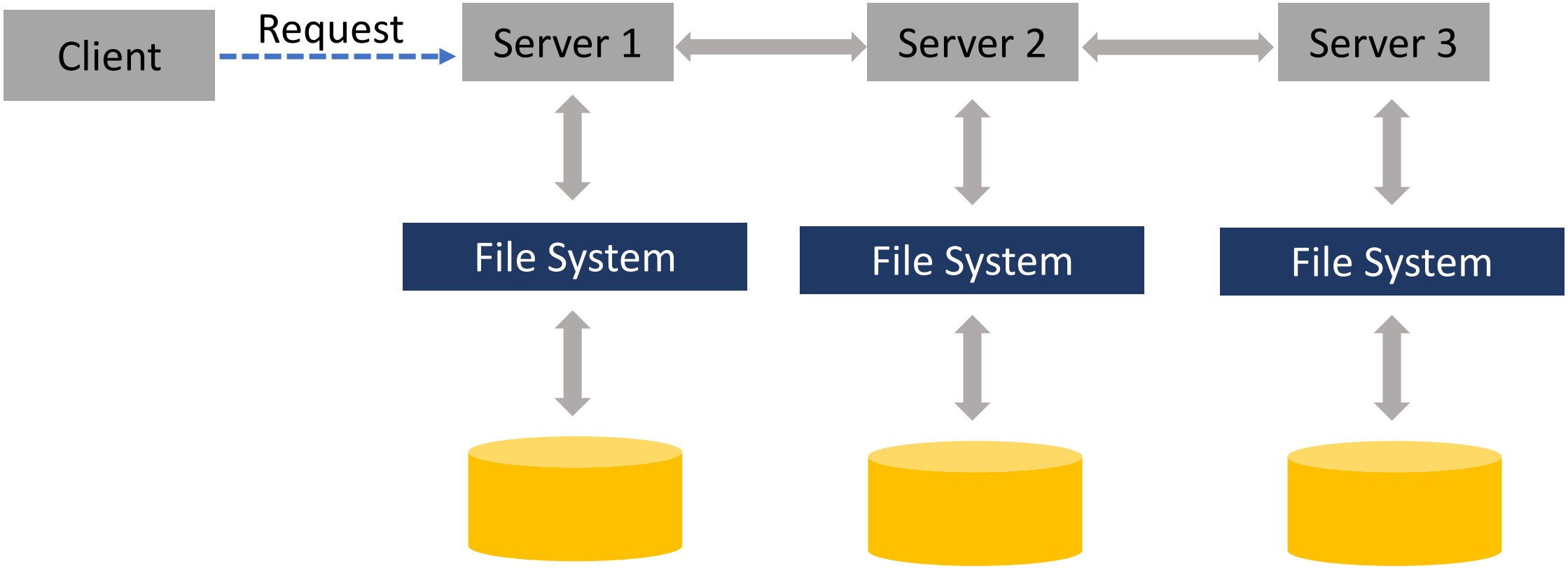
Fault Model



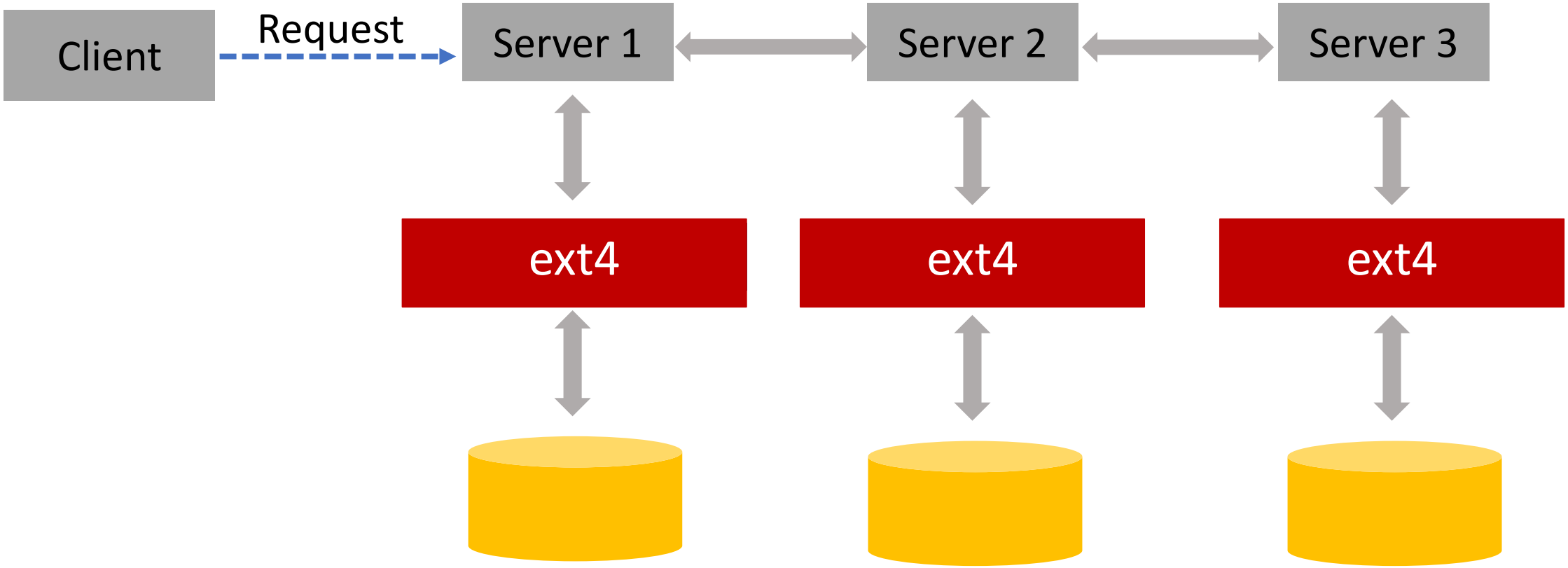
A single fault to a single file-system block in a single node

Faults injected only to user data not filesystem metadata

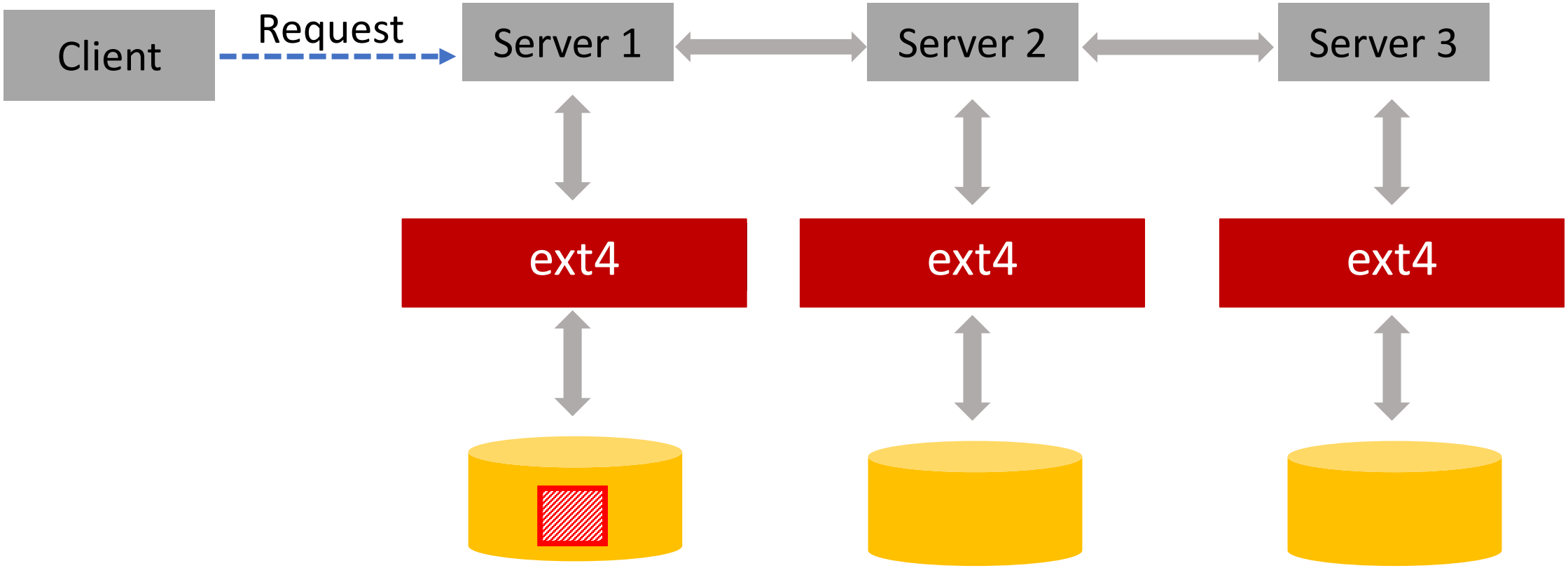
Fault Model: ext4 and btrfs



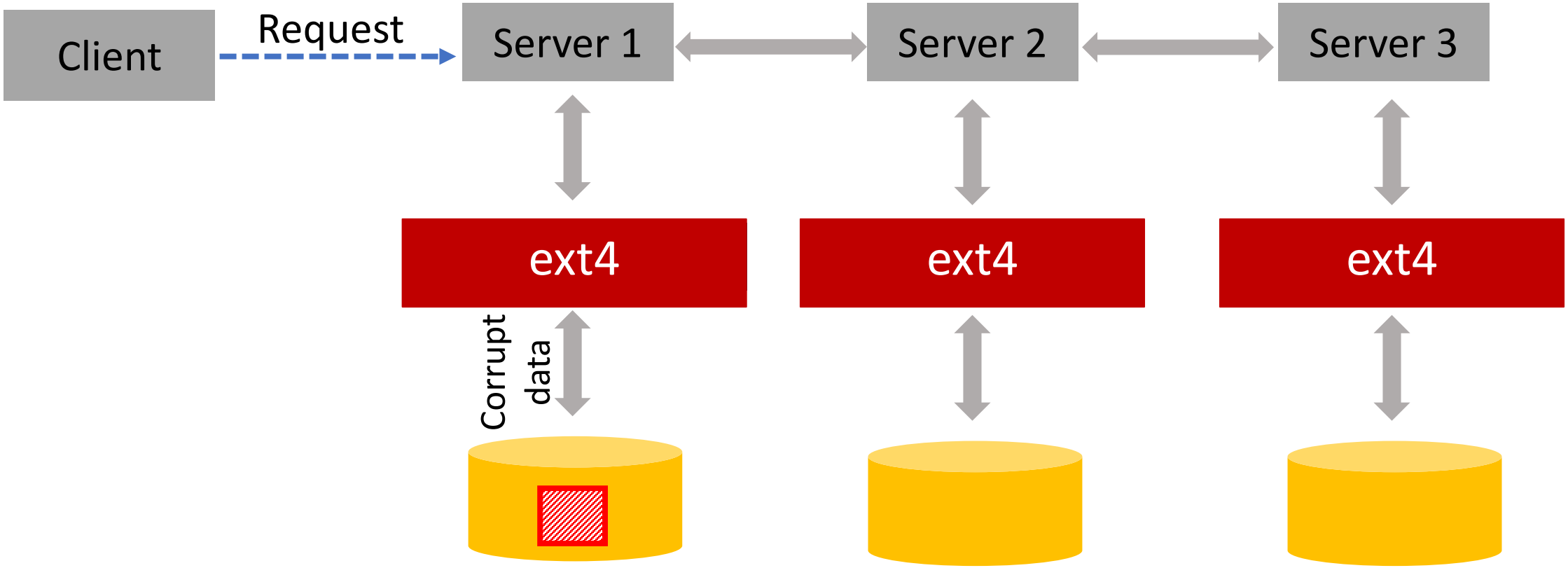
Fault Model: ext4 and btrfs



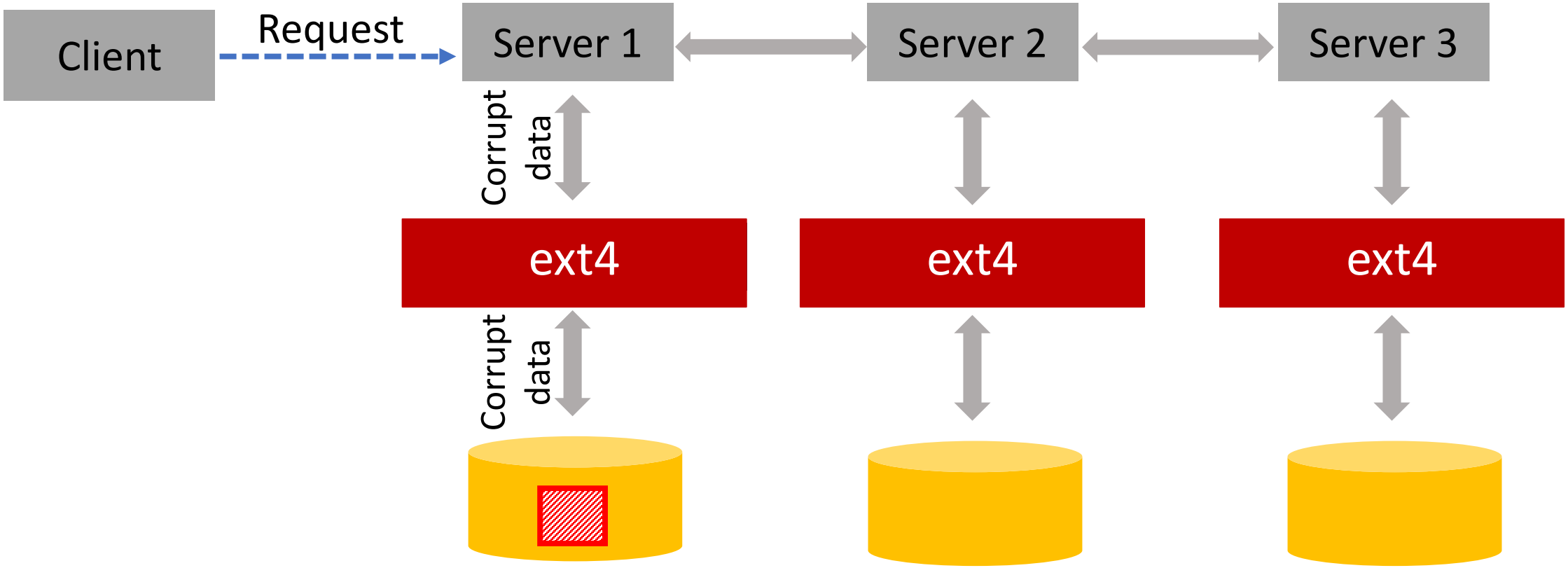
Fault Model: ext4 and btrfs



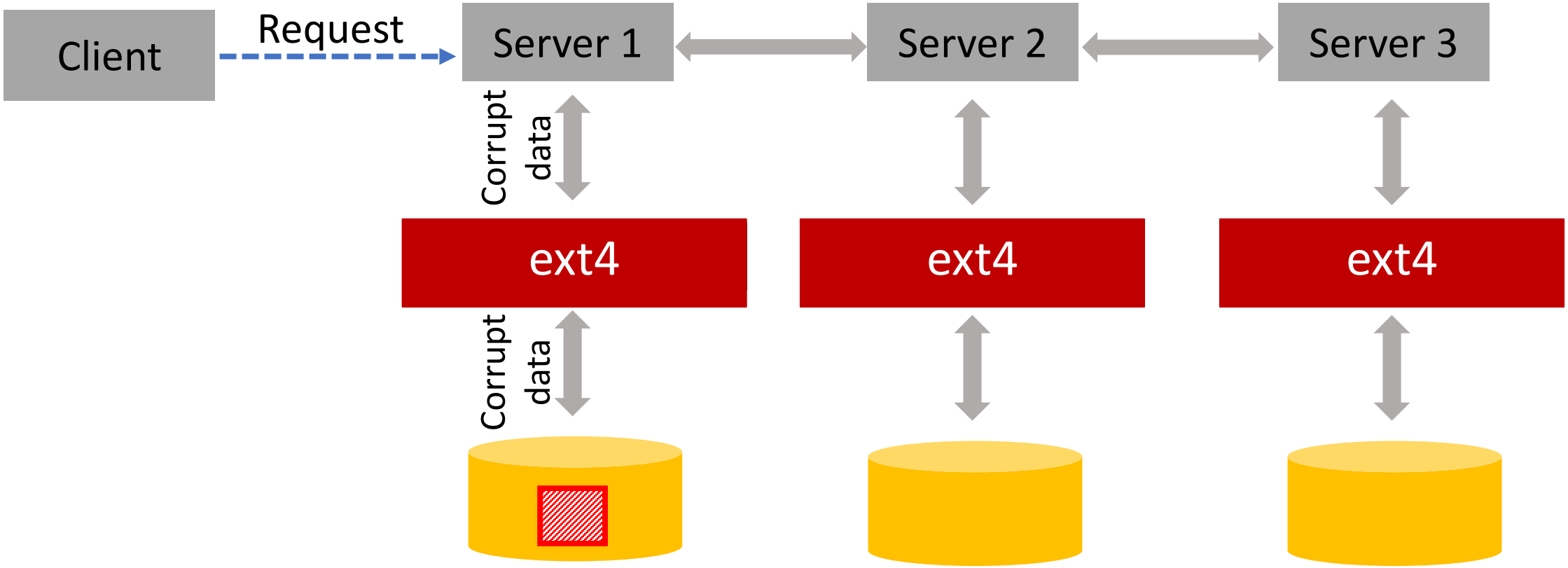
Fault Model: ext4 and btrfs



Fault Model: ext4 and btrfs

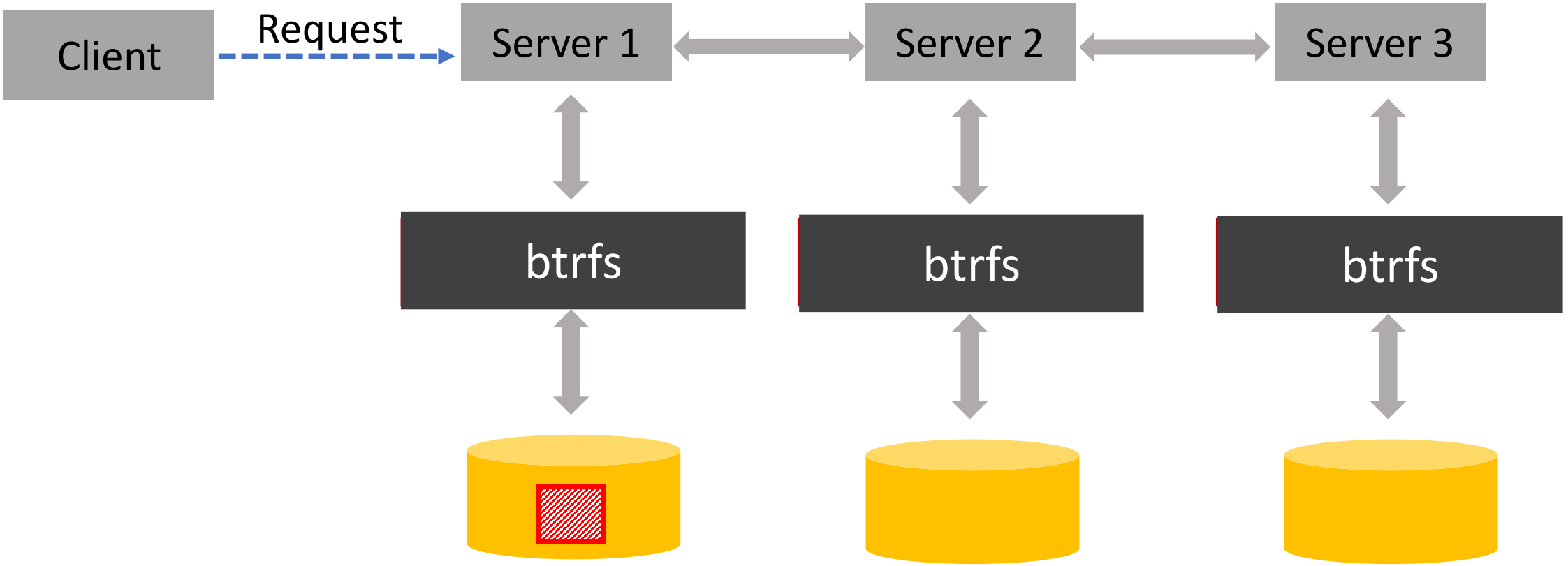


Fault Model: ext4 and btrfs



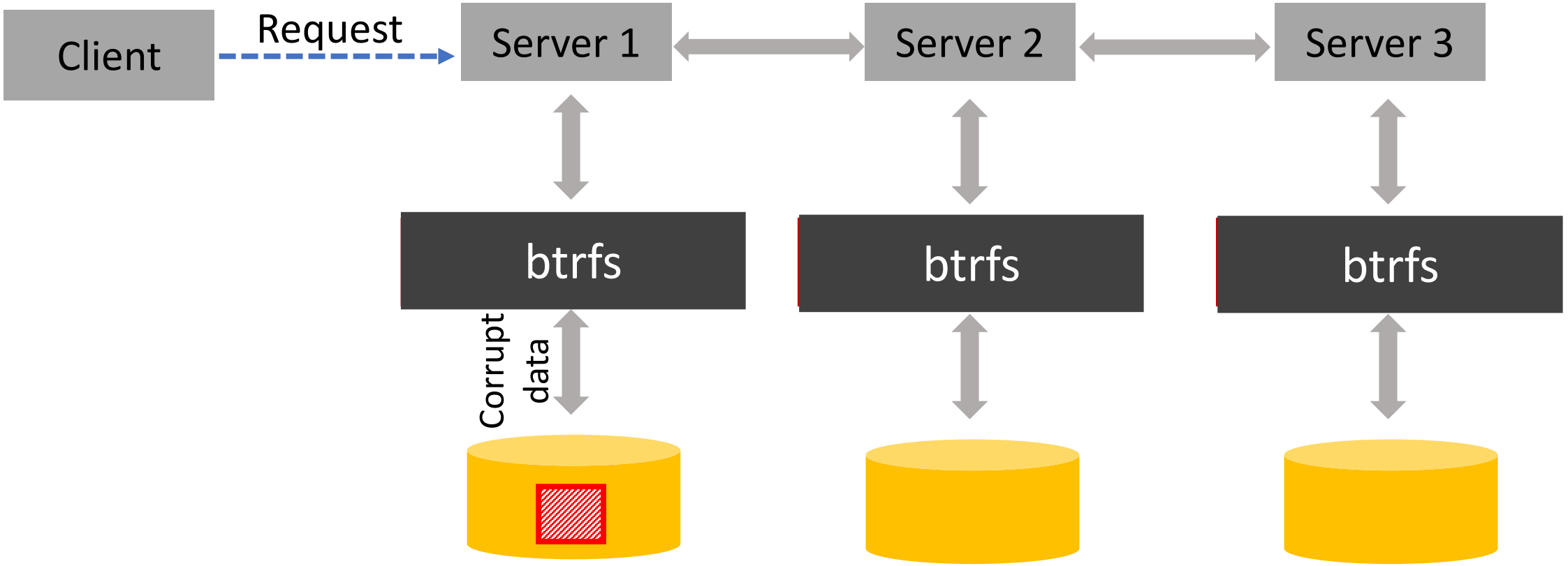
Ext4: disk corruption → corrupted data to applications

Fault Model: ext4 and btrfs



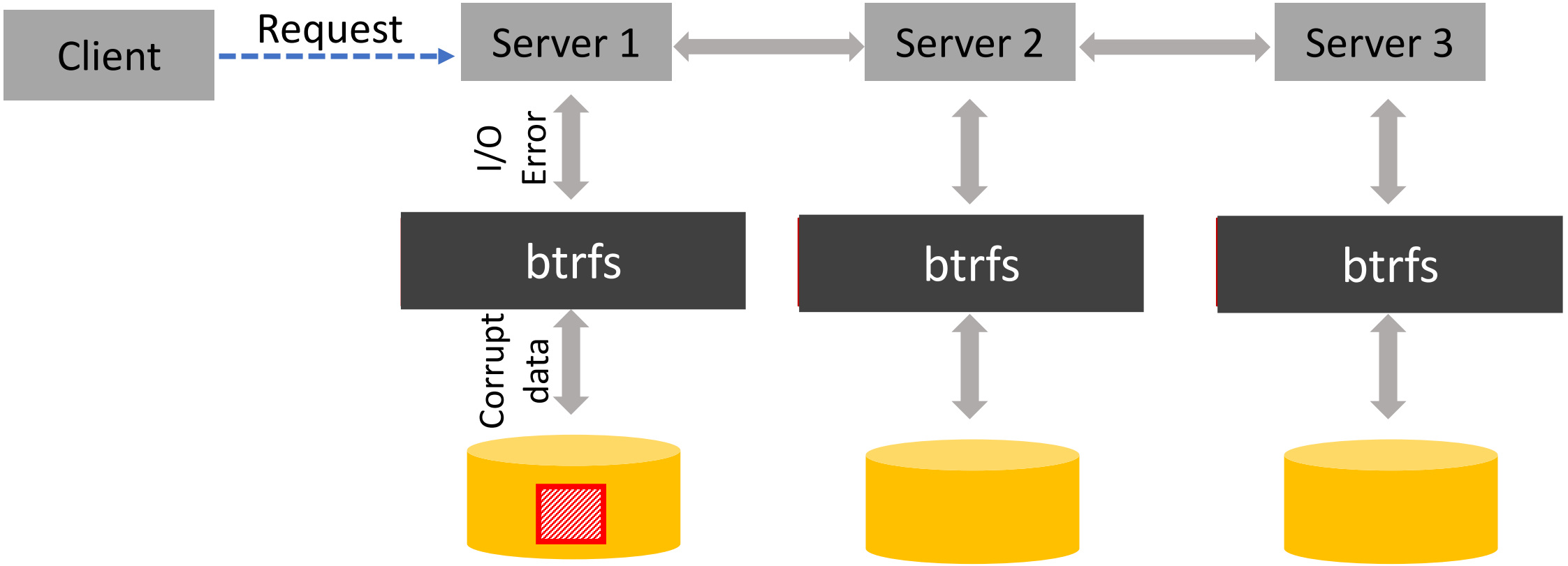
Ext4: disk corruption → corrupted data to applications

Fault Model: ext4 and btrfs



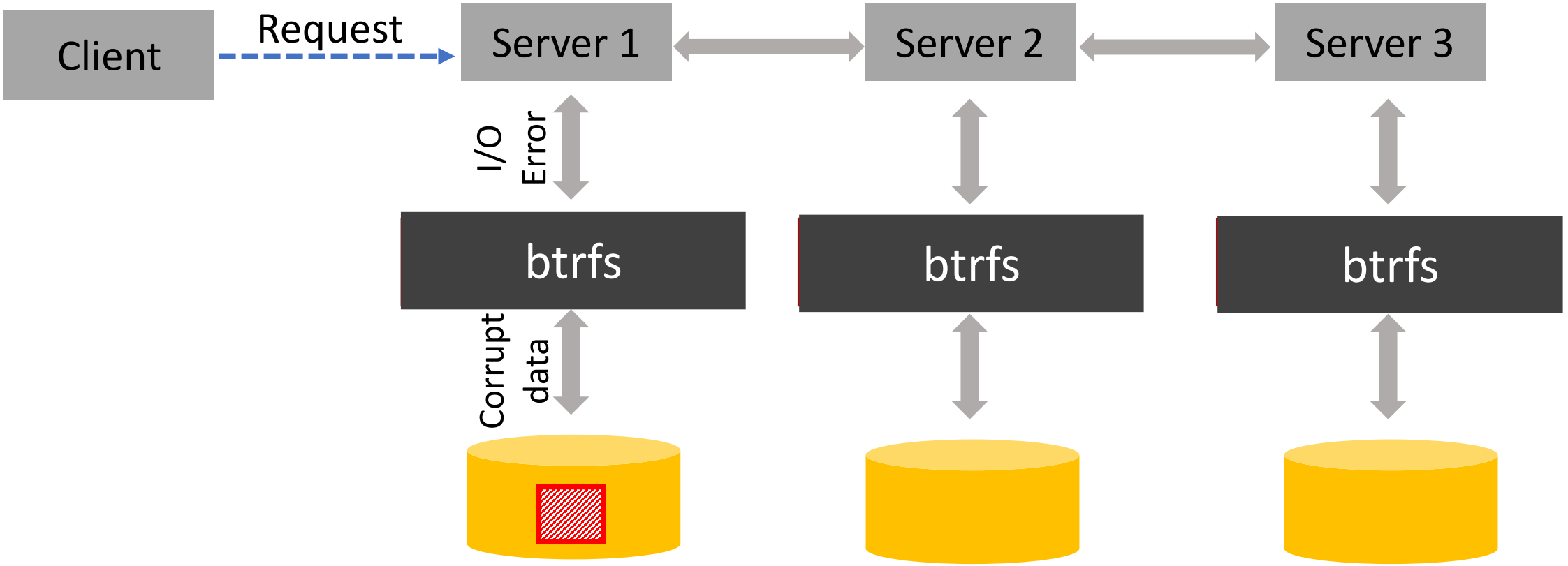
Ext4: disk corruption → corrupted data to applications

Fault Model: ext4 and btrfs



Ext4: disk corruption → corrupted data to applications

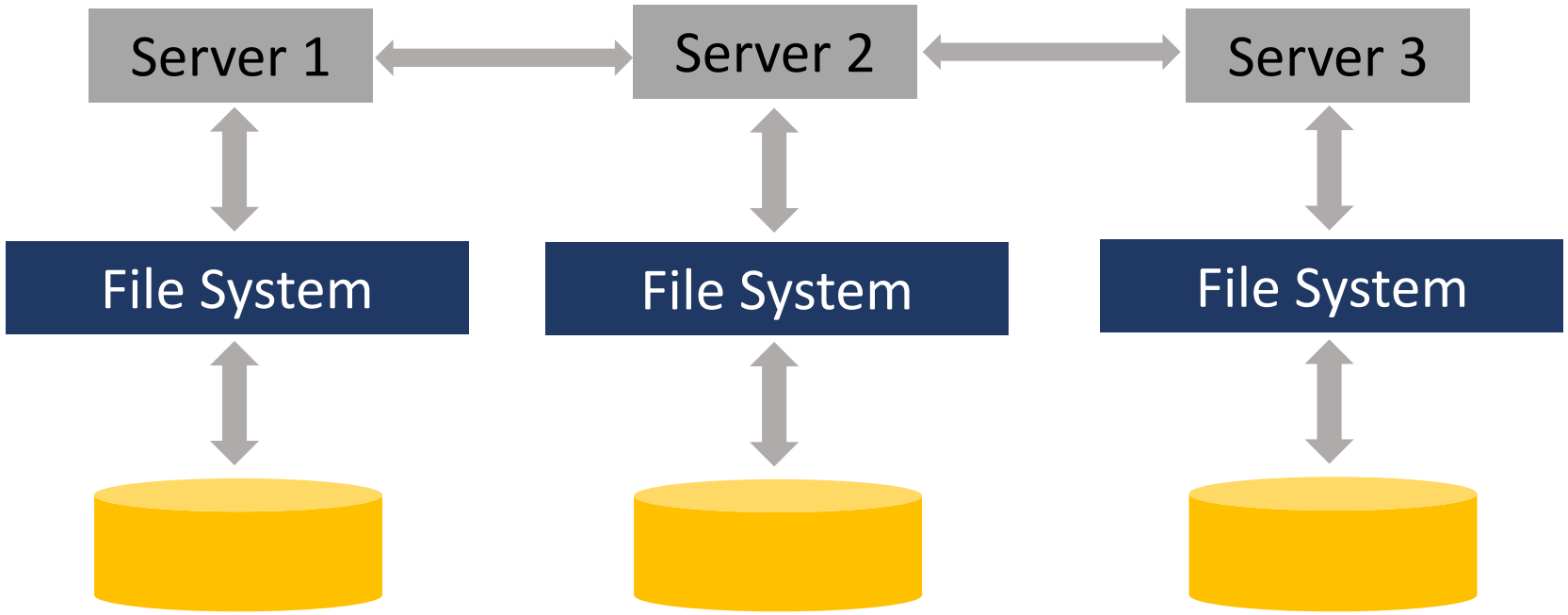
Fault Model: ext4 and btrfs



Ext4: disk corruption → corrupted data to applications

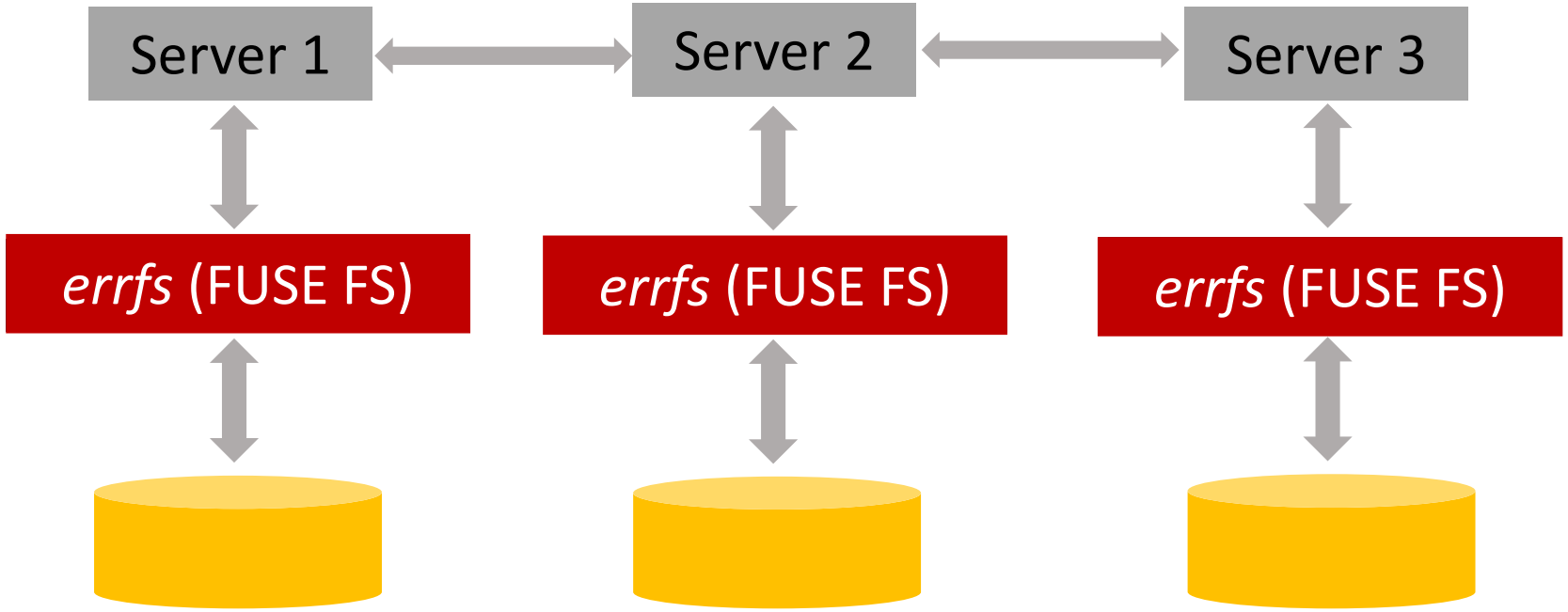
Btrfs: disk corruption → I/O error to applications

Fault Injection Methodology



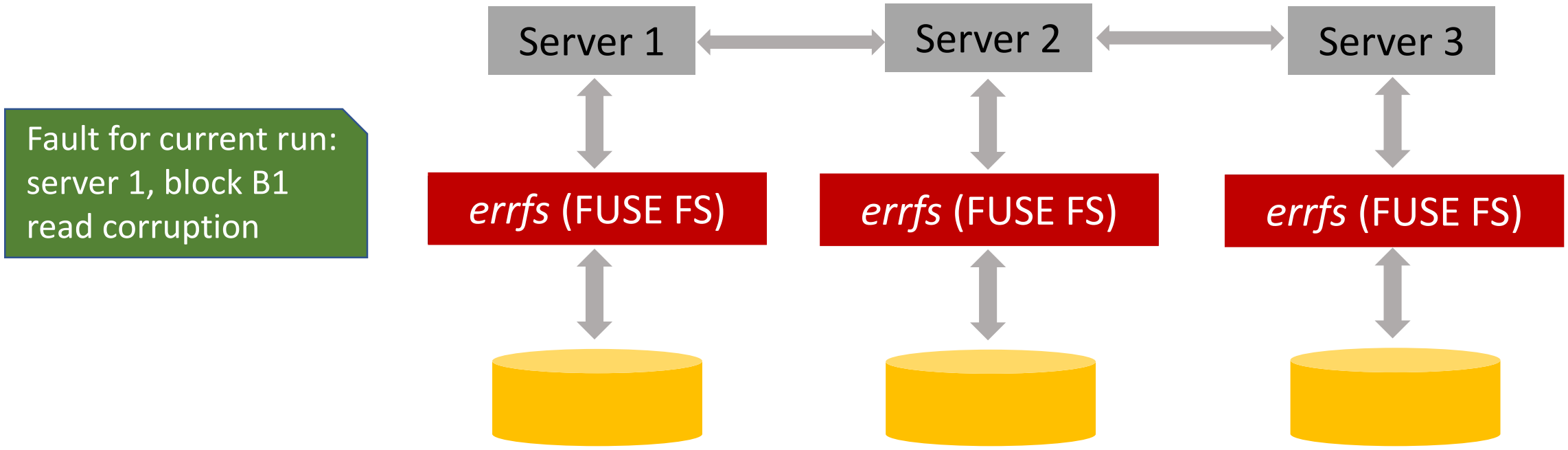
Fault Injection Methodology

errfs - a FUSE file system to inject file-system faults



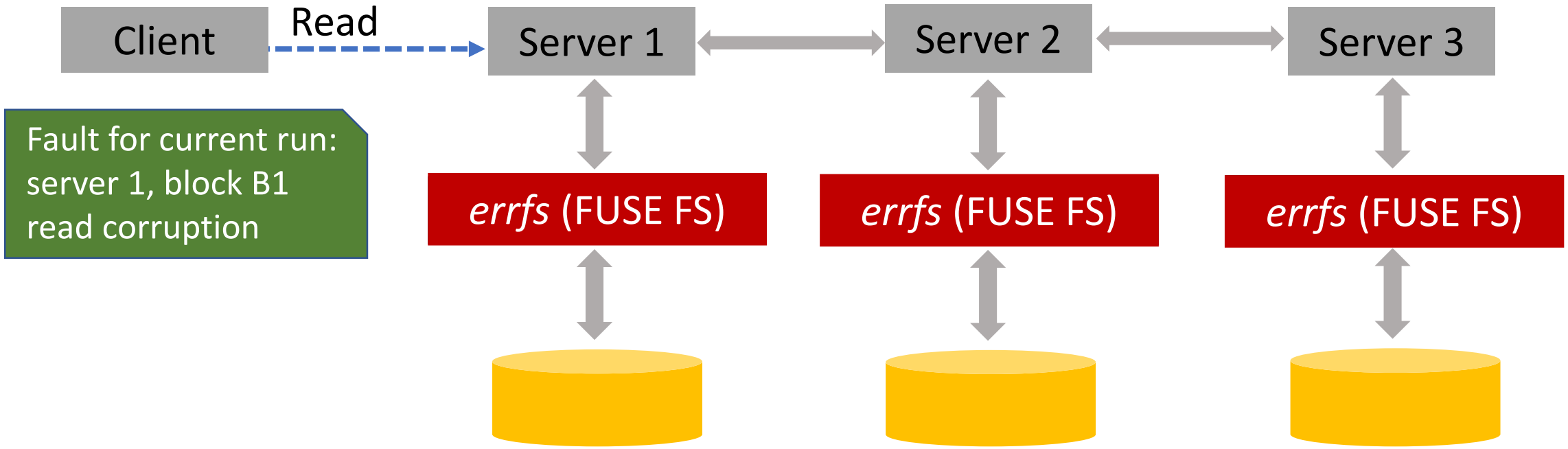
Fault Injection Methodology

errfs - a FUSE file system to inject file-system faults



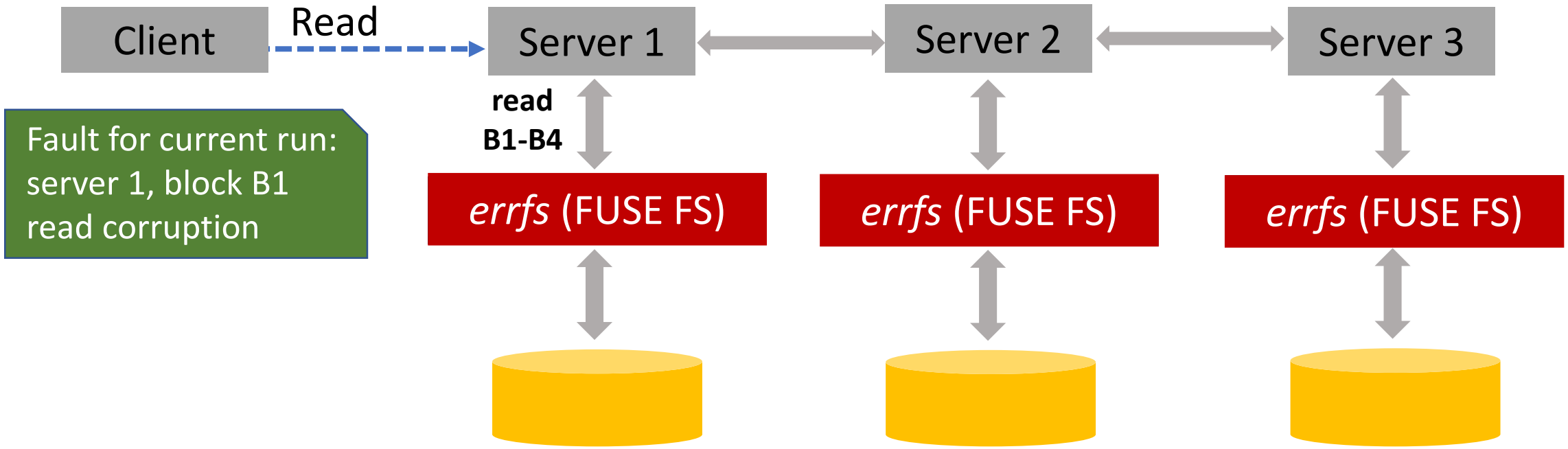
Fault Injection Methodology

errfs - a FUSE file system to inject file-system faults



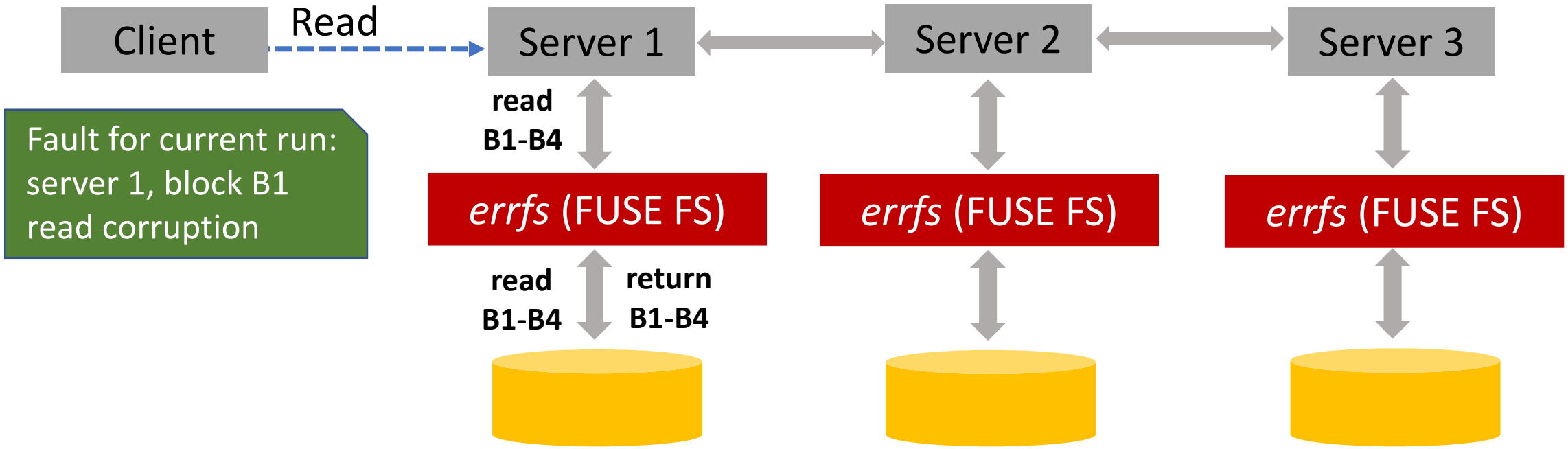
Fault Injection Methodology

errfs - a FUSE file system to inject file-system faults



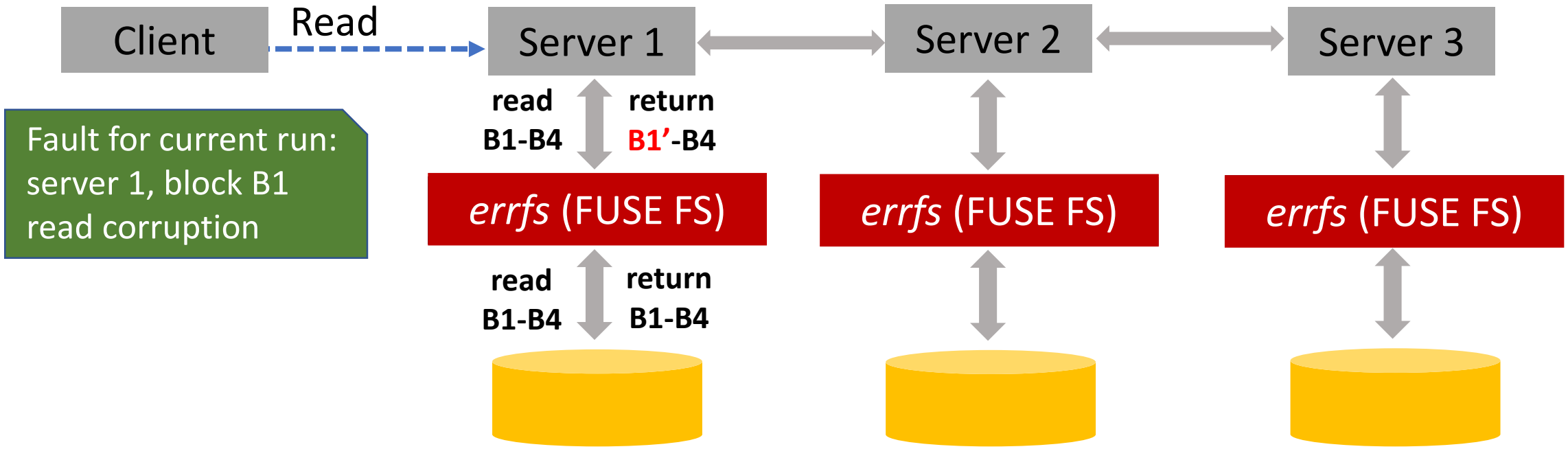
Fault Injection Methodology

errfs - a FUSE file system to inject file-system faults



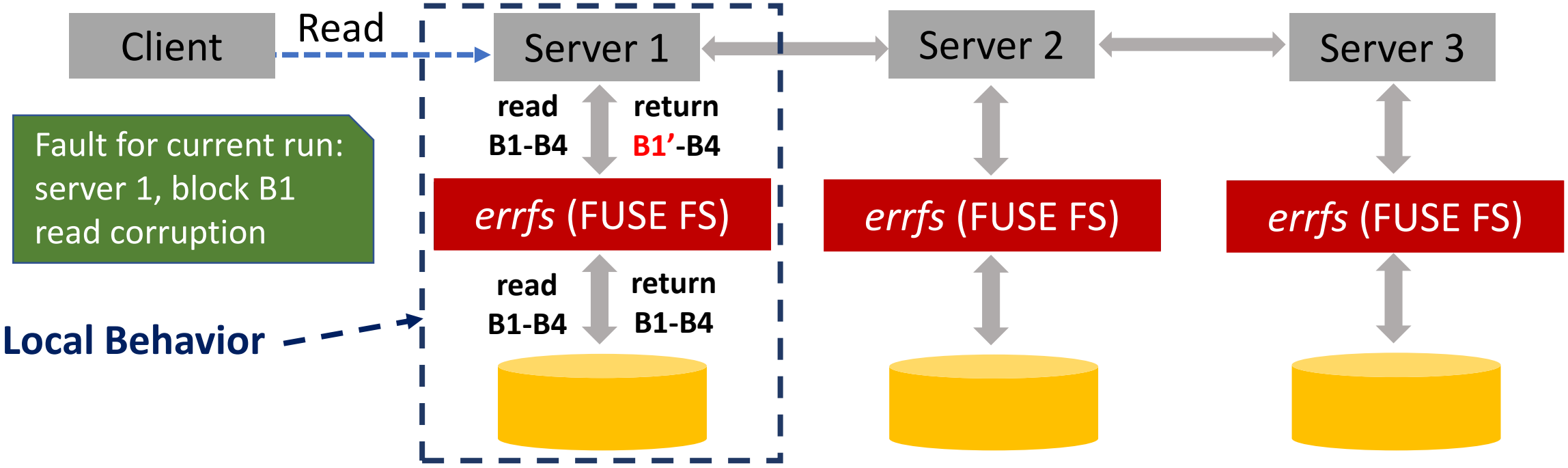
Fault Injection Methodology

errfs - a FUSE file system to inject file-system faults



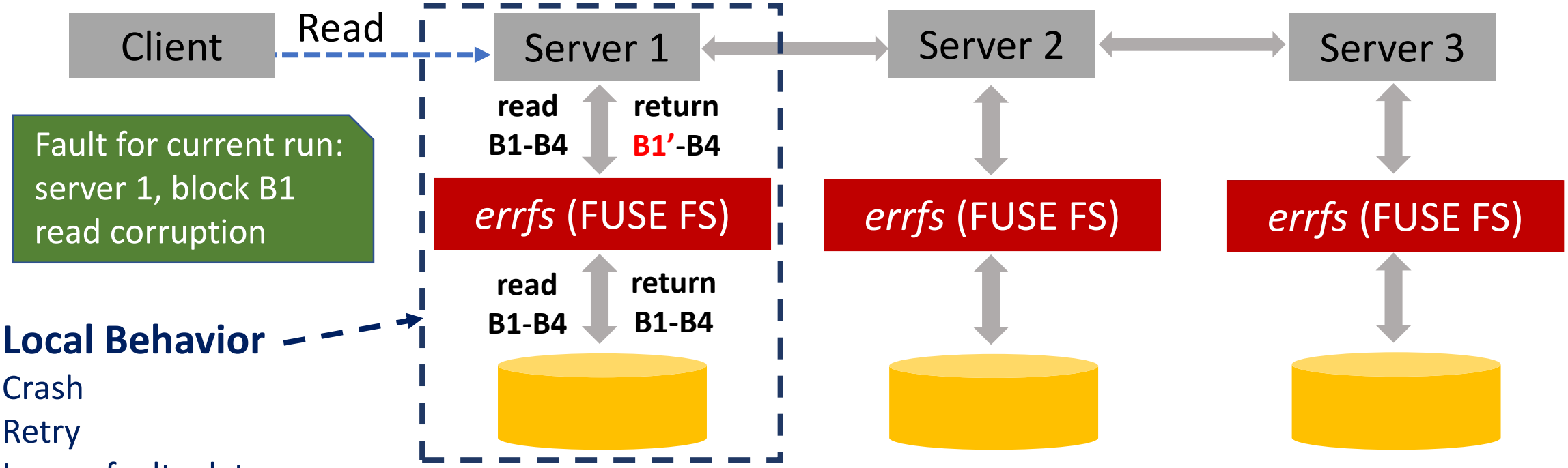
Fault Injection Methodology

errfs - a FUSE file system to inject file-system faults



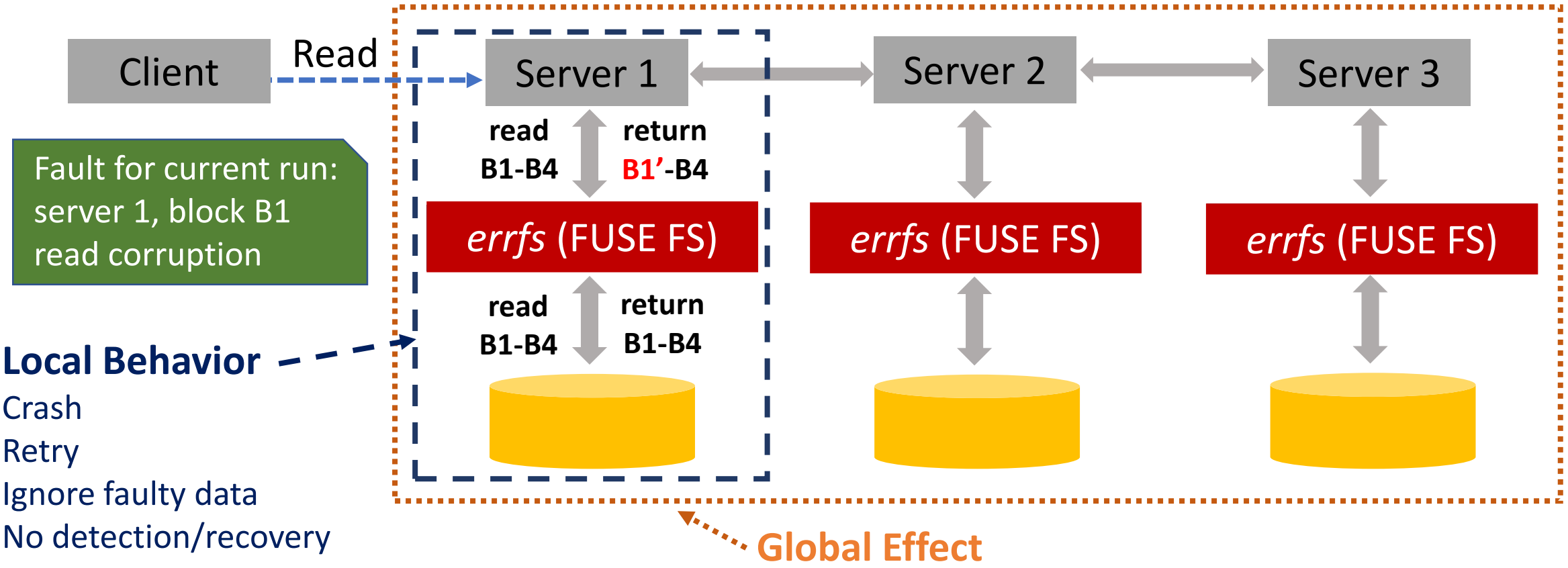
Fault Injection Methodology

errfs - a FUSE file system to inject file-system faults



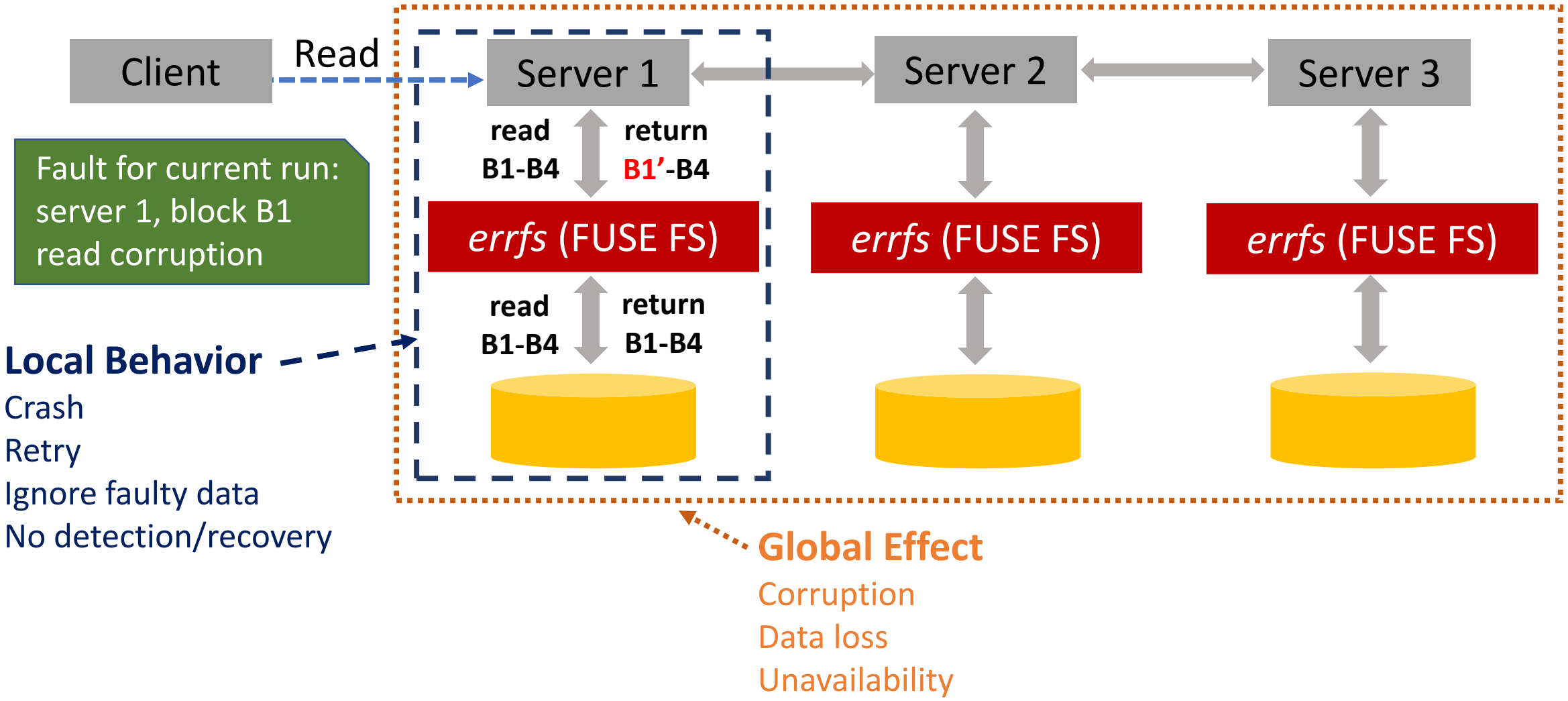
Fault Injection Methodology

errfs - a FUSE file system to inject file-system faults



Fault Injection Methodology

errfs - a FUSE file system to inject file-system faults



Outline

Introduction

Fault Injection

System Behavior Analysis

Major Results

Observations Across Systems

Conclusion

System Behavior Analysis

Behavior of eight distributed systems in response to file-system faults

Broad spectrum of replication and consensus protocols

Replicated state machines

- ZooKeeper (uses ZAB for consensus)
- LogCabin, CockroachDB, and RethinkDB (uses RAFT for consensus)

Primary backup replication

- MongoDB
- Redis
- Kafka (in-sync replicas for leader election)

Dynamo-style quorum

- Cassandra (decentralized, no leader/follower)

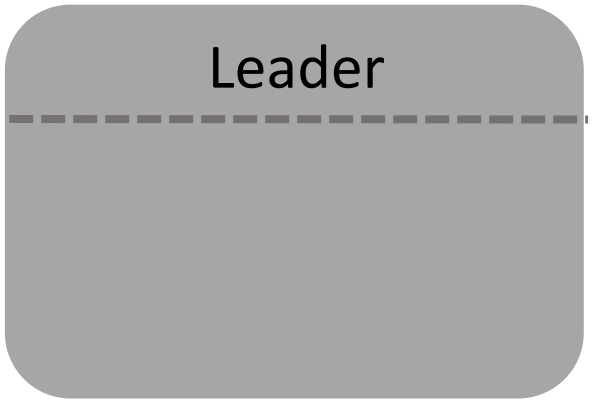
An Example: Redis

An Example: Redis

Redis is a popular data structure store

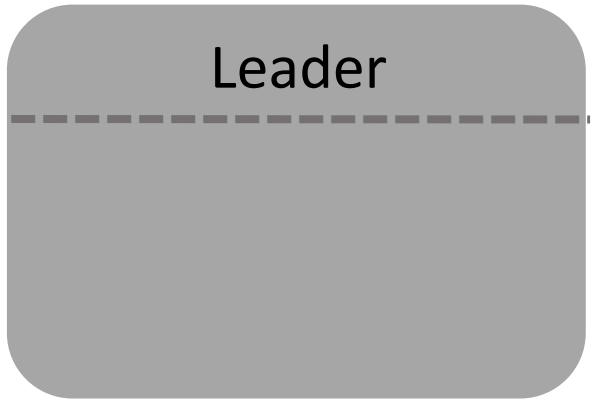
An Example: Redis

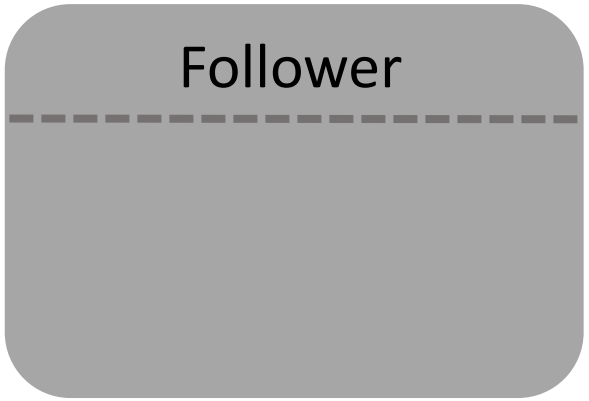
Redis is a popular data structure store

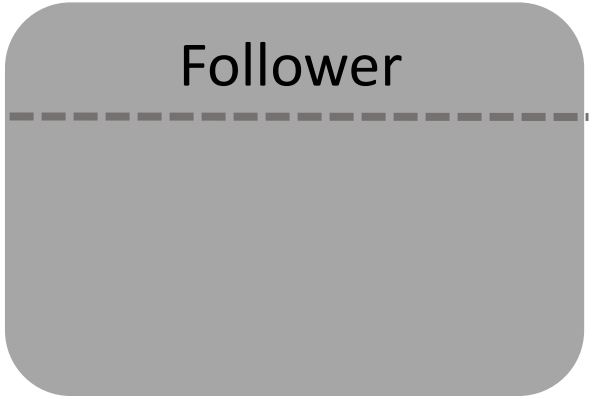


An Example: Redis

Redis is a popular data structure store

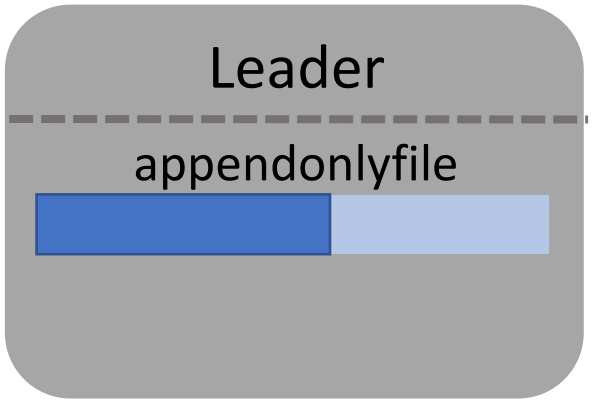


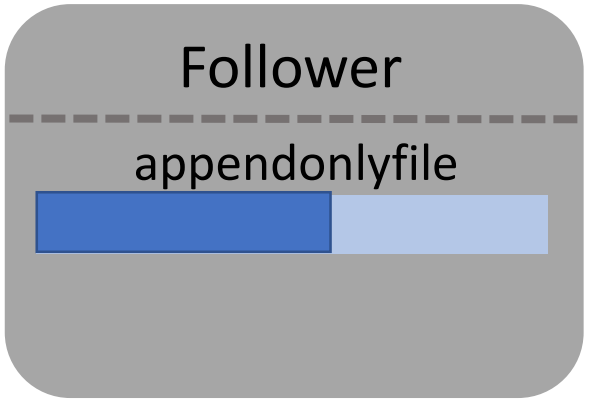


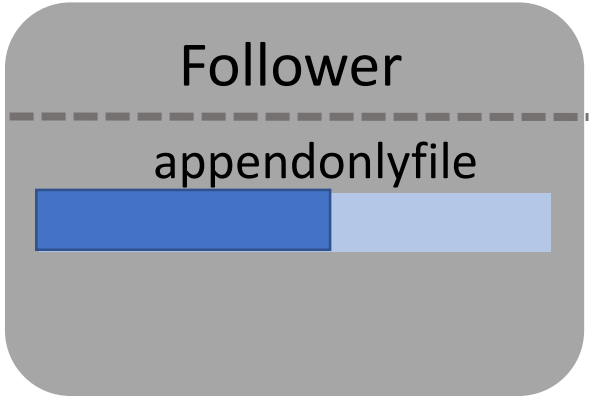


An Example: Redis

Redis is a popular data structure store

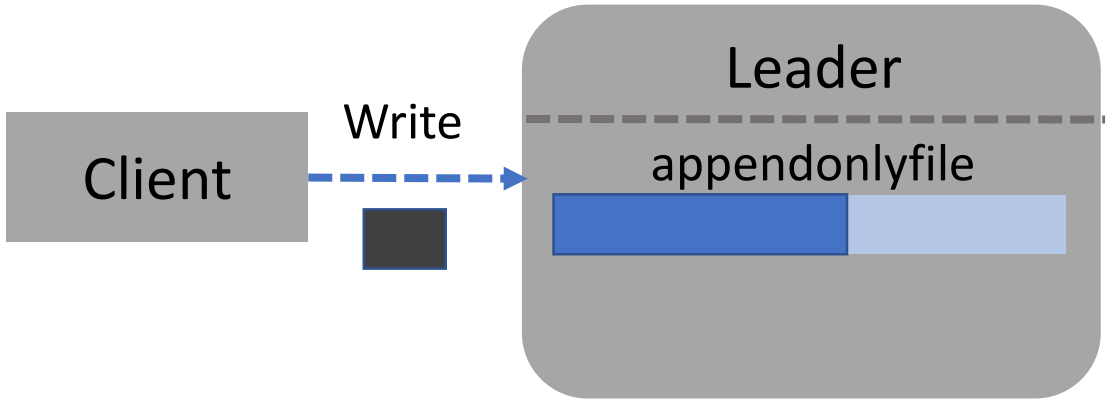


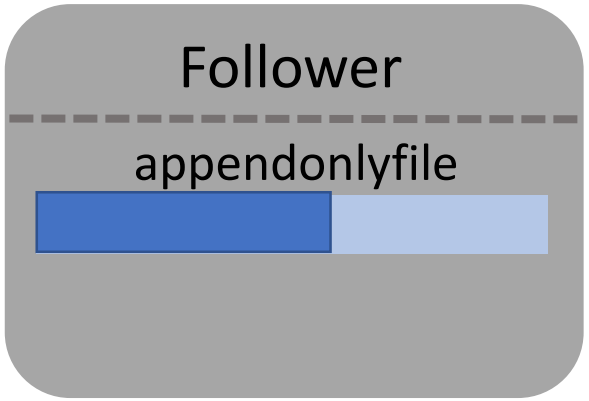


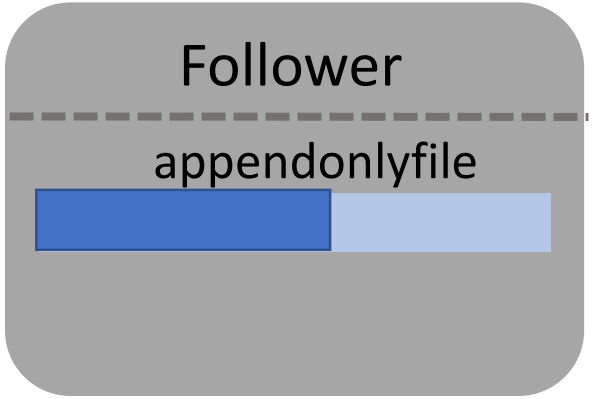


An Example: Redis

Redis is a popular data structure store

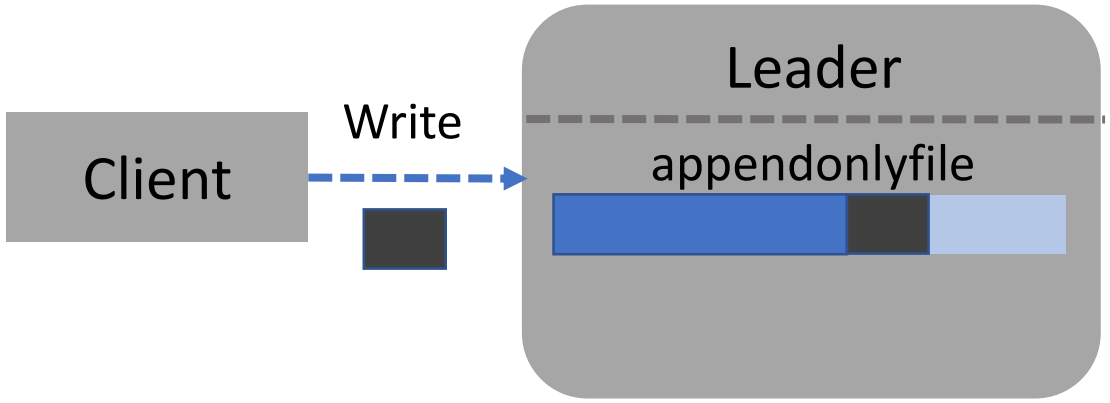


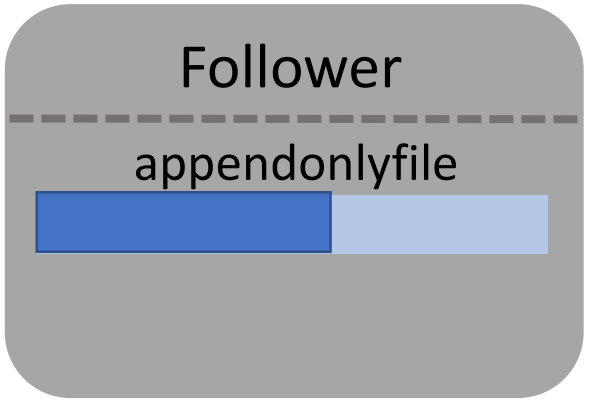


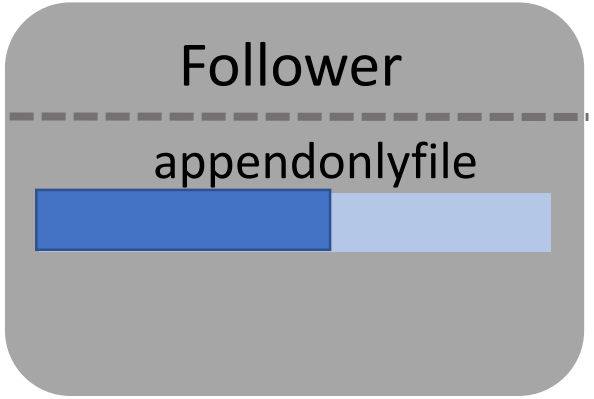


An Example: Redis

Redis is a popular data structure store

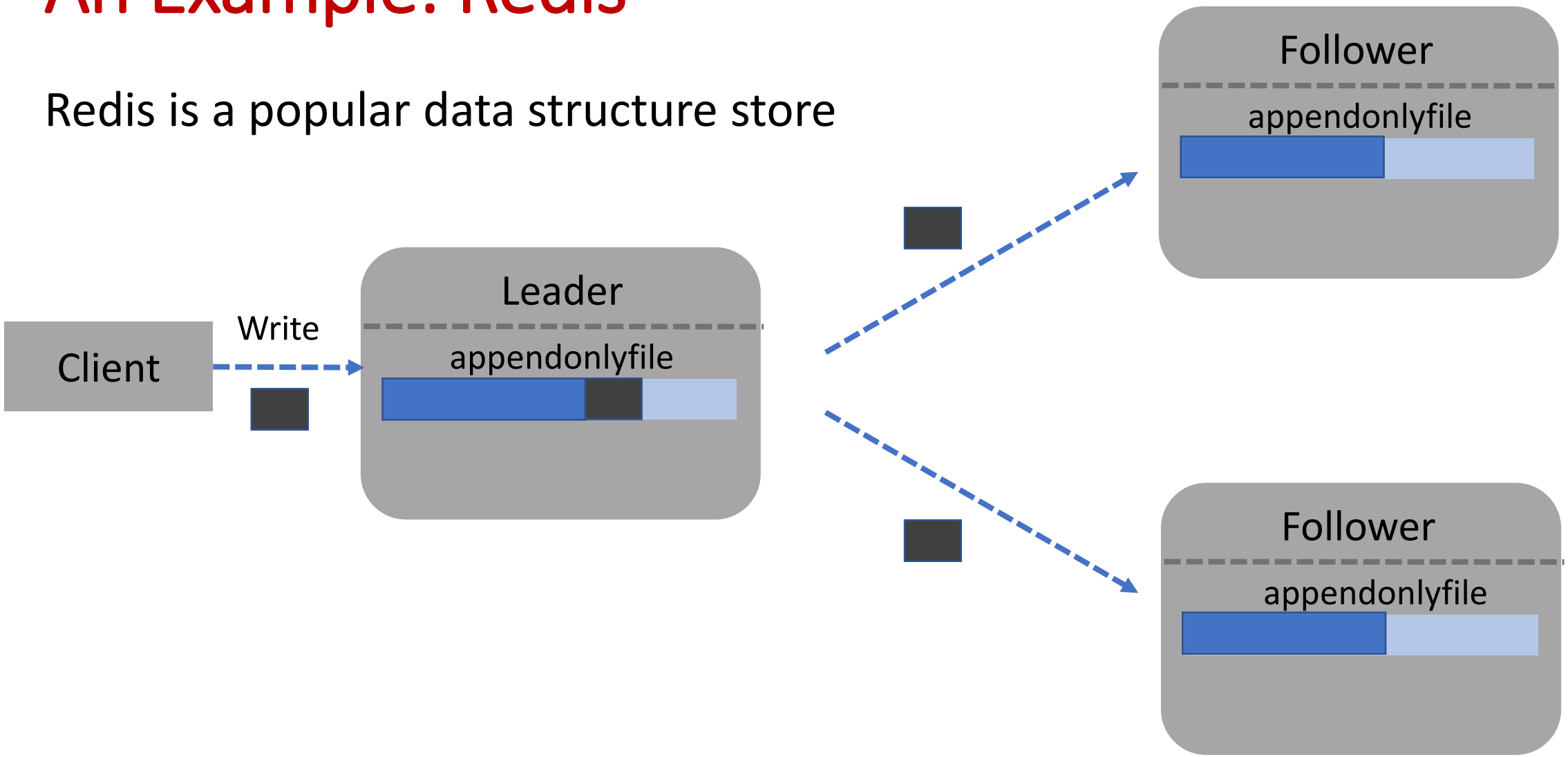






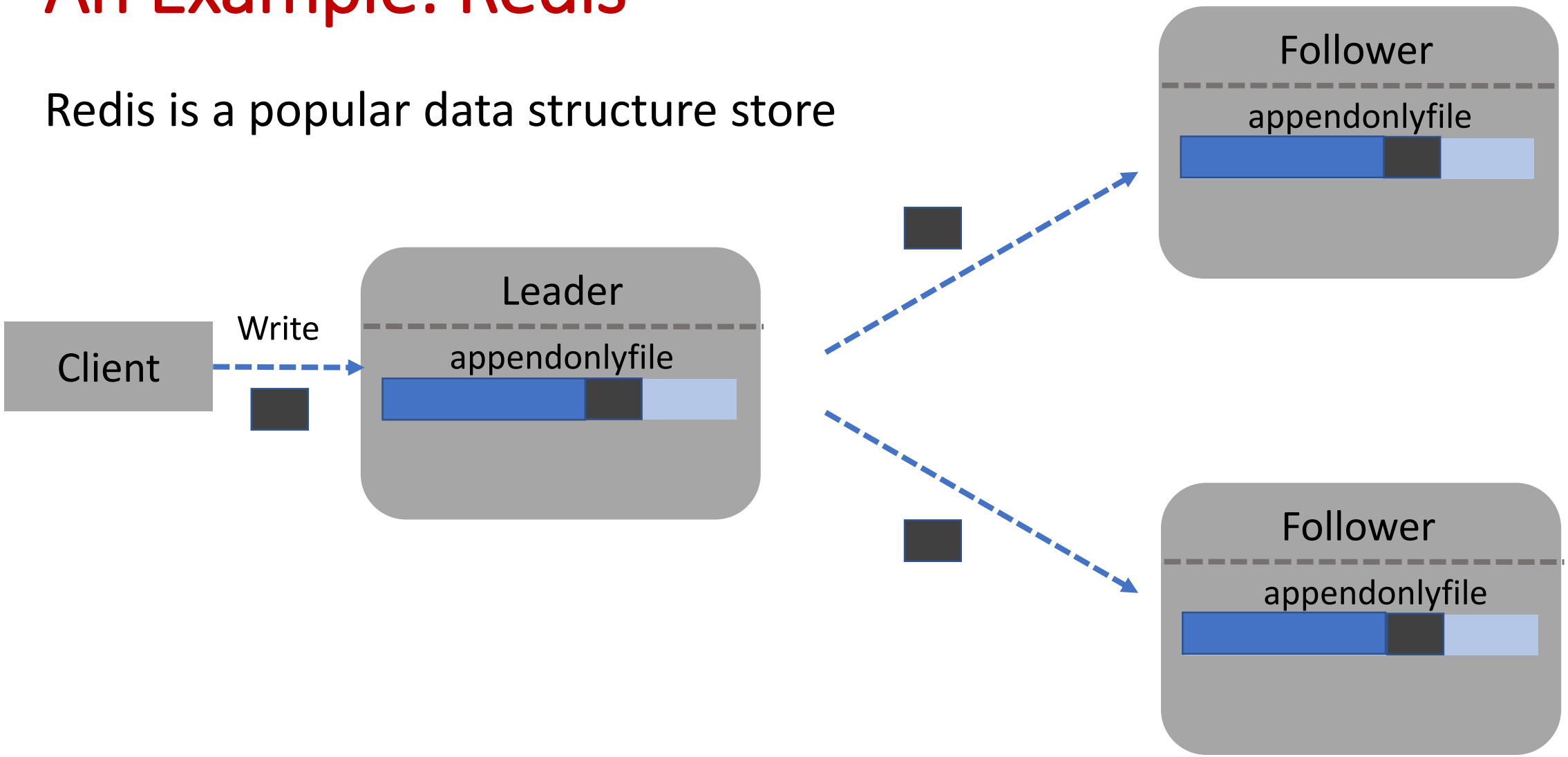
An Example: Redis

Redis is a popular data structure store



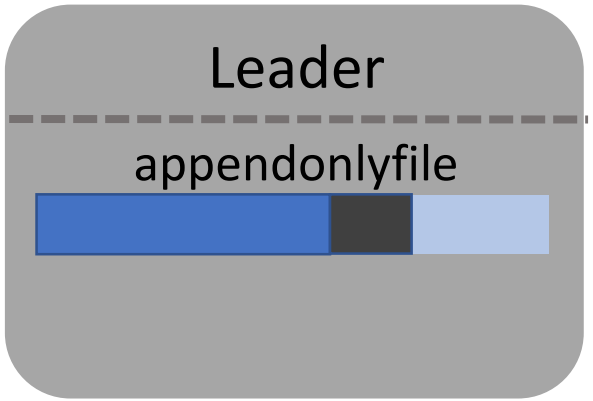
An Example: Redis

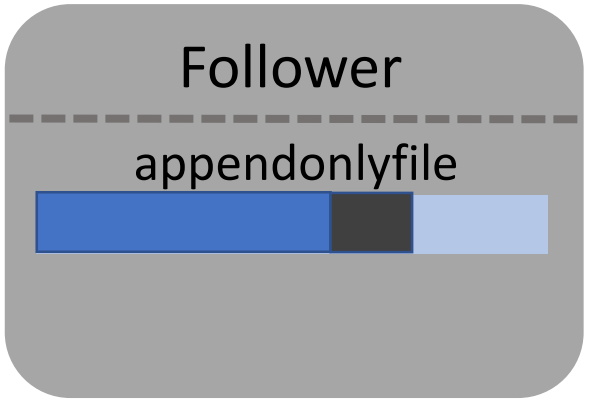
Redis is a popular data structure store

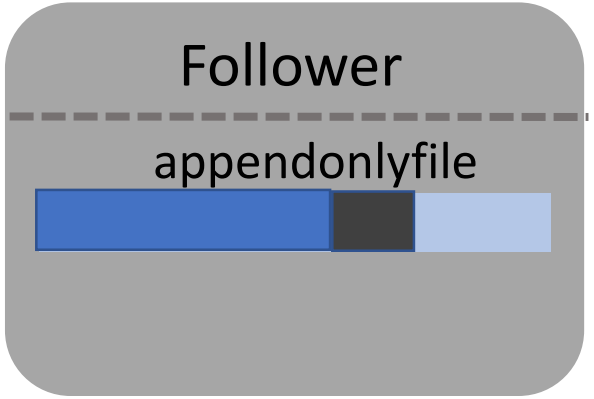


An Example: Redis

Redis is a popular data structure store

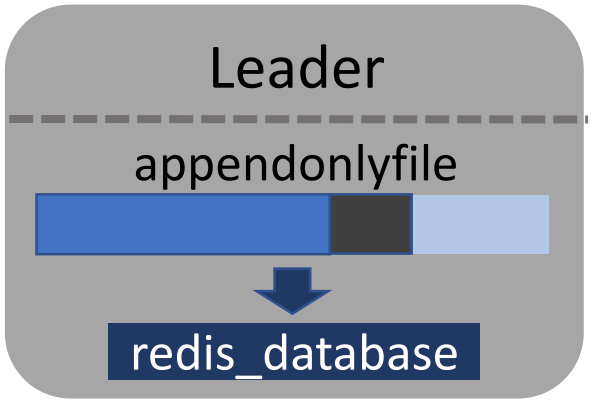


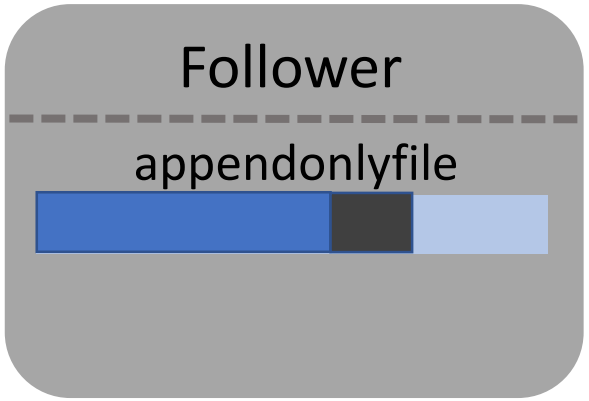


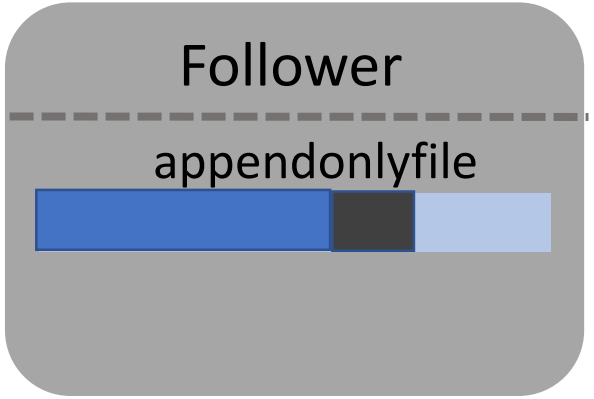


An Example: Redis

Redis is a popular data structure store

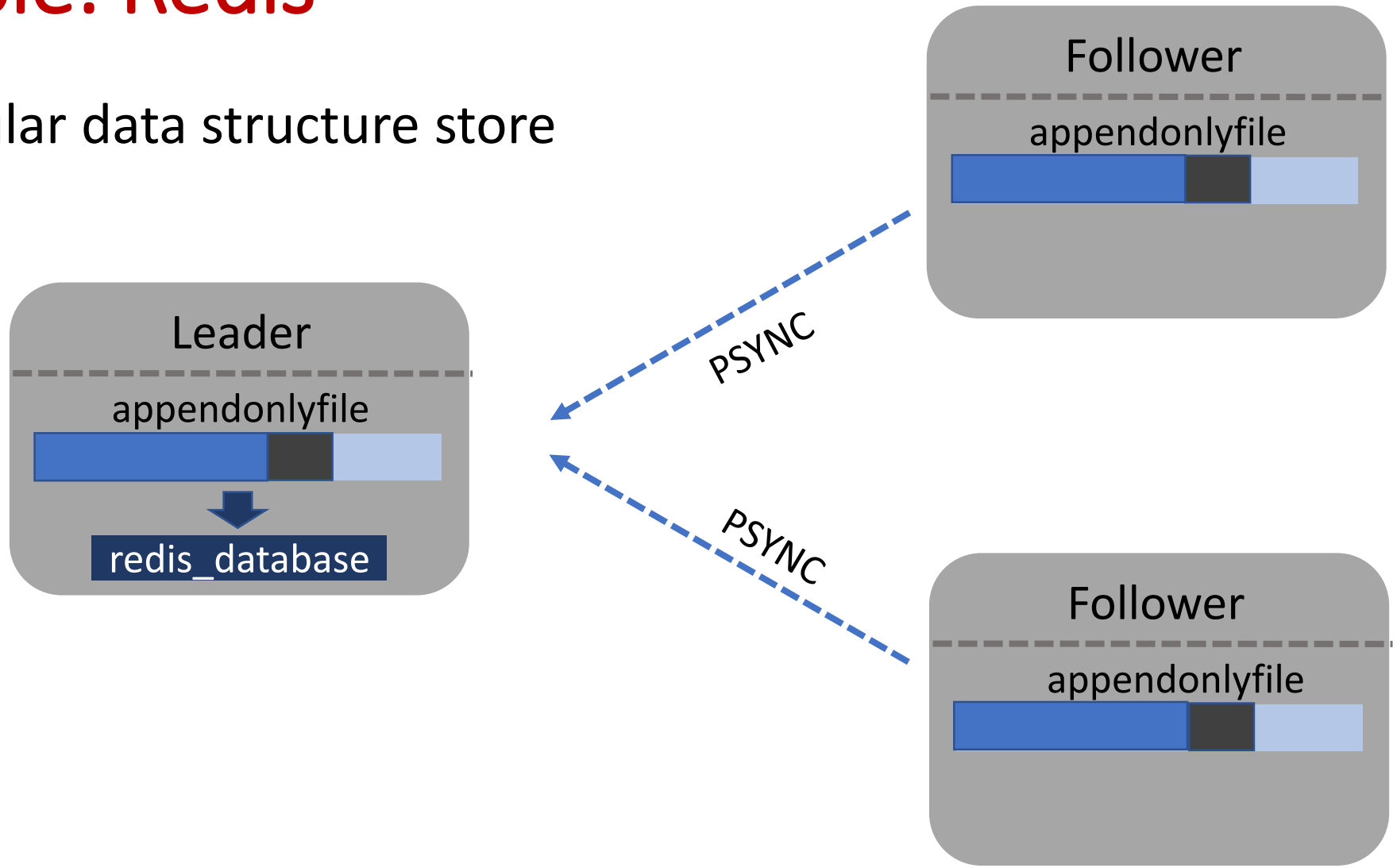






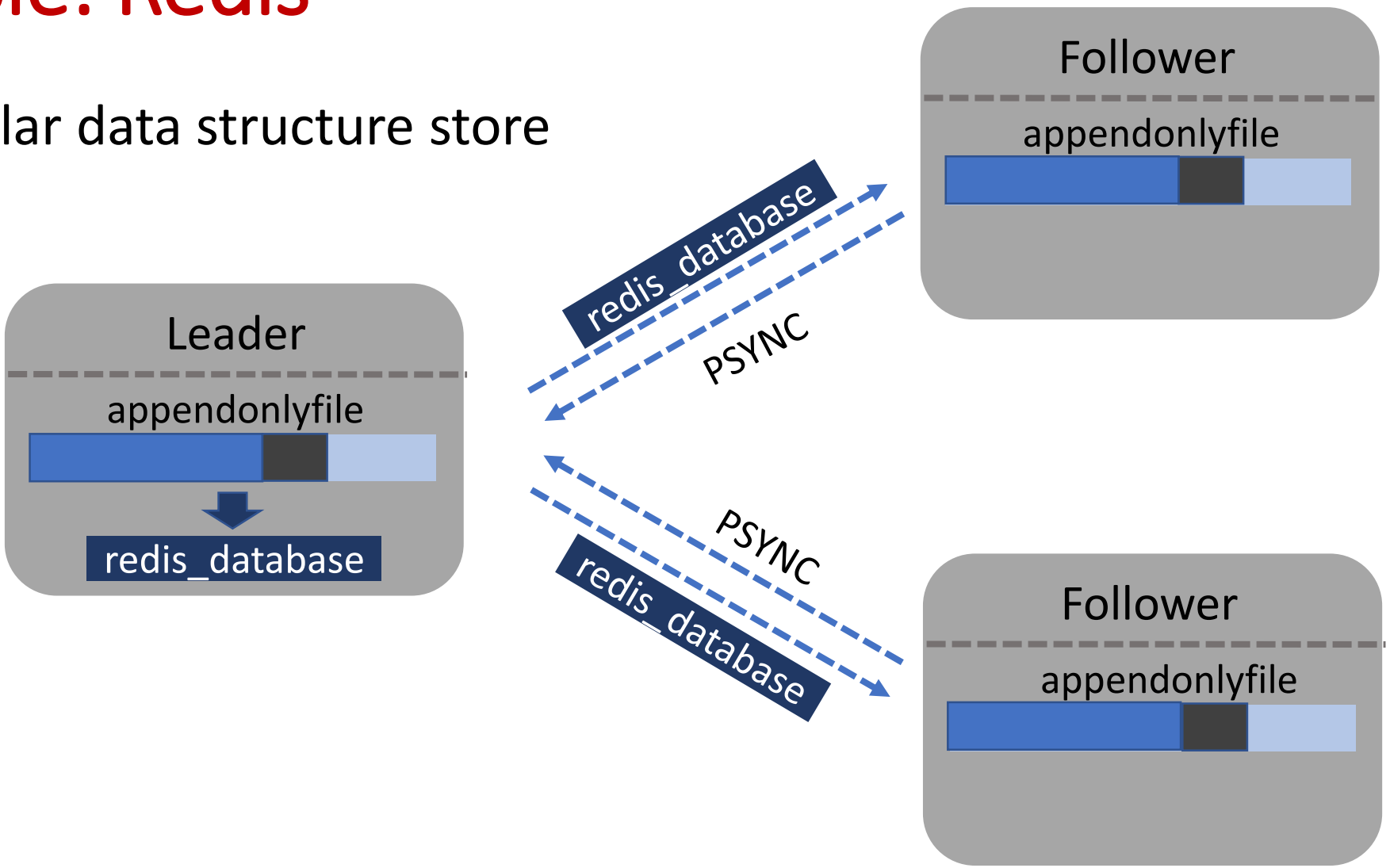
An Example: Redis

Redis is a popular data structure store



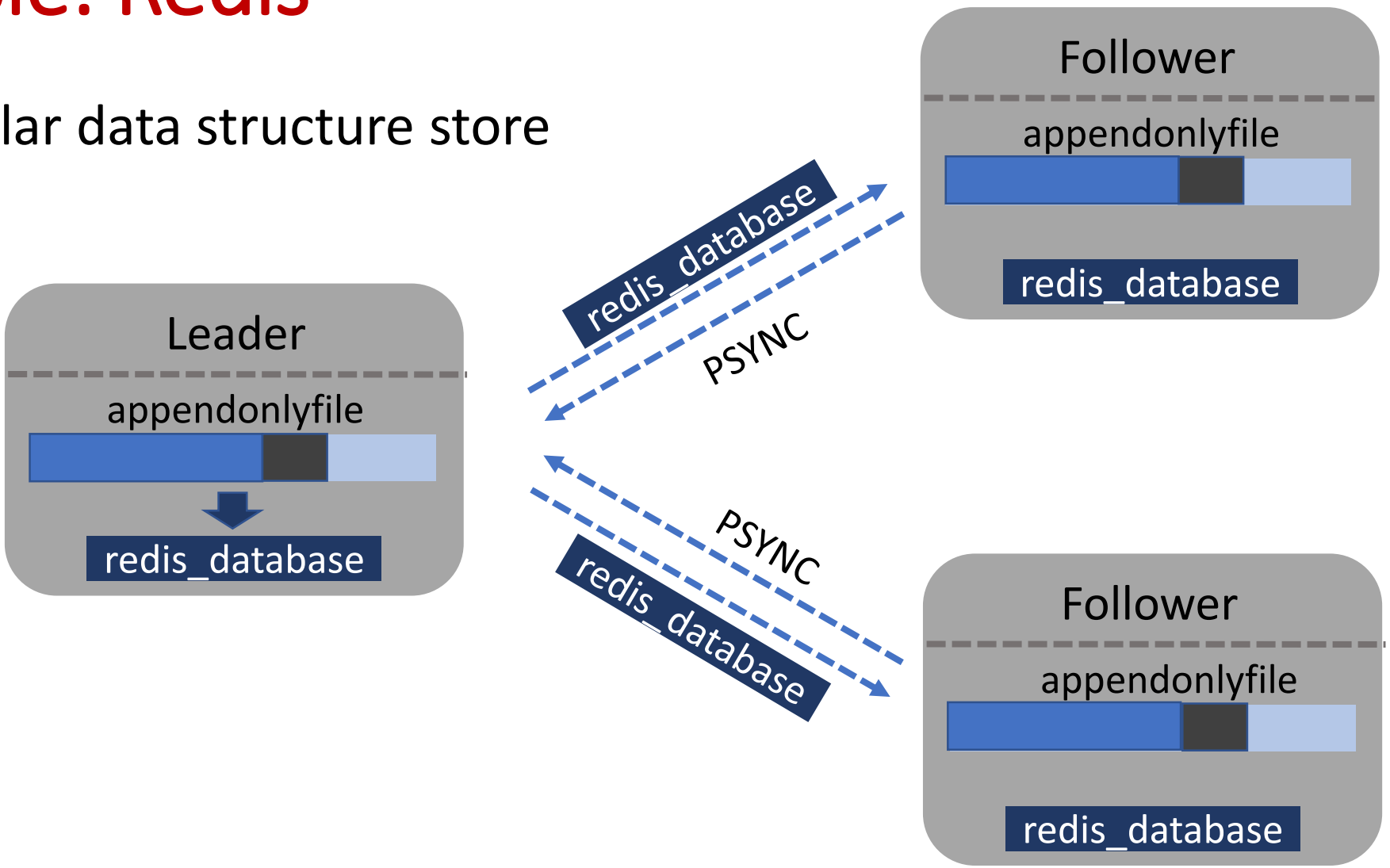
An Example: Redis

Redis is a popular data structure store



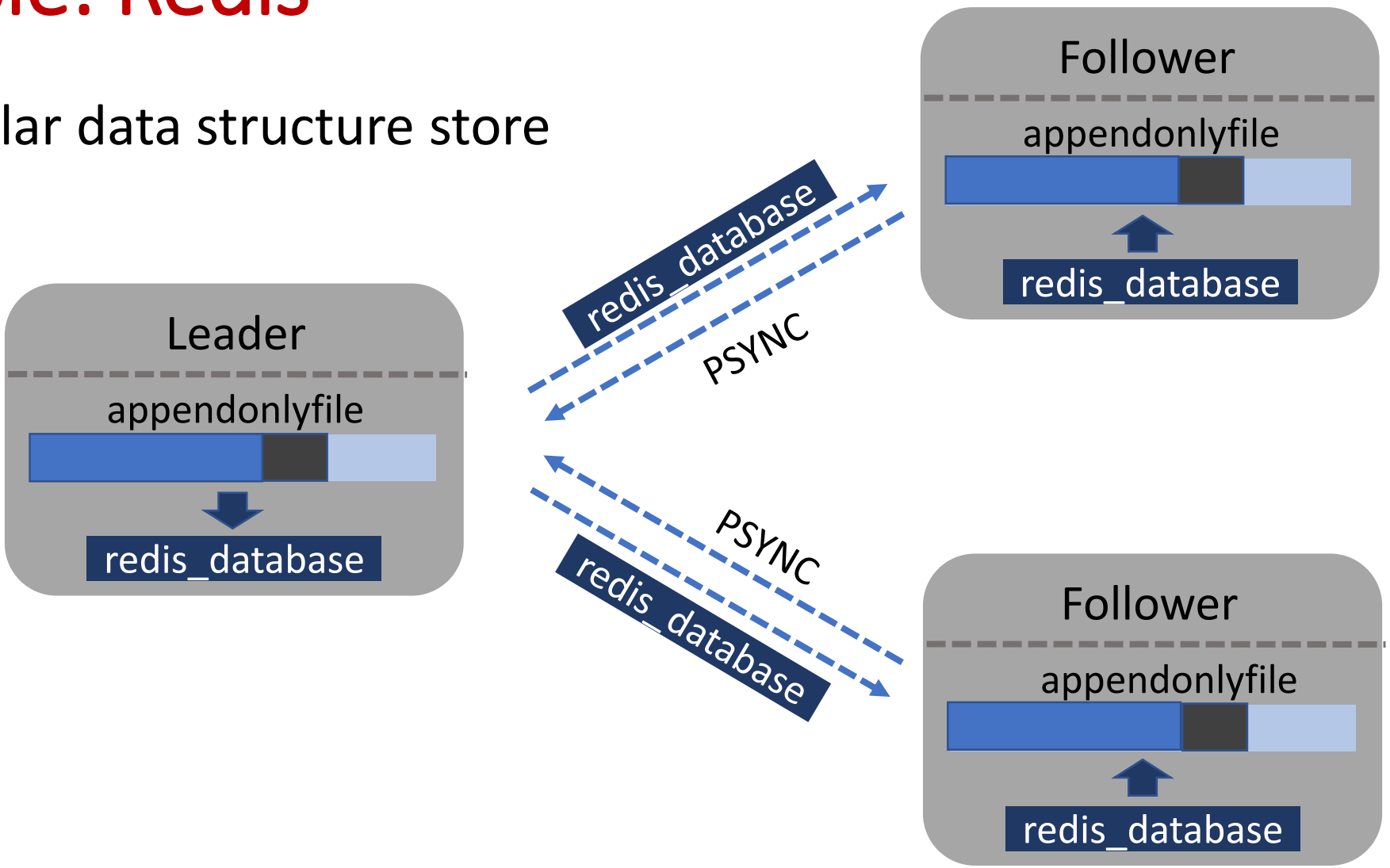
An Example: Redis

Redis is a popular data structure store



An Example: Redis

Redis is a popular data structure store



Redis: Behavior Analysis

Redis: Behavior Analysis

Read Workload

Redis: Behavior Analysis

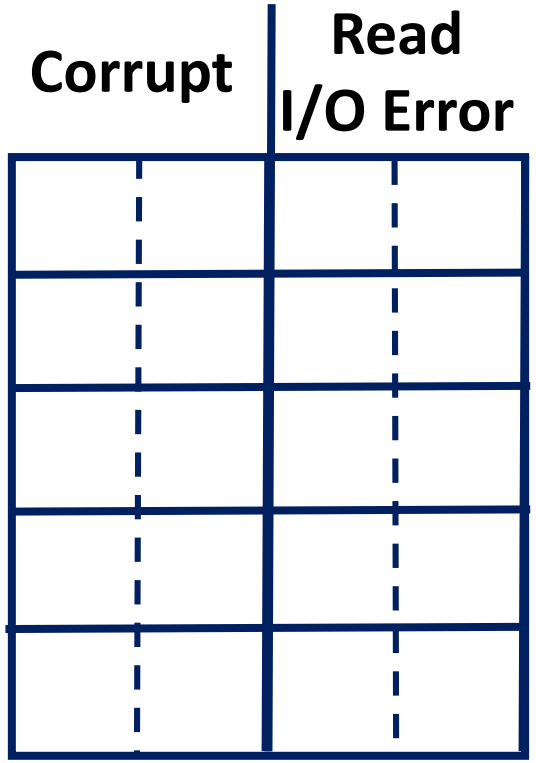
Read Workload

Local Behavior

Redis: Behavior Analysis

Read Workload

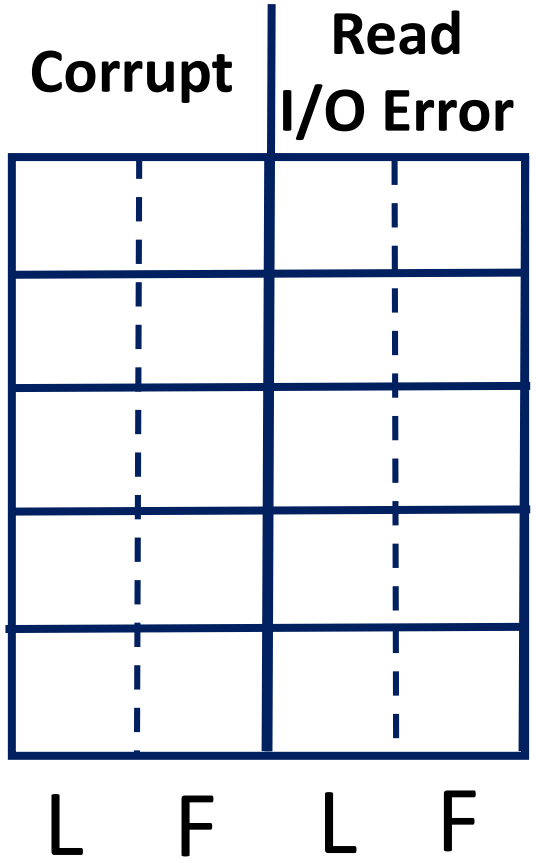
Local Behavior



Redis: Behavior Analysis

Read Workload

Local Behavior

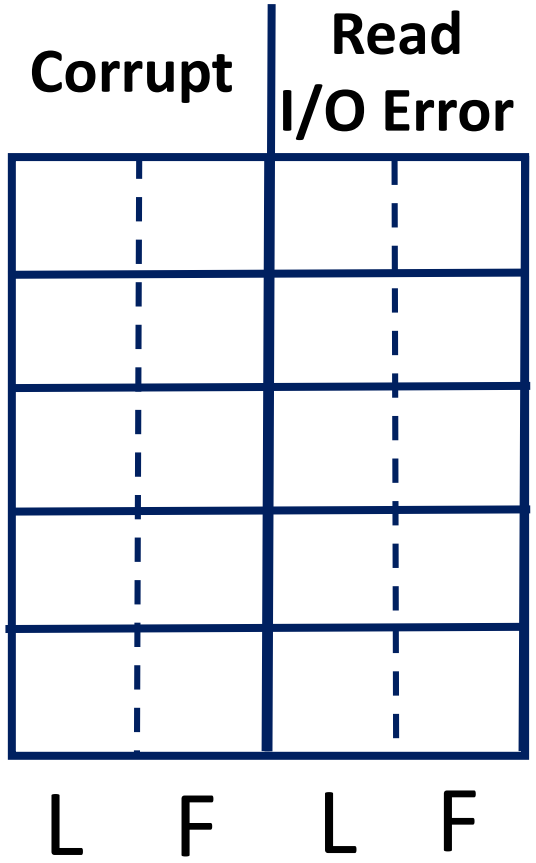


Redis: Behavior Analysis

L Leader

Read Workload

Local Behavior



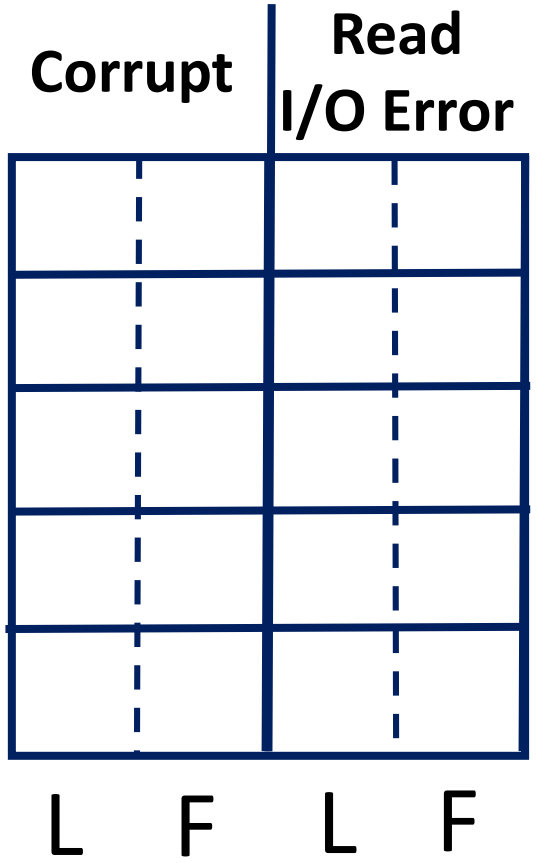
Redis: Behavior Analysis

L Leader

F Follower

Read Workload

Local Behavior



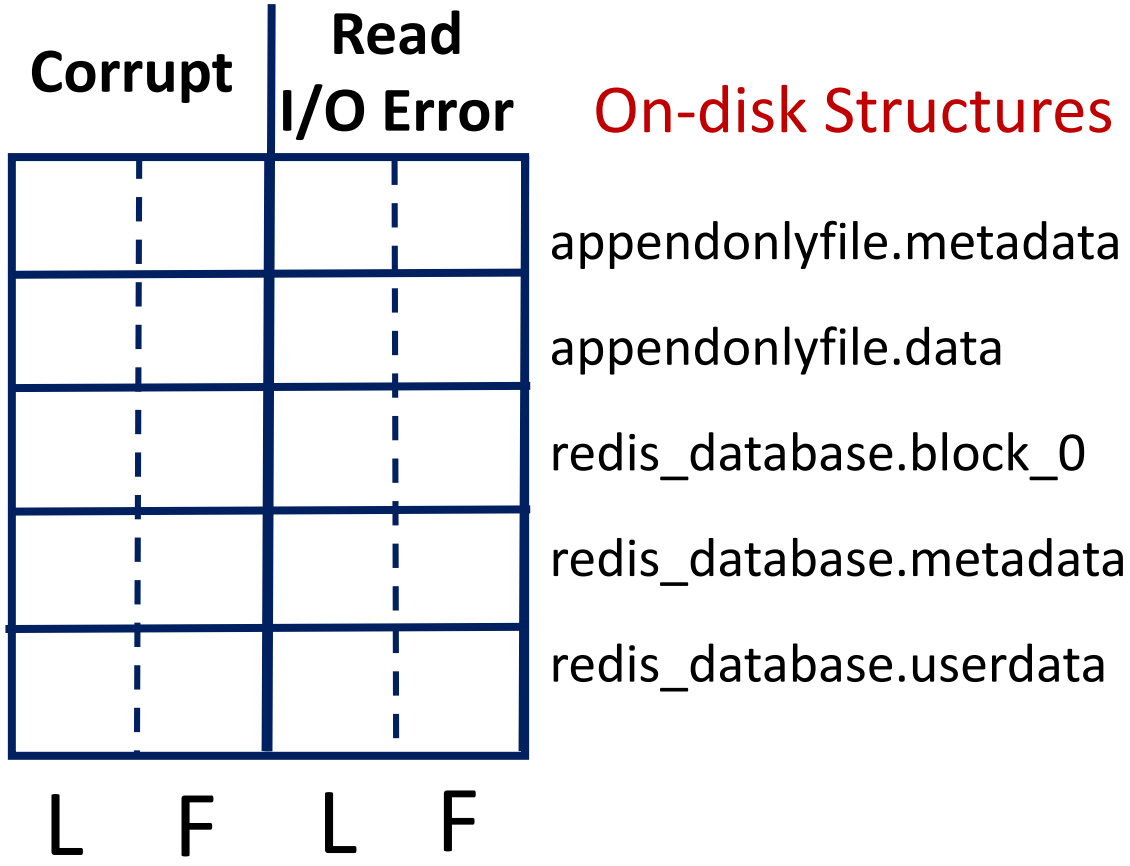
Redis: Behavior Analysis

L Leader

F Follower

Read Workload

Local Behavior



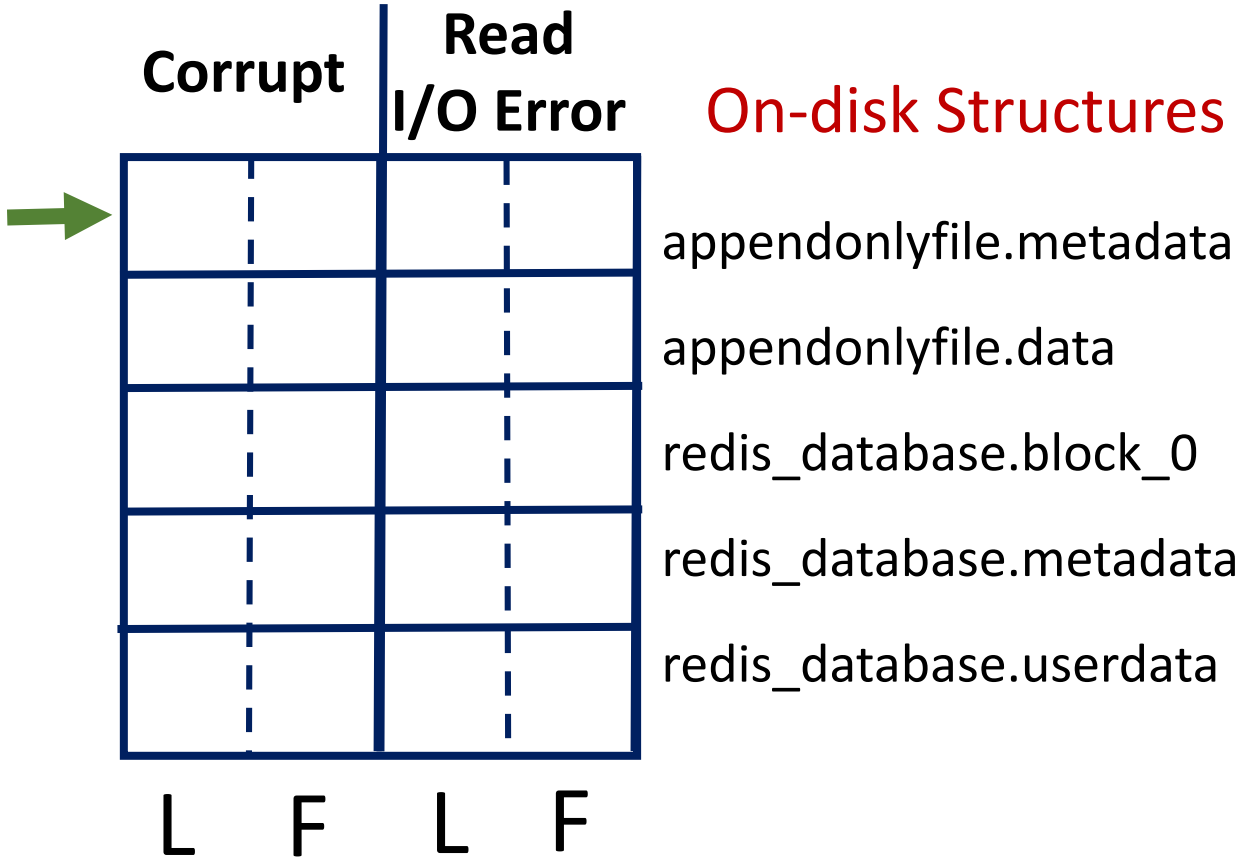
Redis: Behavior Analysis

L Leader

F Follower

Read Workload

Local Behavior



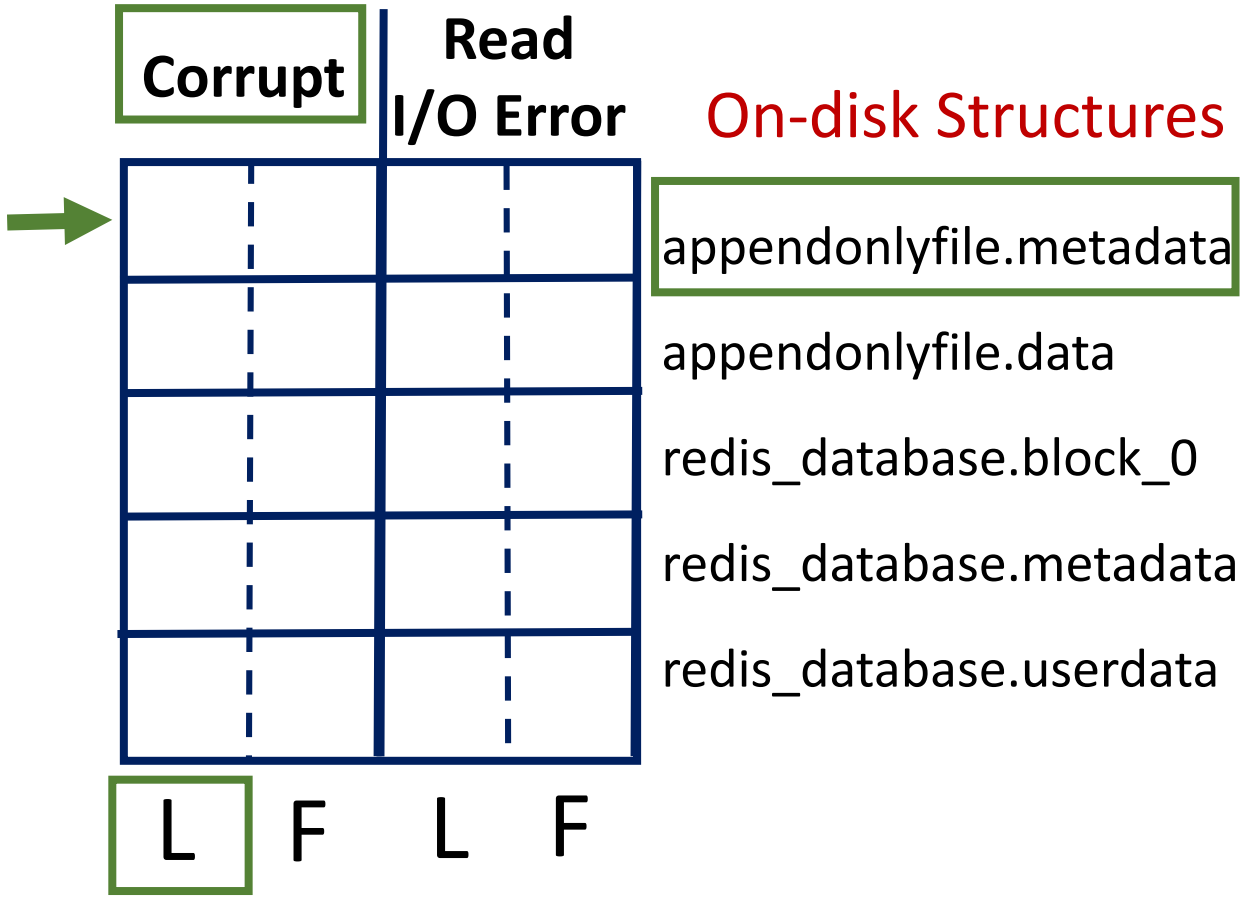
Redis: Behavior Analysis

L Leader

F Follower

Read Workload

Local Behavior



Redis: Behavior Analysis

L Leader

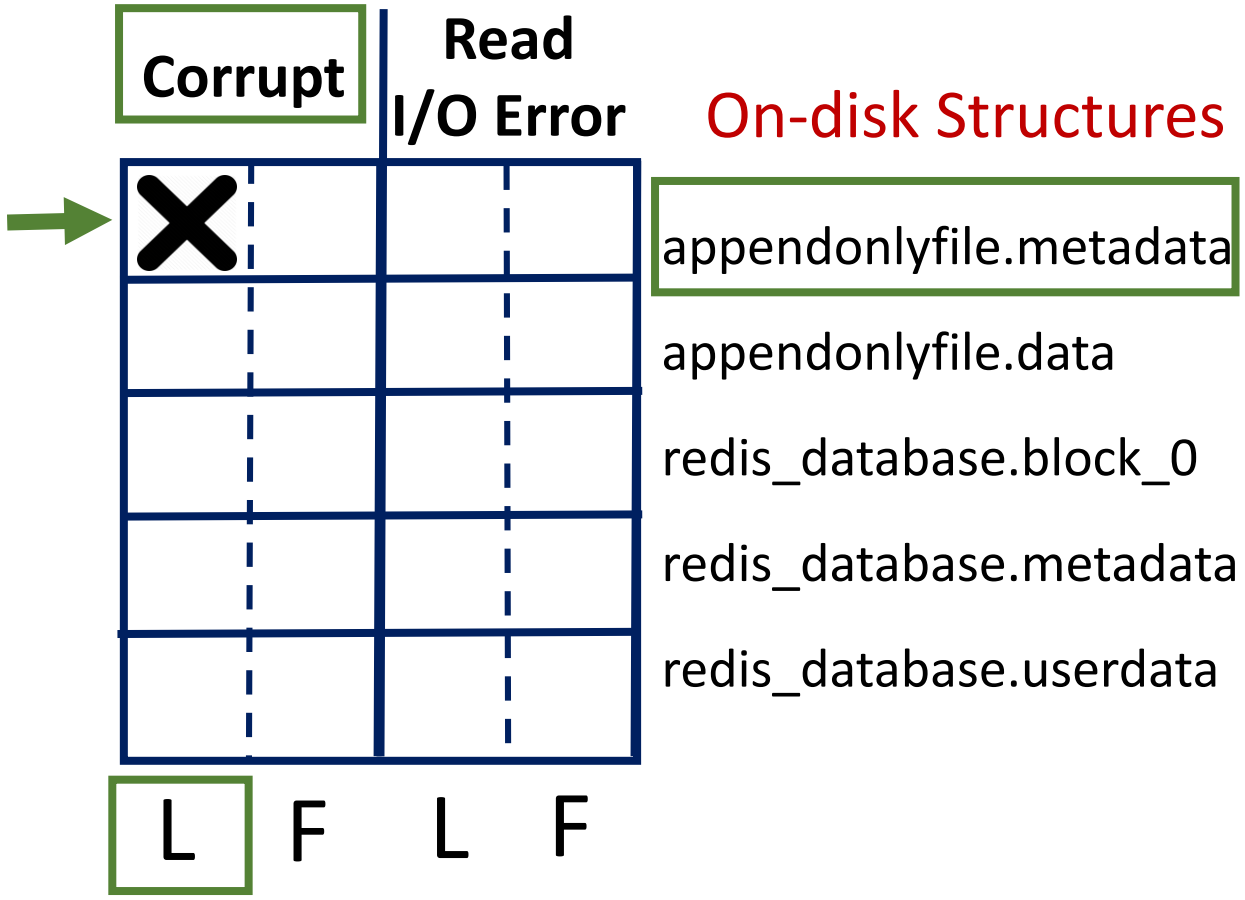
F Follower

Local Behavior

Crash

Read Workload

Local Behavior



Redis: Behavior Analysis

L Leader

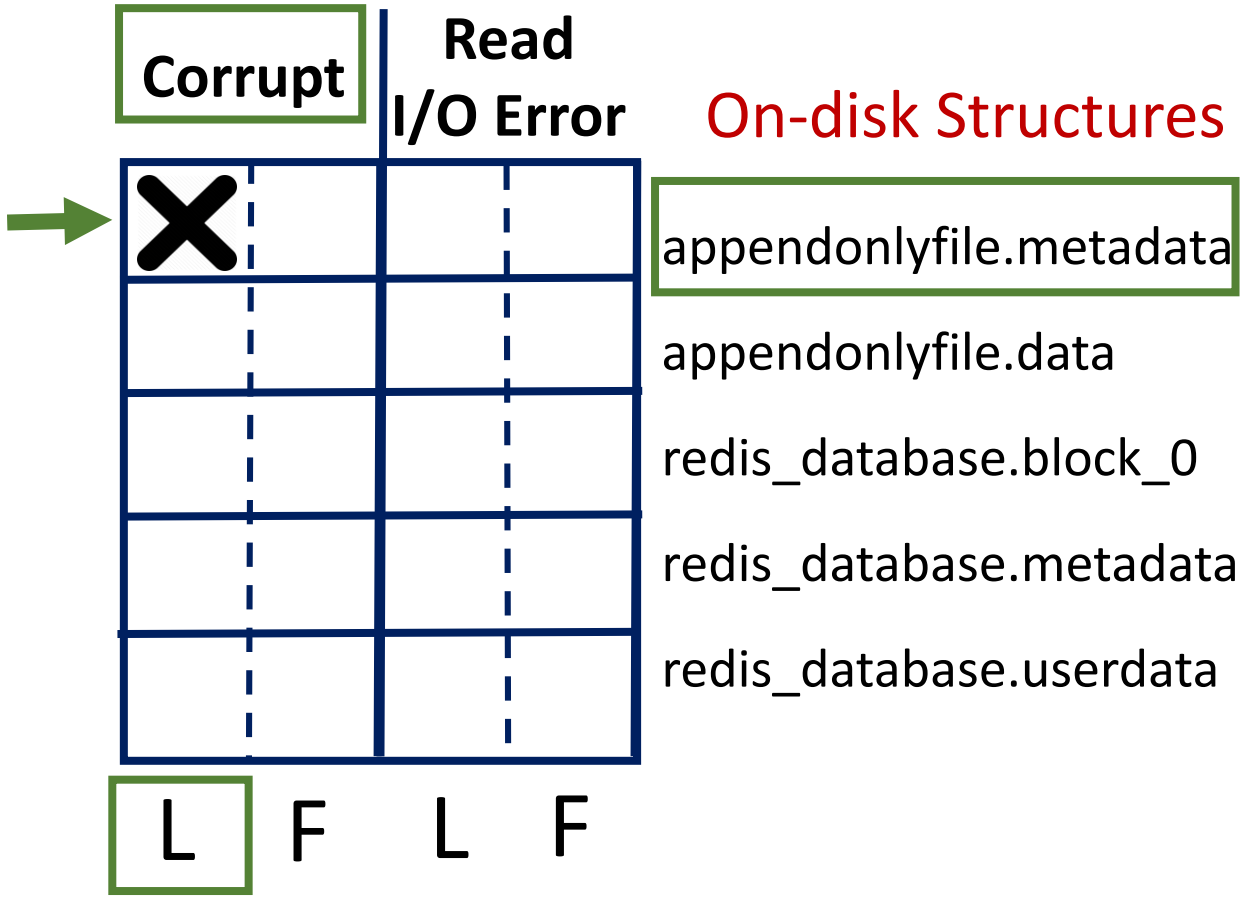
F Follower

Local Behavior

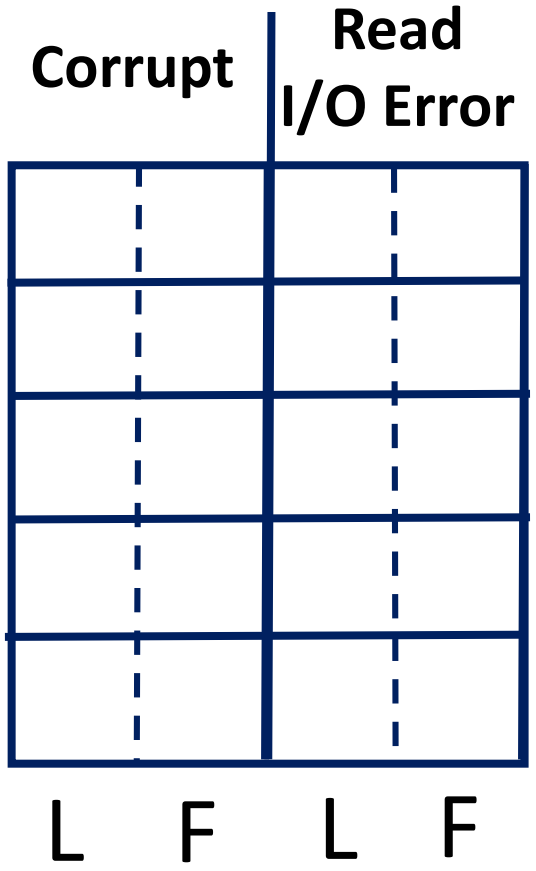
Crash

Read Workload

Local Behavior



Global Effect



Redis: Behavior Analysis

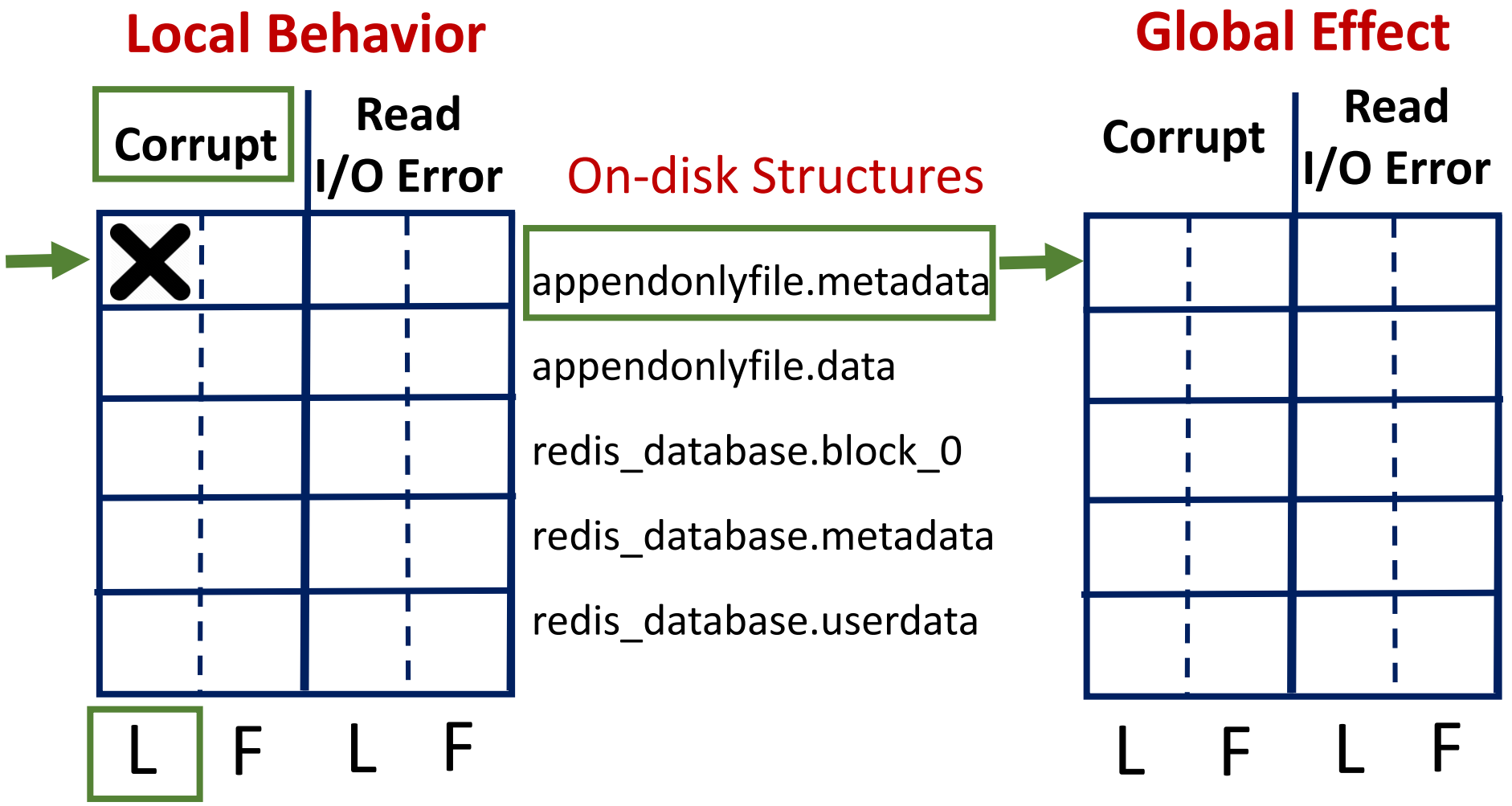
L Leader

F Follower

Local Behavior

Crash

Read Workload



Redis: Behavior Analysis

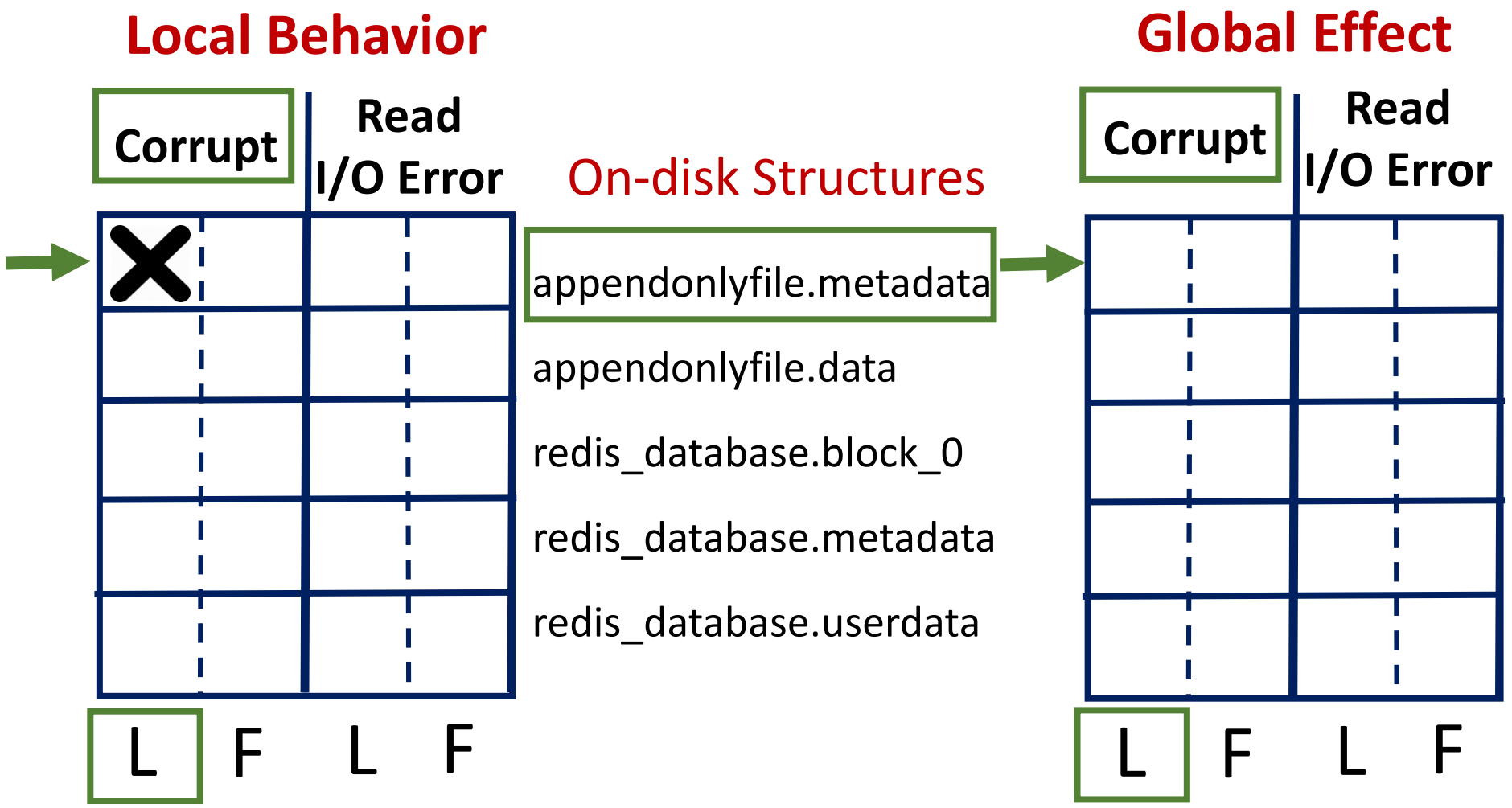
L Leader

F Follower

Local Behavior

Crash

Read Workload



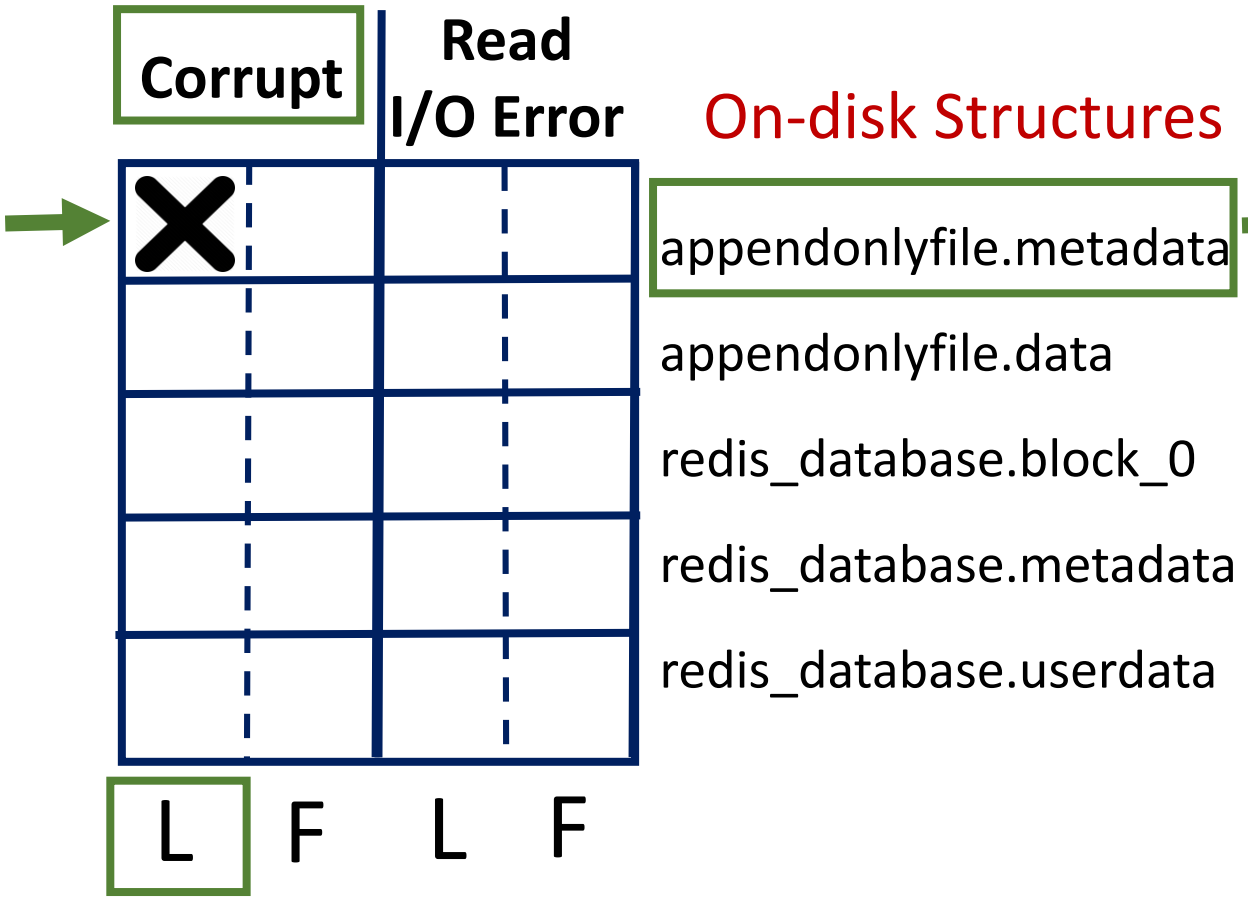
Redis: Behavior Analysis

Read Workload

L Leader

F Follower

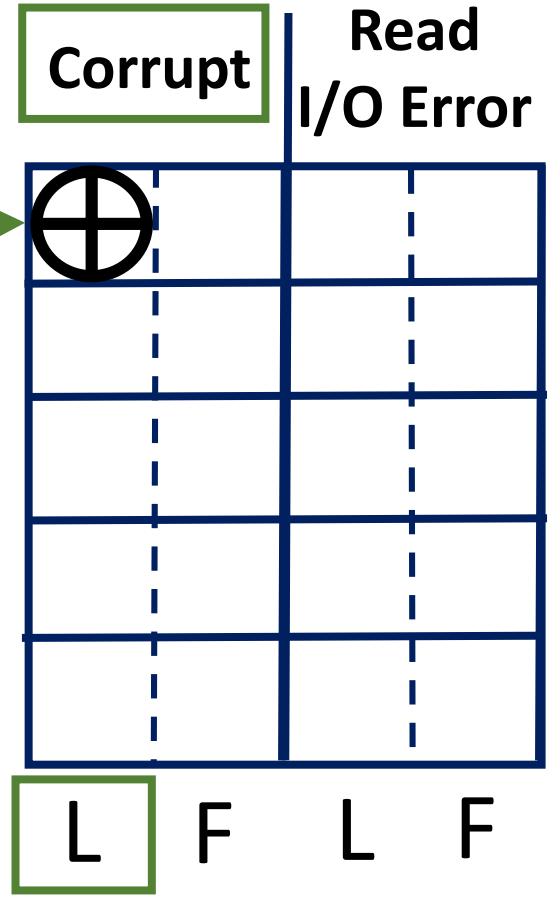
Local Behavior



On-disk Structures

- appendonlyfile.metadata
- appendonlyfile.data
- redis_database.block_0
- redis_database.metadata
- redis_database.userdata

Global Effect



Local Behavior

Crash

Global Effect

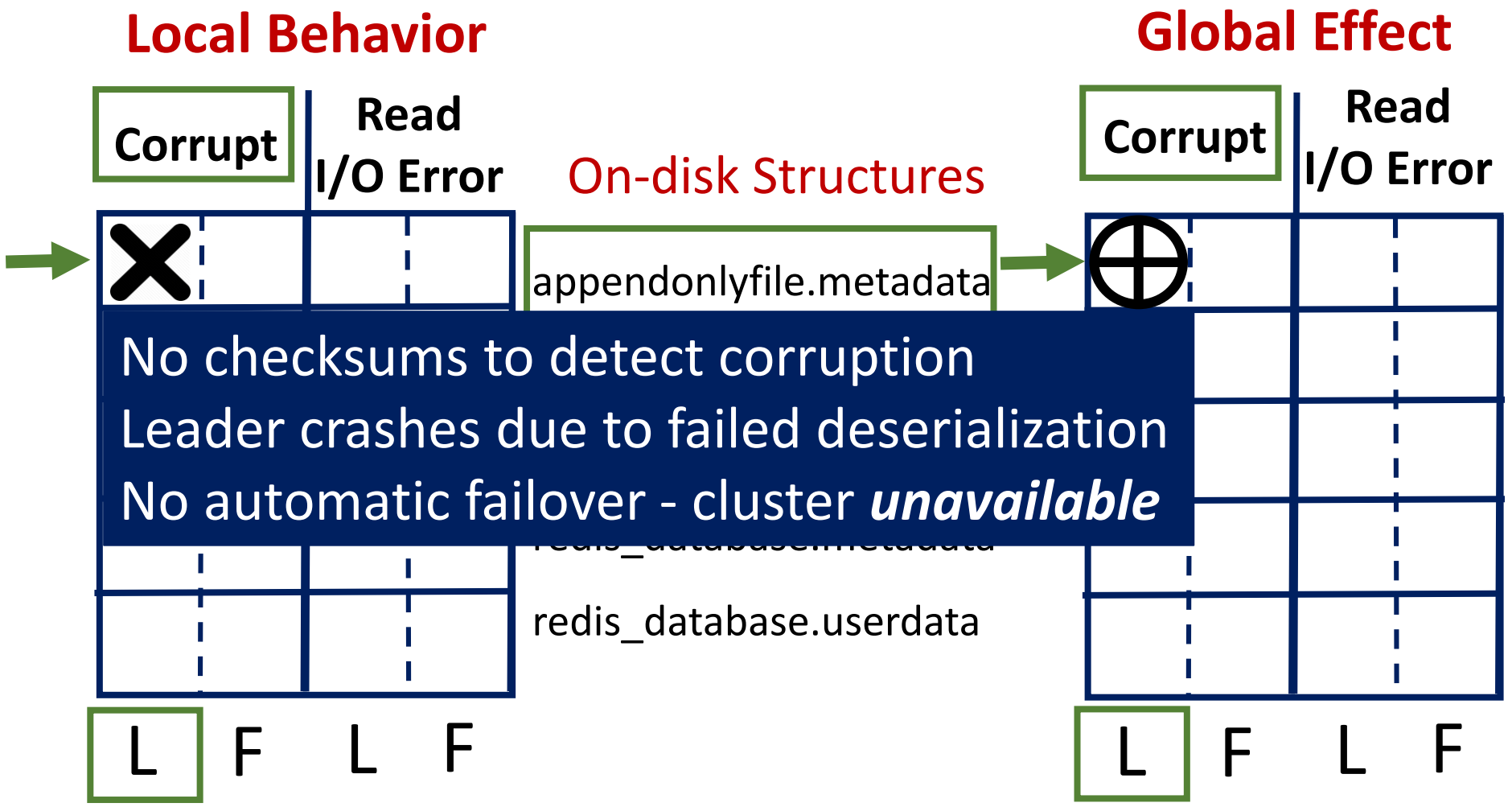
Unavailability

Redis: Behavior Analysis

Read Workload

L Leader

F Follower



Local Behavior

Crash

Global Effect

Unavailability

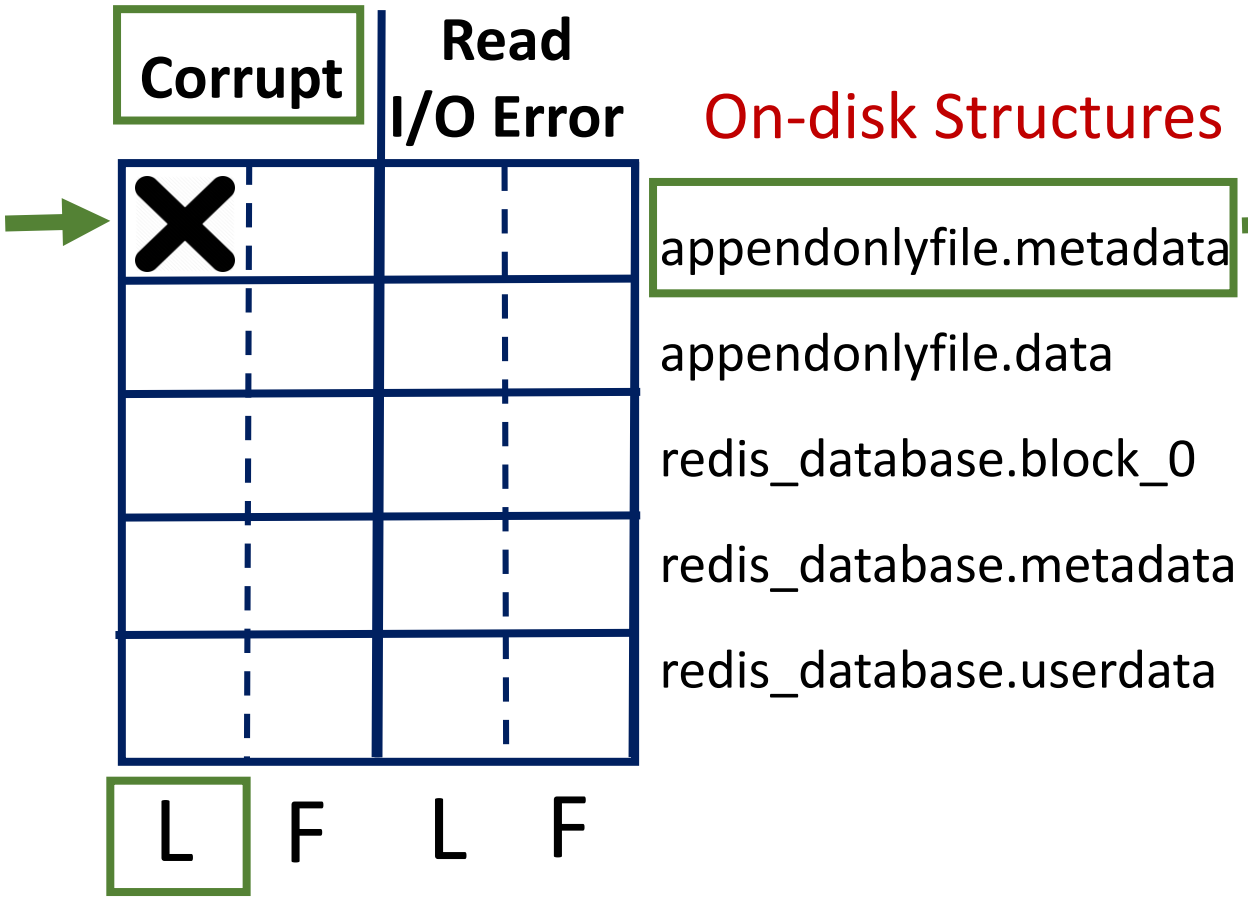
Redis: Behavior Analysis

Read Workload

L Leader

F Follower

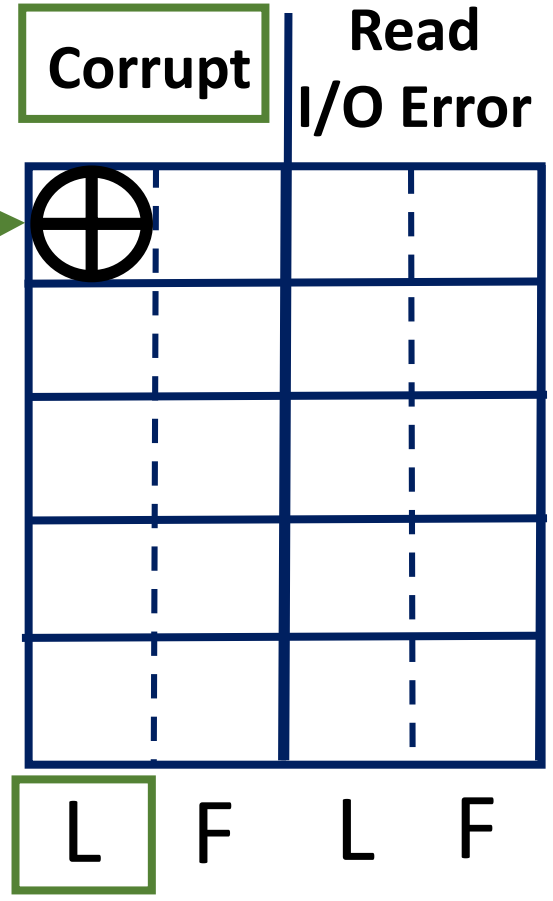
Local Behavior



On-disk Structures

- appendonlyfile.metadata
- appendonlyfile.data
- redis_database.block_0
- redis_database.metadata
- redis_database.userdata

Global Effect



Local Behavior

Crash

Global Effect

Unavailability

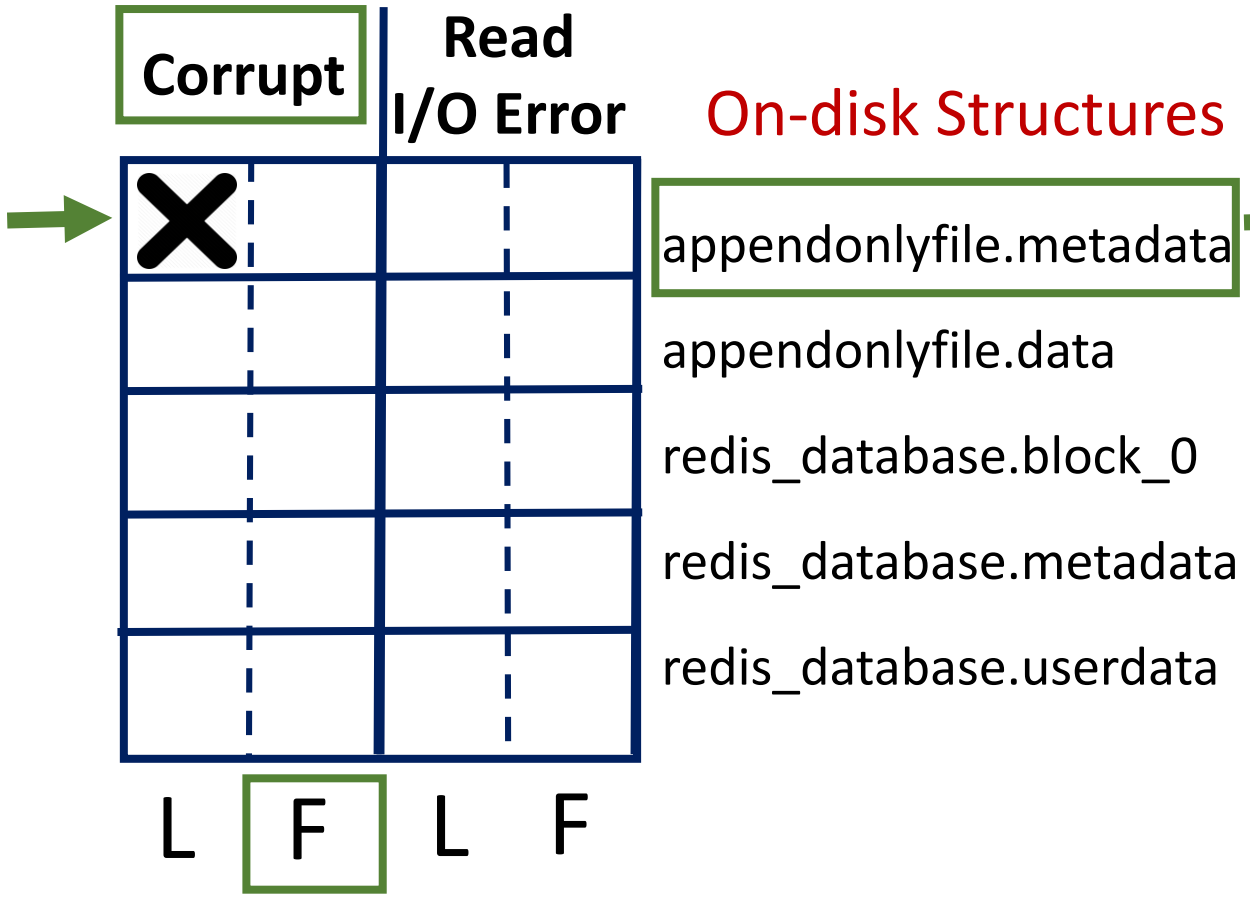
Redis: Behavior Analysis

L Leader

F Follower

Read Workload

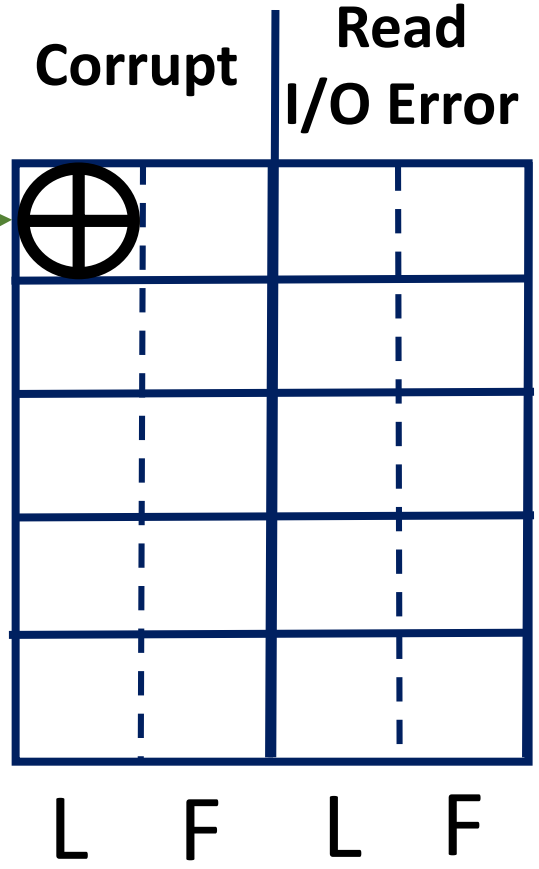
Local Behavior



On-disk Structures

- appendonlyfile.metadata
- appendonlyfile.data
- redis_database.block_0
- redis_database.metadata
- redis_database.userdata

Global Effect



Local Behavior

Crash

Global Effect

Unavailability

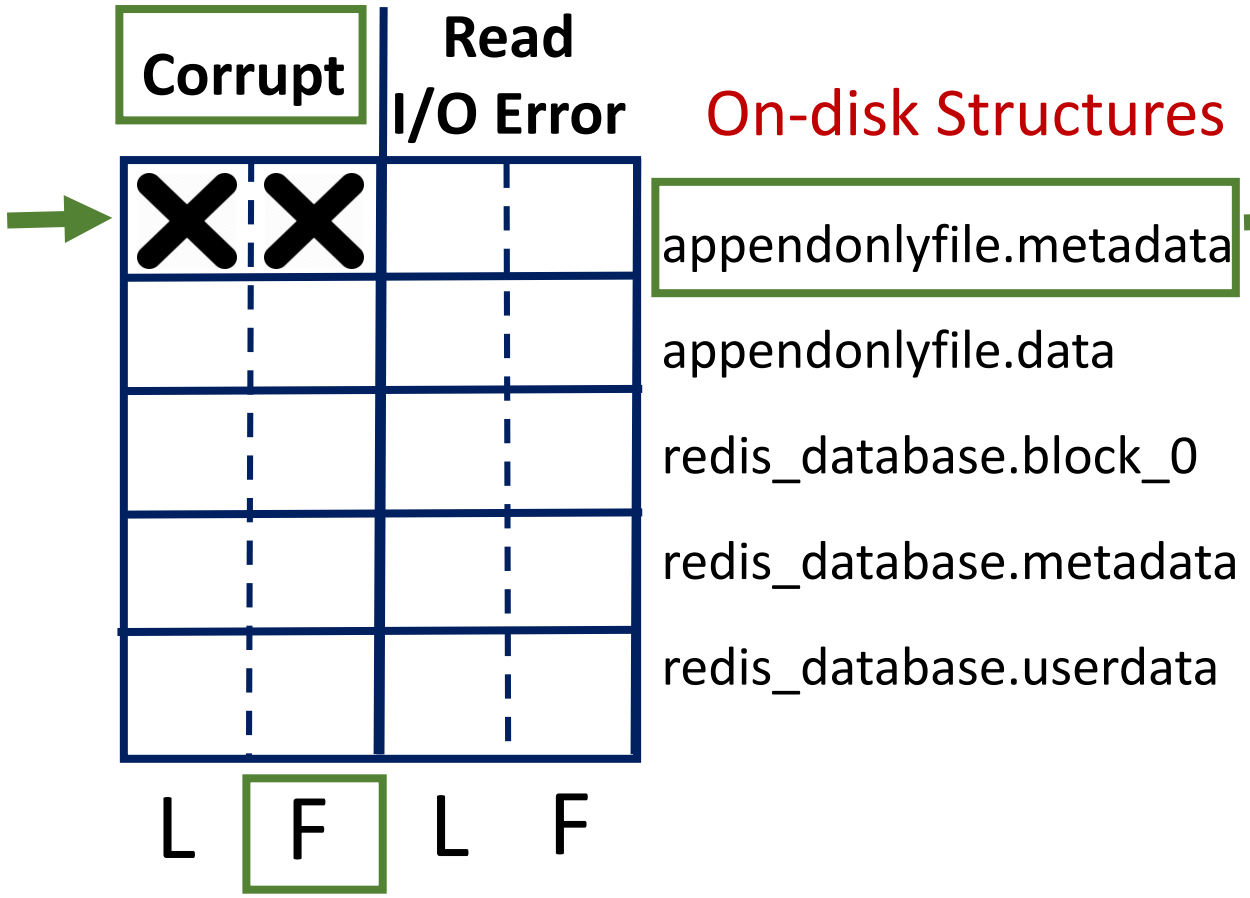
Redis: Behavior Analysis

L Leader

F Follower

Read Workload

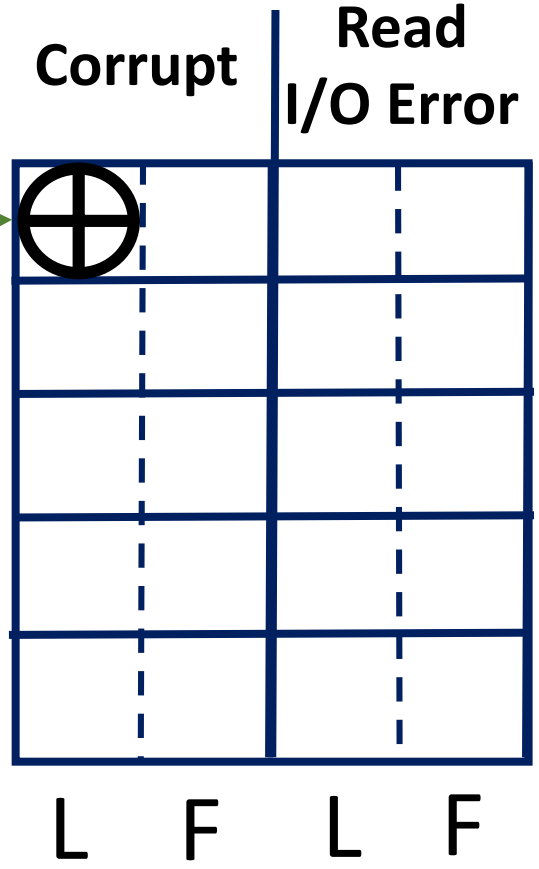
Local Behavior



On-disk Structures

- appendonlyfile.metadata
- appendonlyfile.data
- redis_database.block_0
- redis_database.metadata
- redis_database.userdata

Global Effect



Local Behavior

Crash

Global Effect

Unavailability

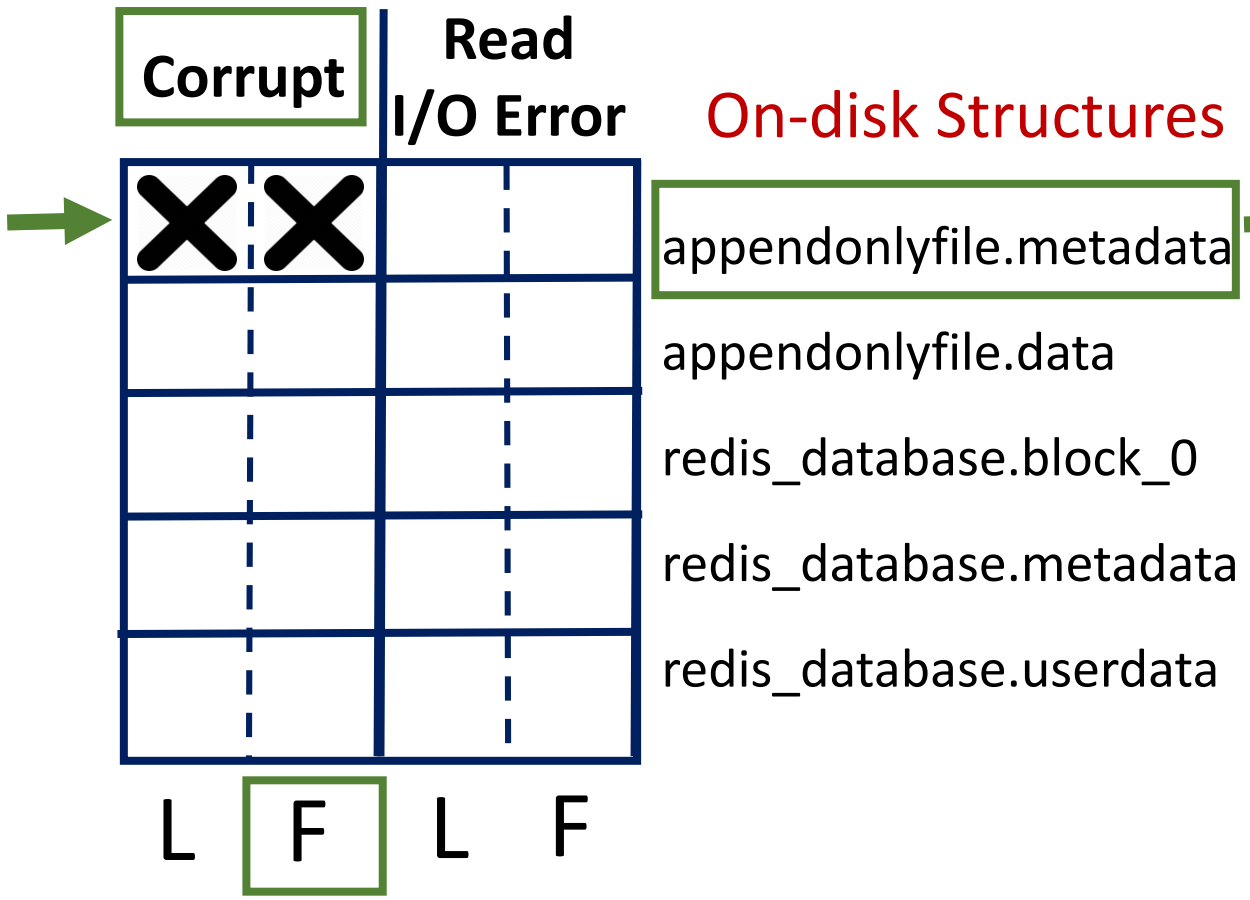
Redis: Behavior Analysis

L Leader

F Follower

Read Workload

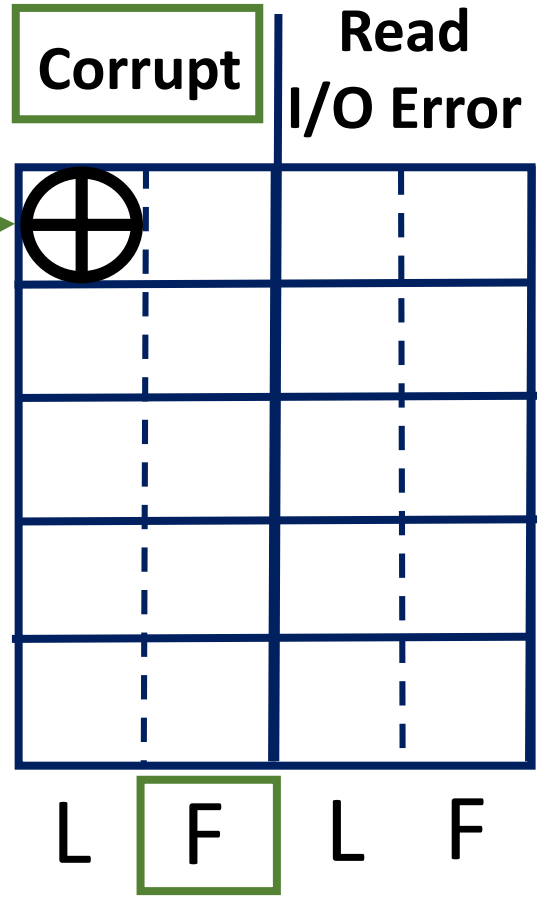
Local Behavior



On-disk Structures

- appendonlyfile.metadata
- appendonlyfile.data
- redis_database.block_0
- redis_database.metadata
- redis_database.userdata

Global Effect



Local Behavior

Crash

Global Effect

Unavailability

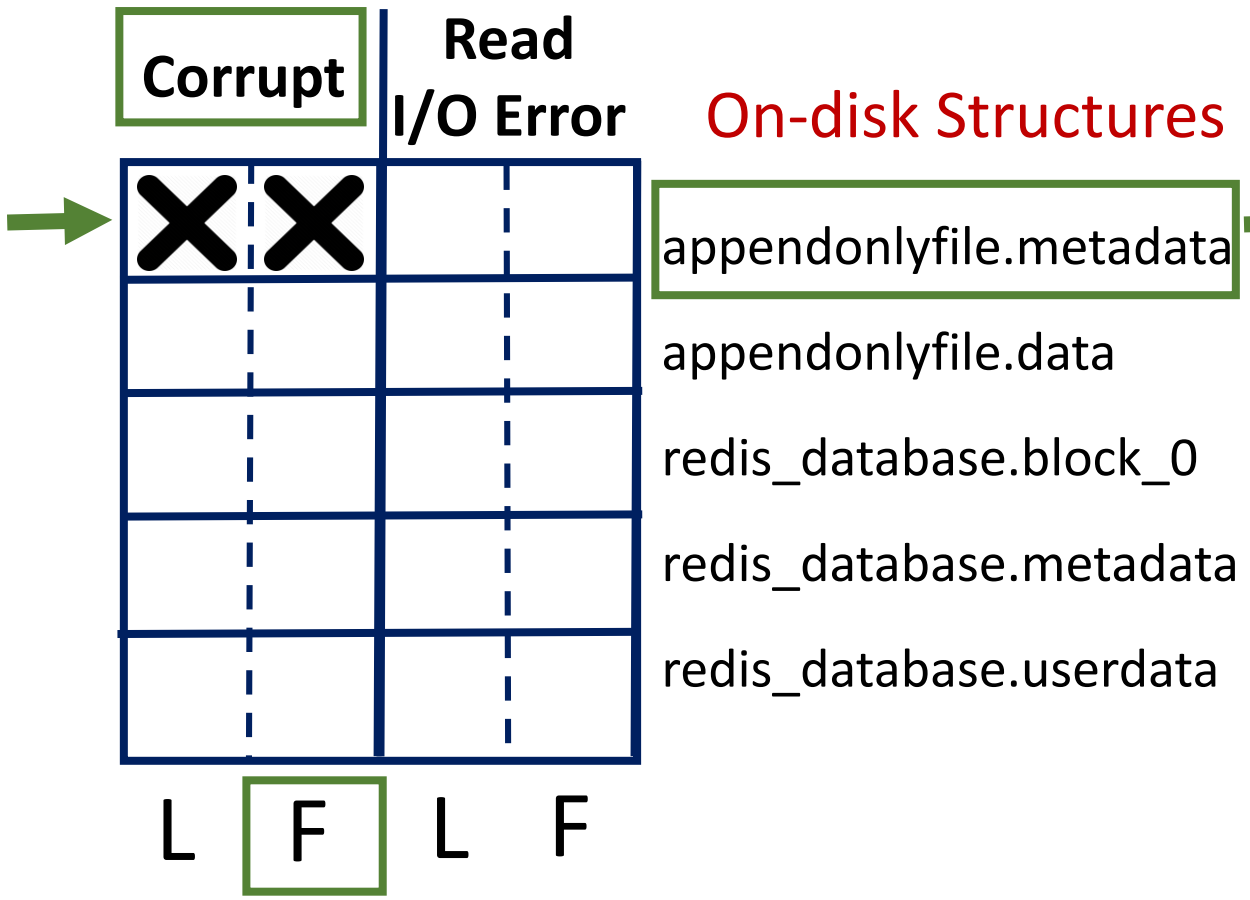
Redis: Behavior Analysis

L Leader

F Follower

Read Workload

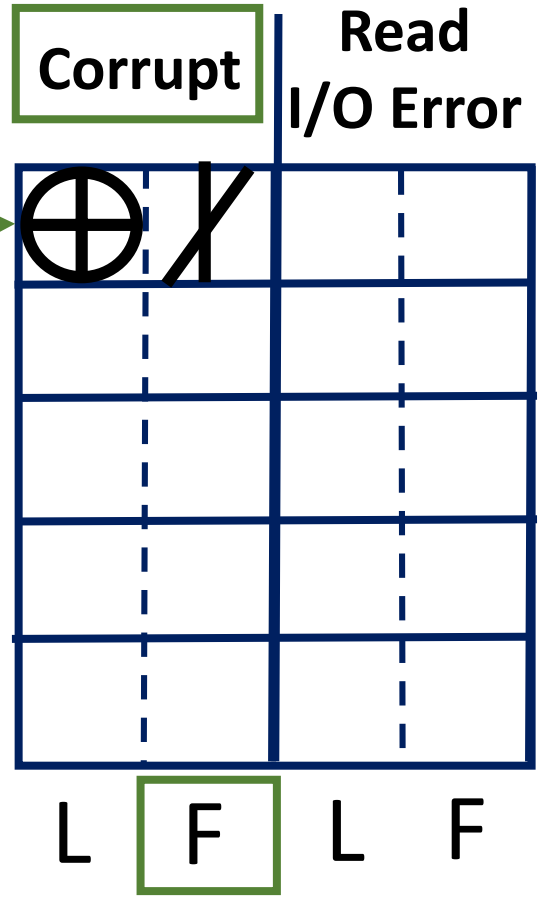
Local Behavior



On-disk Structures

- appendonlyfile.metadata
- appendonlyfile.data
- redis_database.block_0
- redis_database.metadata
- redis_database.userdata

Global Effect



Local Behavior

Crash

Global Effect

Unavailability

Reduced Redundancy

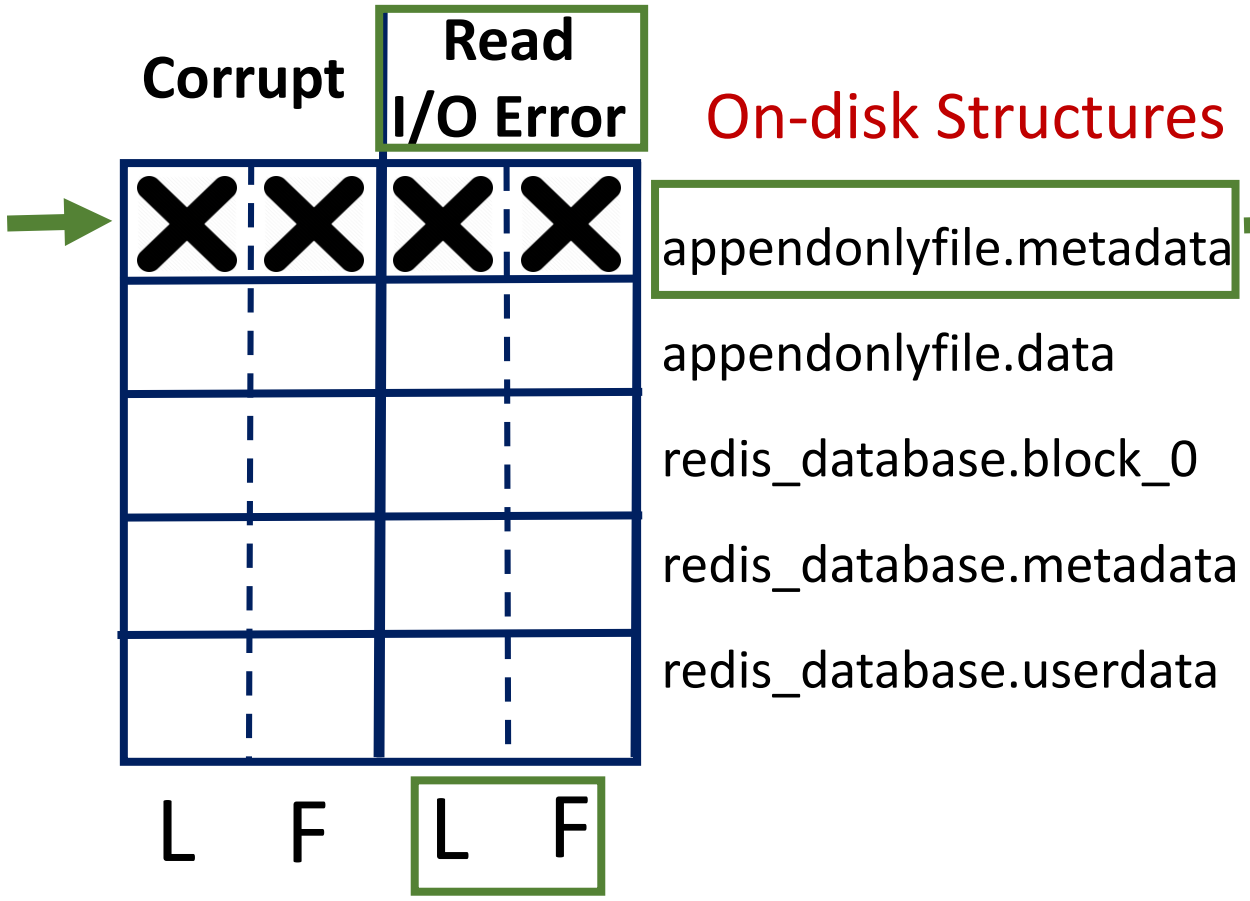
Redis: Behavior Analysis

Read Workload

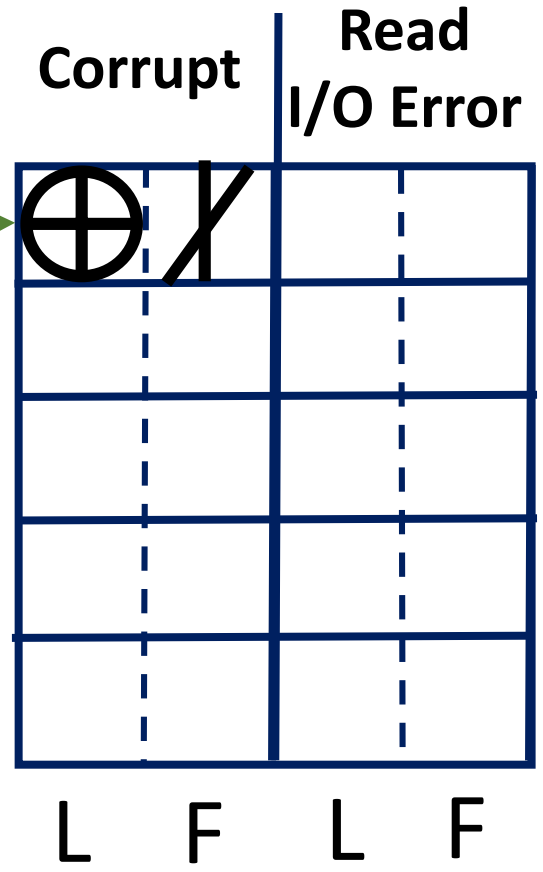
L Leader

F Follower

Local Behavior



Global Effect



Local Behavior

Crash

Global Effect

Unavailability

Reduced Redundancy

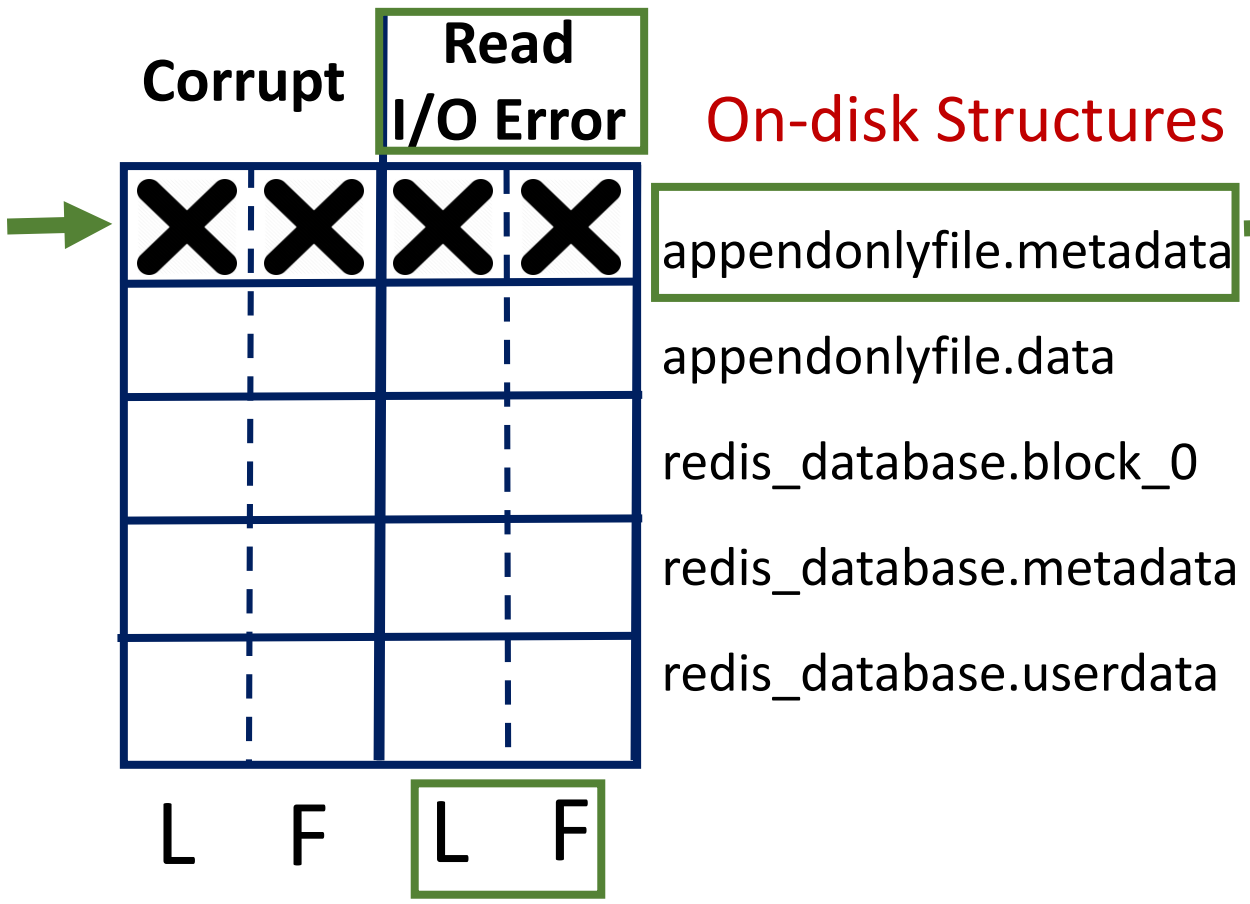
Redis: Behavior Analysis

Read Workload

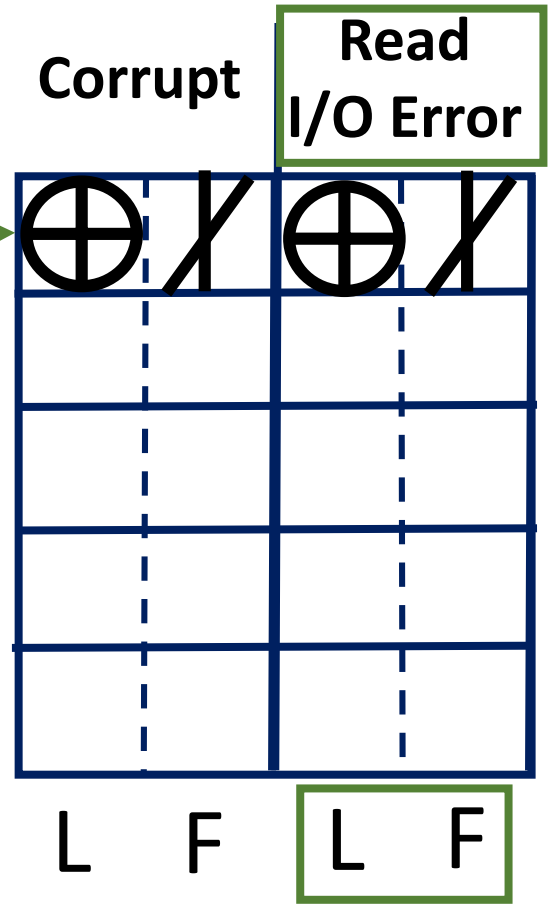
L Leader

F Follower

Local Behavior



Global Effect



Local Behavior

Crash

Global Effect

Unavailability

Reduced Redundancy

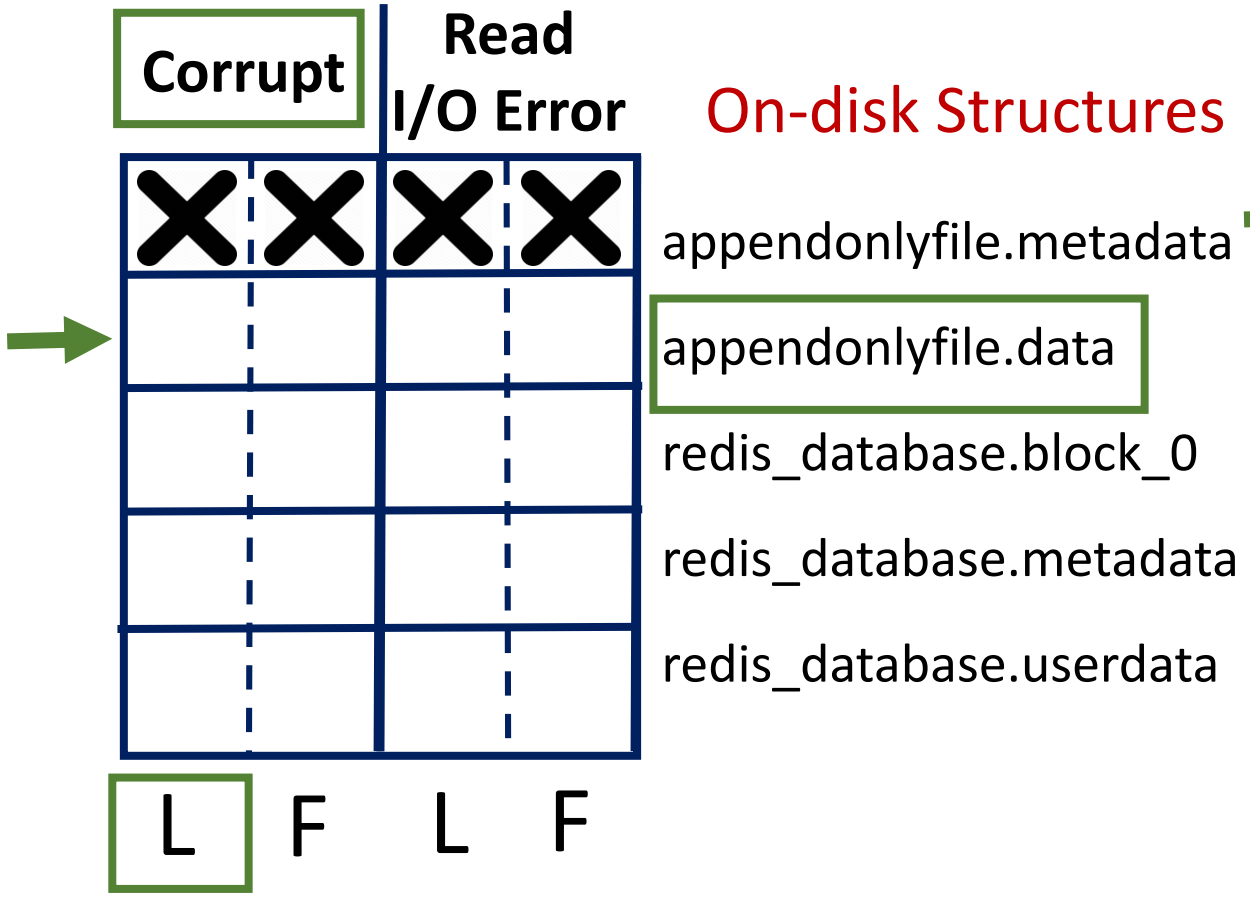
Redis: Behavior Analysis

L Leader

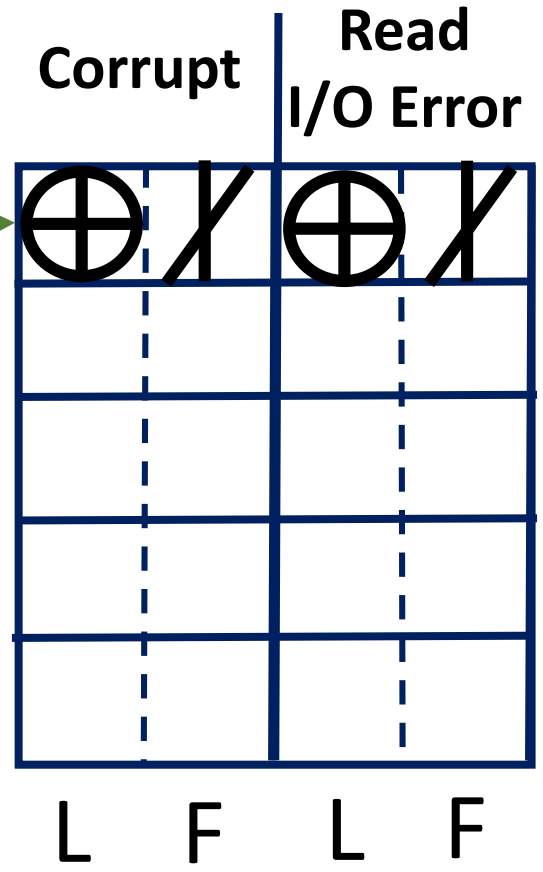
F Follower

Read Workload

Local Behavior



Global Effect



Local Behavior

Crash

Global Effect

Unavailability

Reduced Redundancy

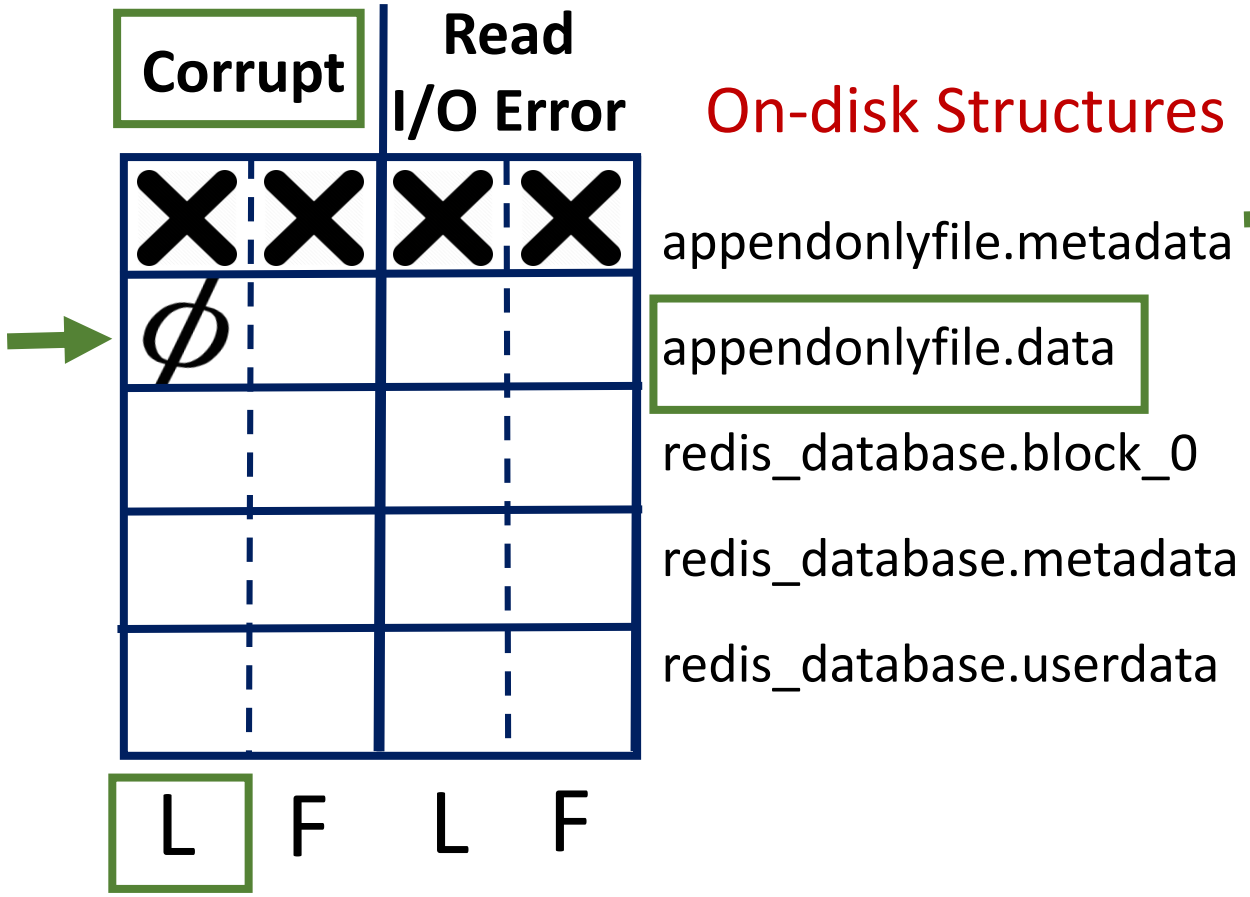
Redis: Behavior Analysis

Read Workload

L Leader

F Follower

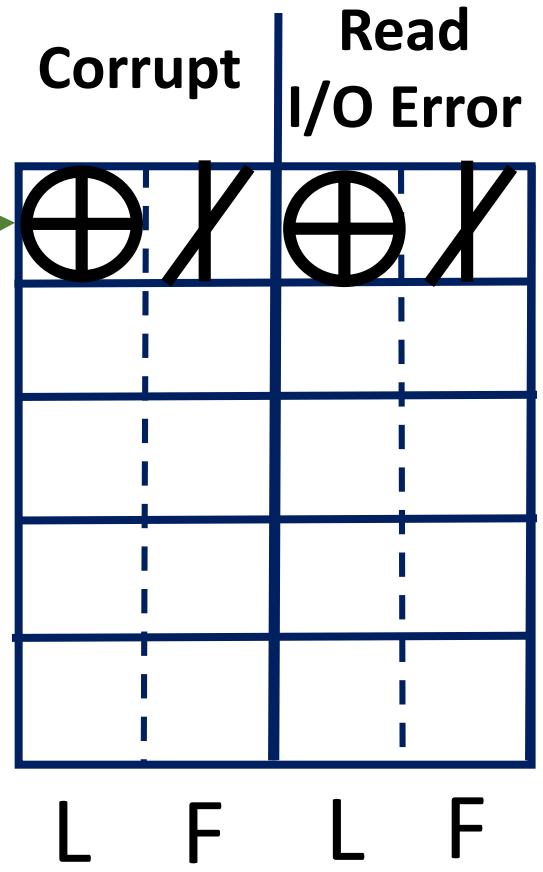
Local Behavior



On-disk Structures

- appendonlyfile.metadata
- appendonlyfile.data
- redis_database.block_0
- redis_database.metadata
- redis_database.userdata

Global Effect



Local Behavior

- Crash
- No Detection/ No Recovery

Global Effect

- Unavailability
- Reduced Redundancy

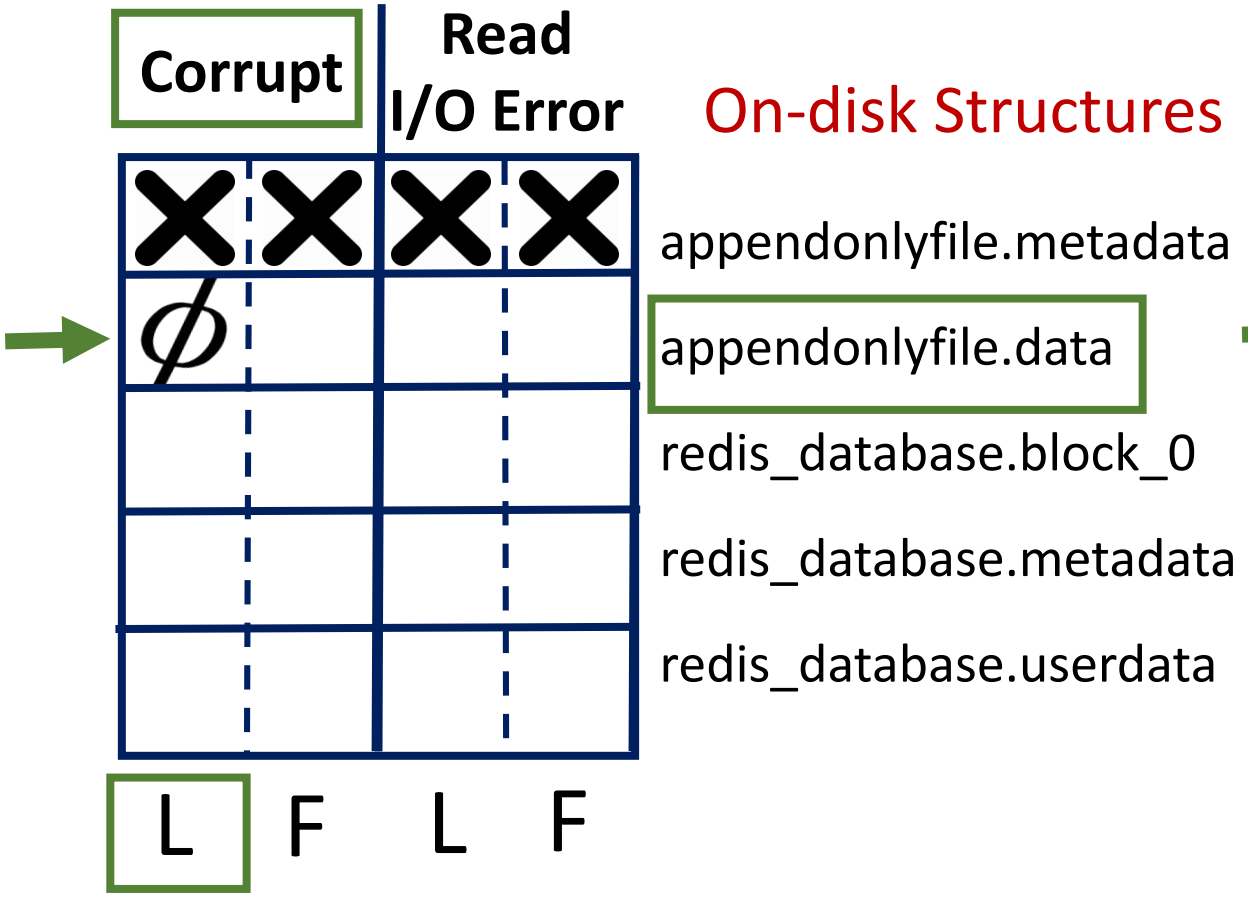
Redis: Behavior Analysis

Read Workload

L Leader

F Follower

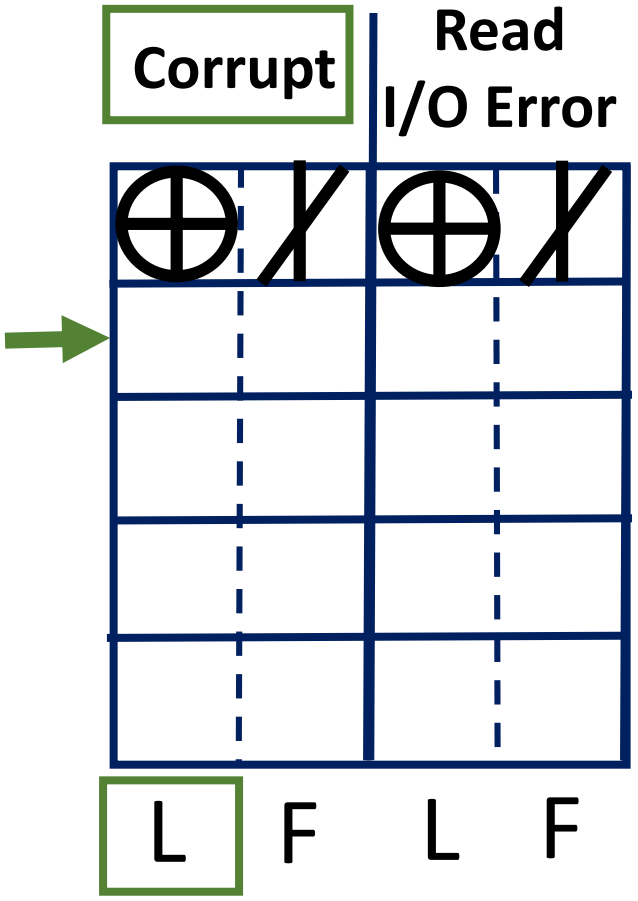
Local Behavior



On-disk Structures

- appendonlyfile.metadata
- appendonlyfile.data
- redis_database.block_0
- redis_database.metadata
- redis_database.userdata

Global Effect



Local Behavior

- Crash
- No Detection/ No Recovery

Global Effect

- Unavailability
- Reduced Redundancy

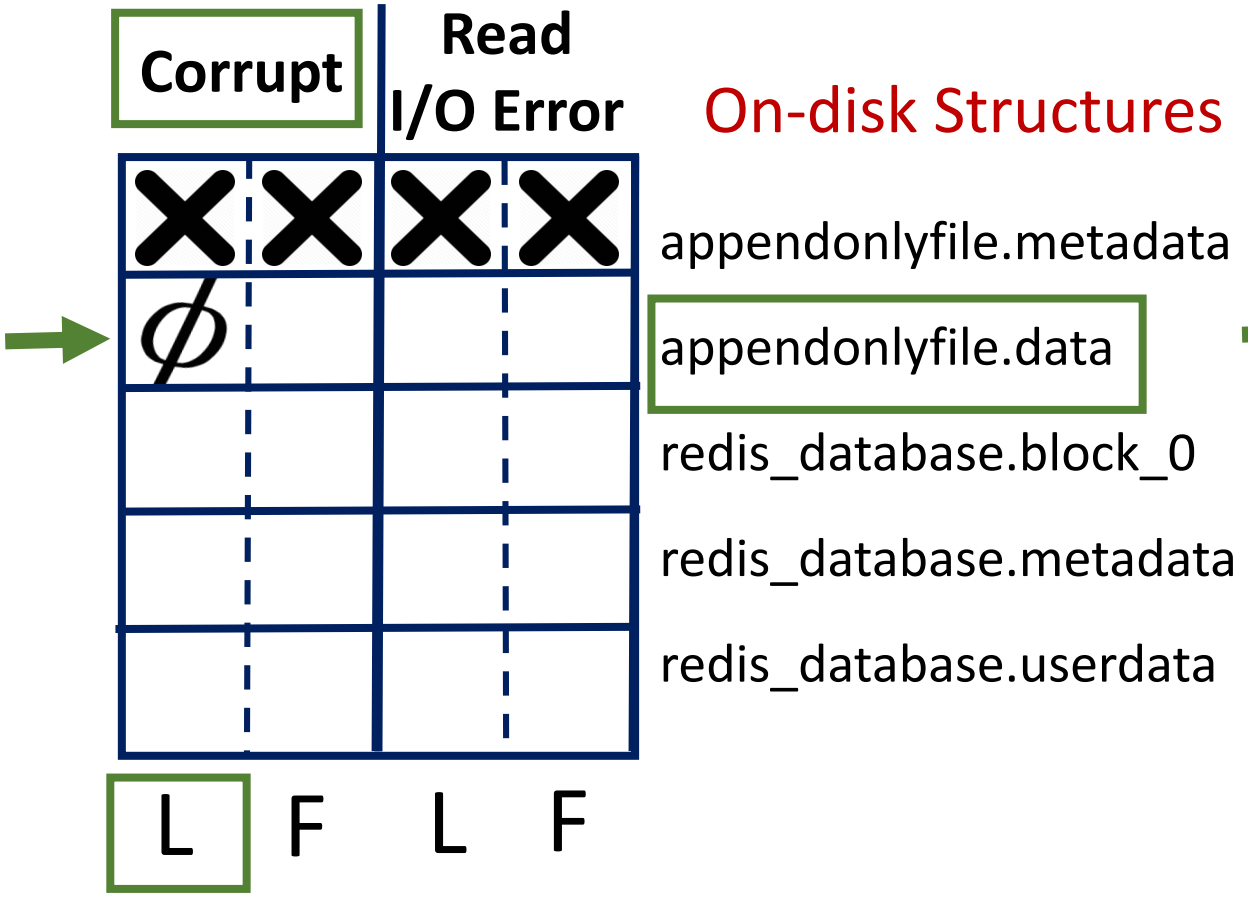
Redis: Behavior Analysis

Read Workload

L Leader

F Follower

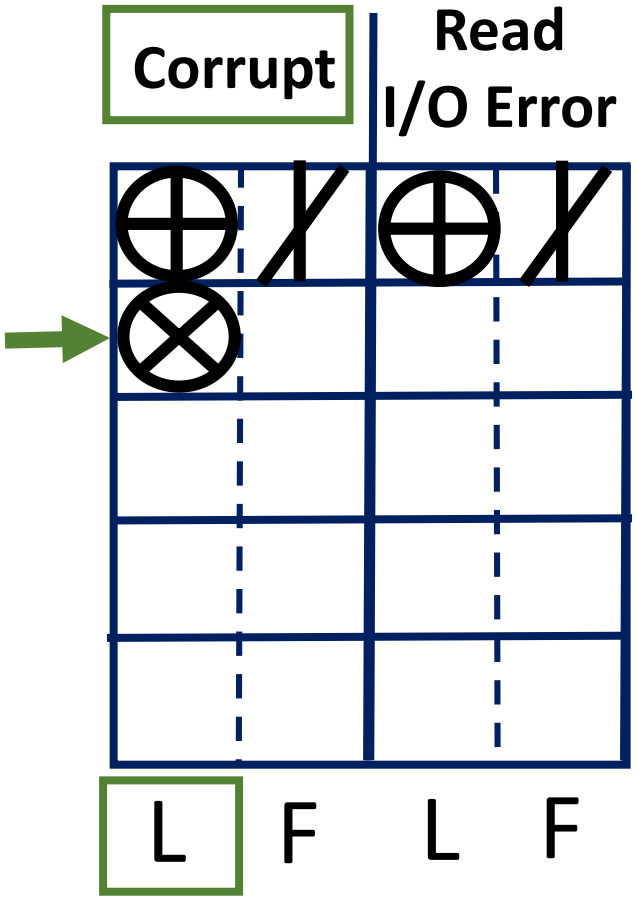
Local Behavior



On-disk Structures

- appendonlyfile.metadata
- appendonlyfile.data
- redis_database.block_0
- redis_database.metadata
- redis_database.userdata

Global Effect



Local Behavior

- Crash
- No Detection/ No Recovery

Global Effect

- Unavailability
- Reduced Redundancy
- Corruption

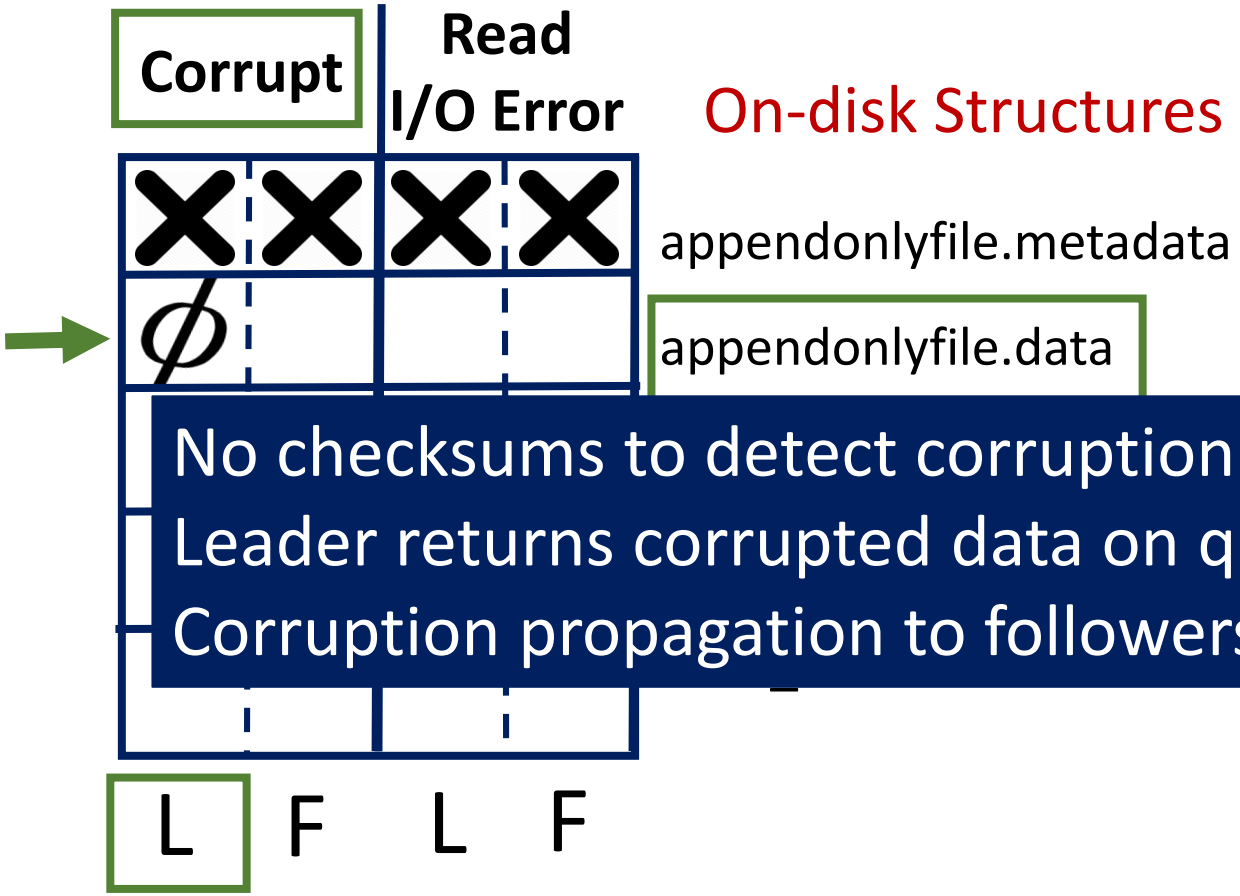
Redis: Behavior Analysis

L Leader

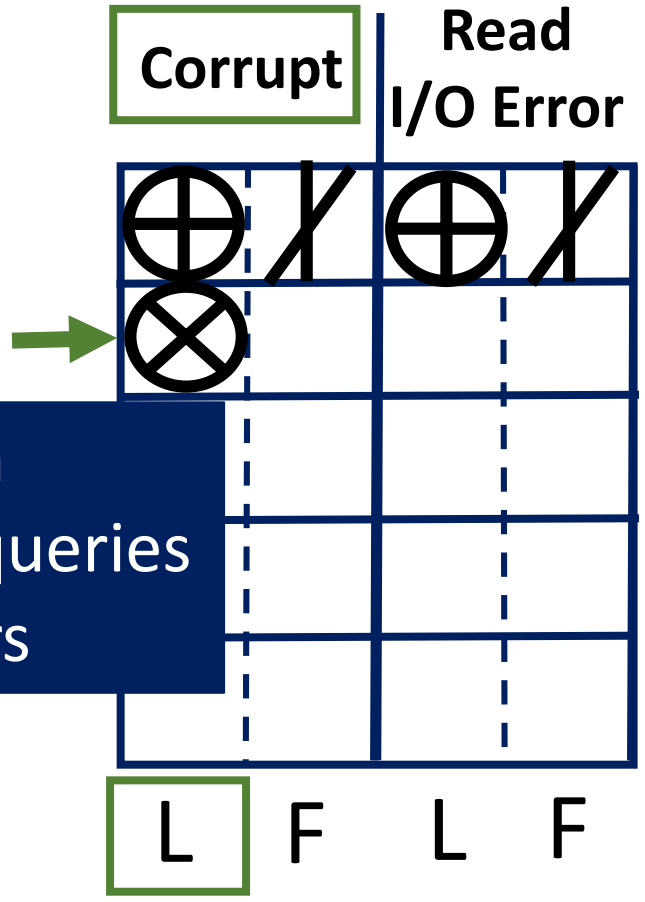
F Follower

Read Workload

Local Behavior



Global Effect



Local Behavior

Crash

No Detection/
No Recovery

Global Effect

Unavailability

Reduced
Redundancy

Corruption

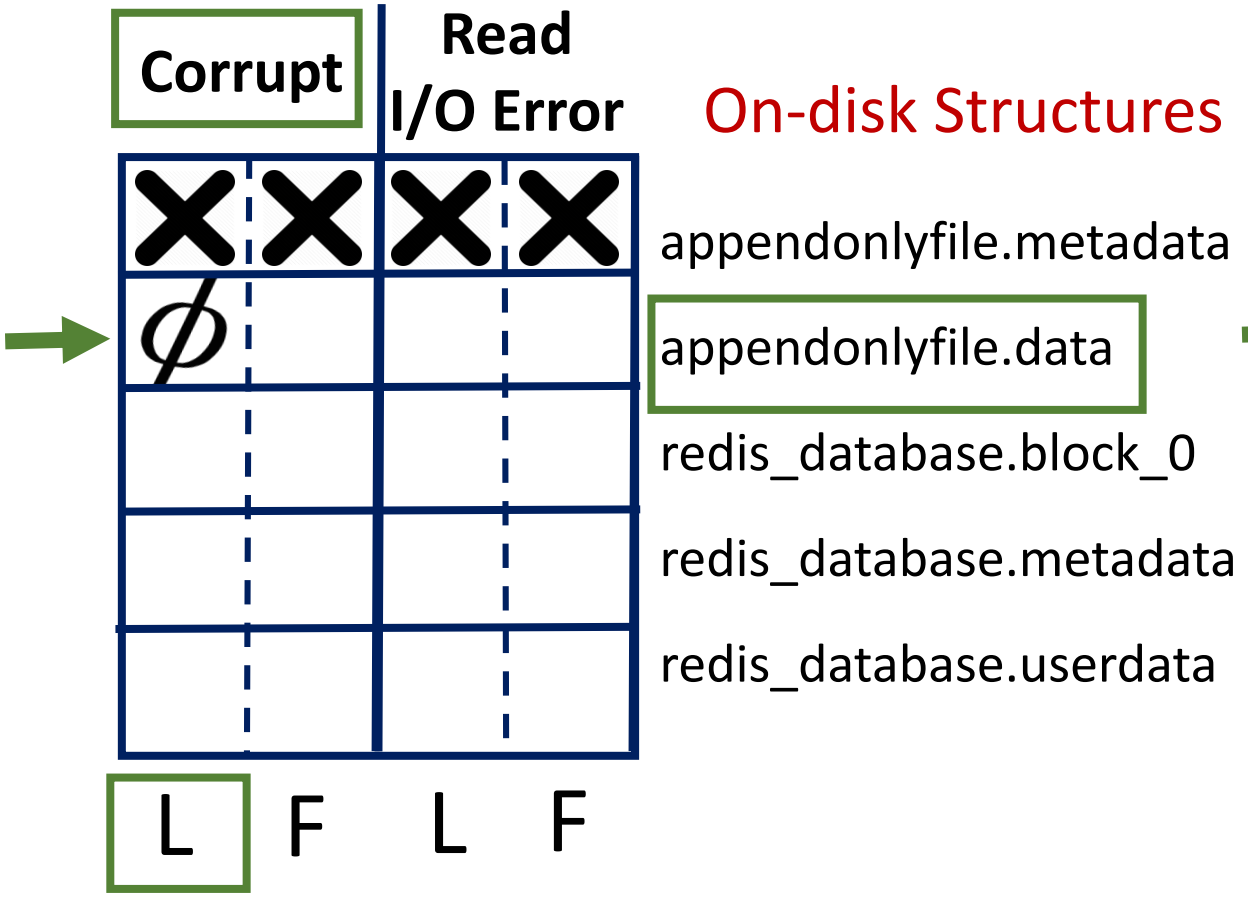
Redis: Behavior Analysis

Read Workload

L Leader

F Follower

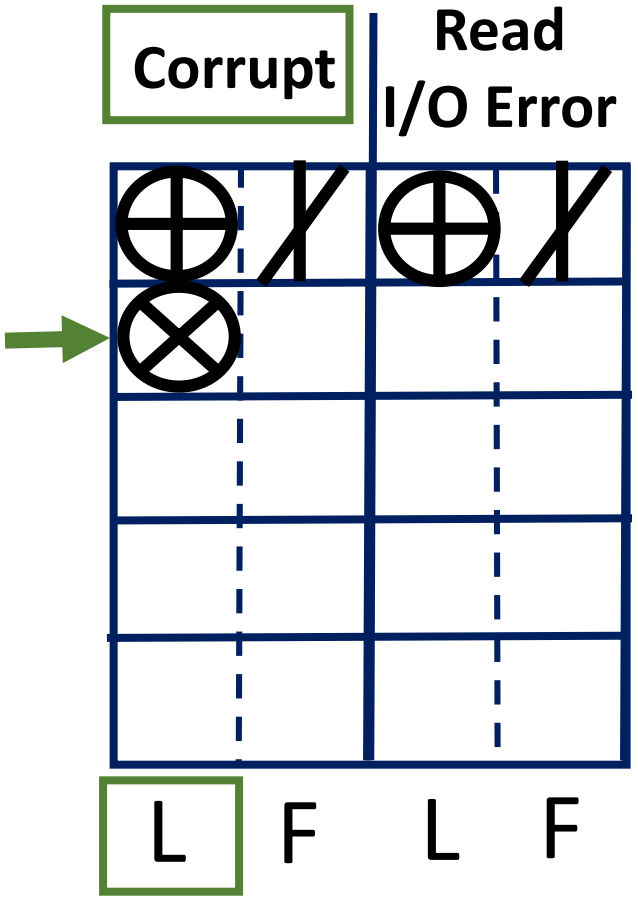
Local Behavior



On-disk Structures

- appendonlyfile.metadata
- appendonlyfile.data
- redis_database.block_0
- redis_database.metadata
- redis_database.userdata

Global Effect



Local Behavior

- Crash
- No Detection/ No Recovery

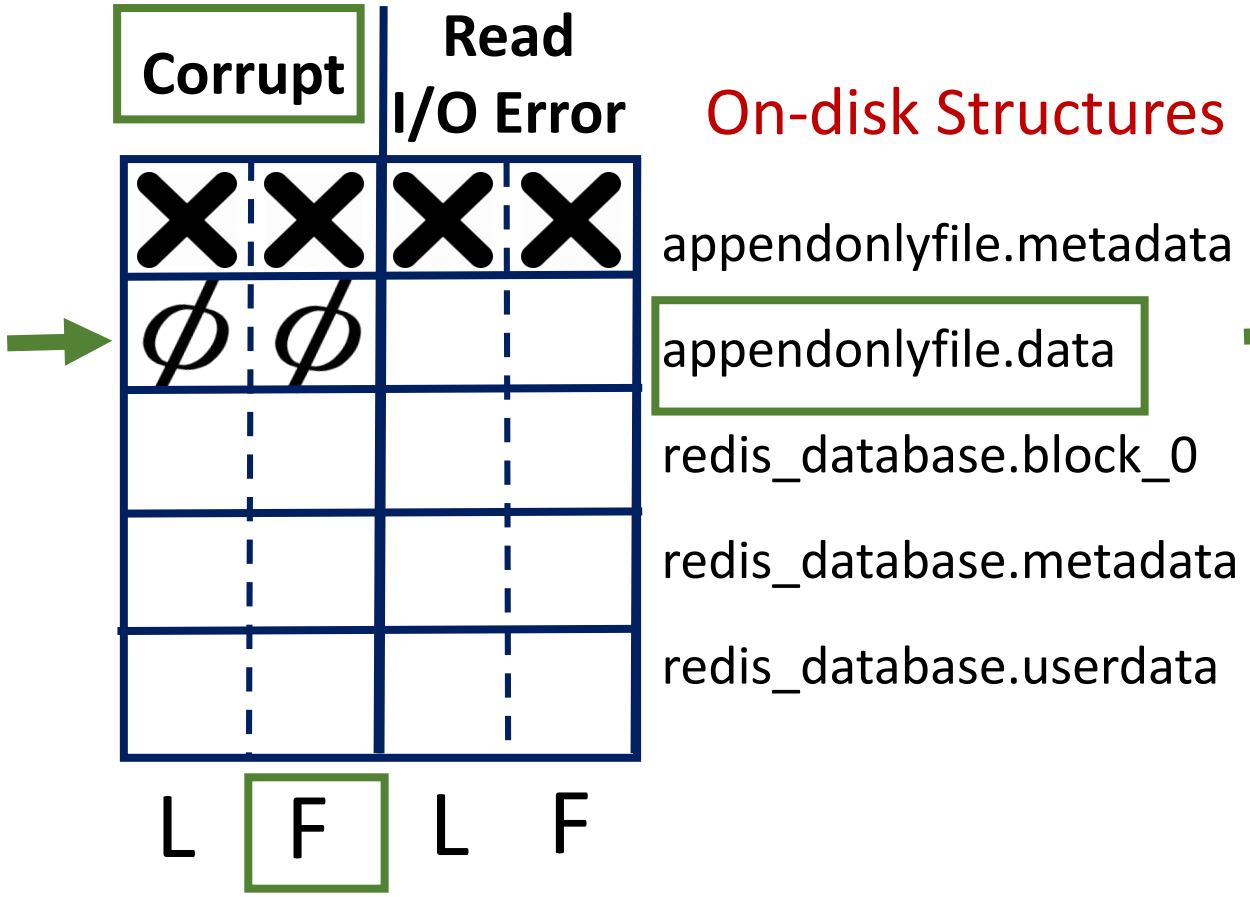
Global Effect

- Unavailability
- Reduced Redundancy
- Corruption

Redis: Behavior Analysis

Read Workload

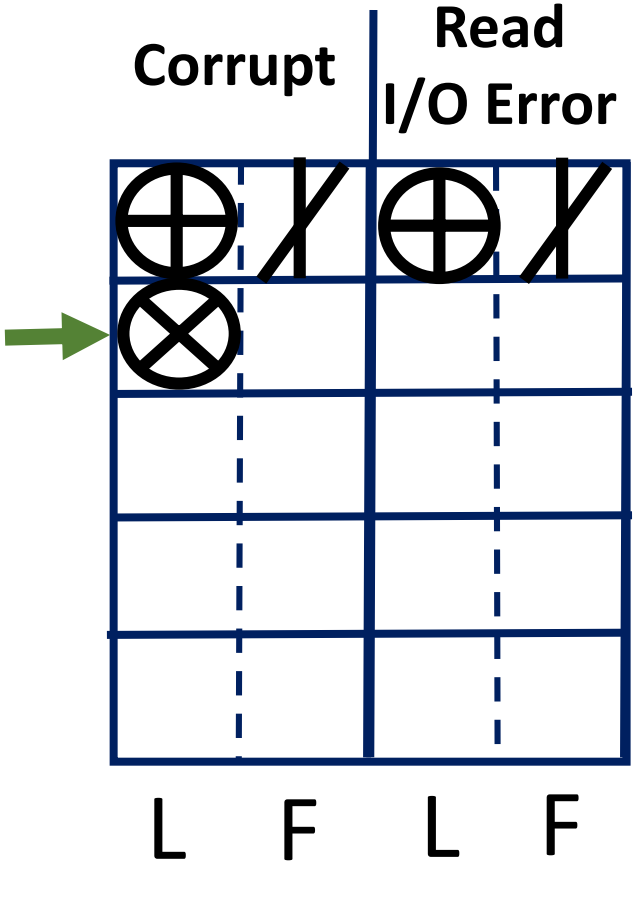
Local Behavior



On-disk Structures

- appendonlyfile.metadata
- appendonlyfile.data
- redis_database.block_0
- redis_database.metadata
- redis_database.userdata

Global Effect



L Leader

F Follower

Local Behavior

- Crash
- No Detection/ No Recovery

Global Effect

- Unavailability
- Reduced Redundancy
- Corruption

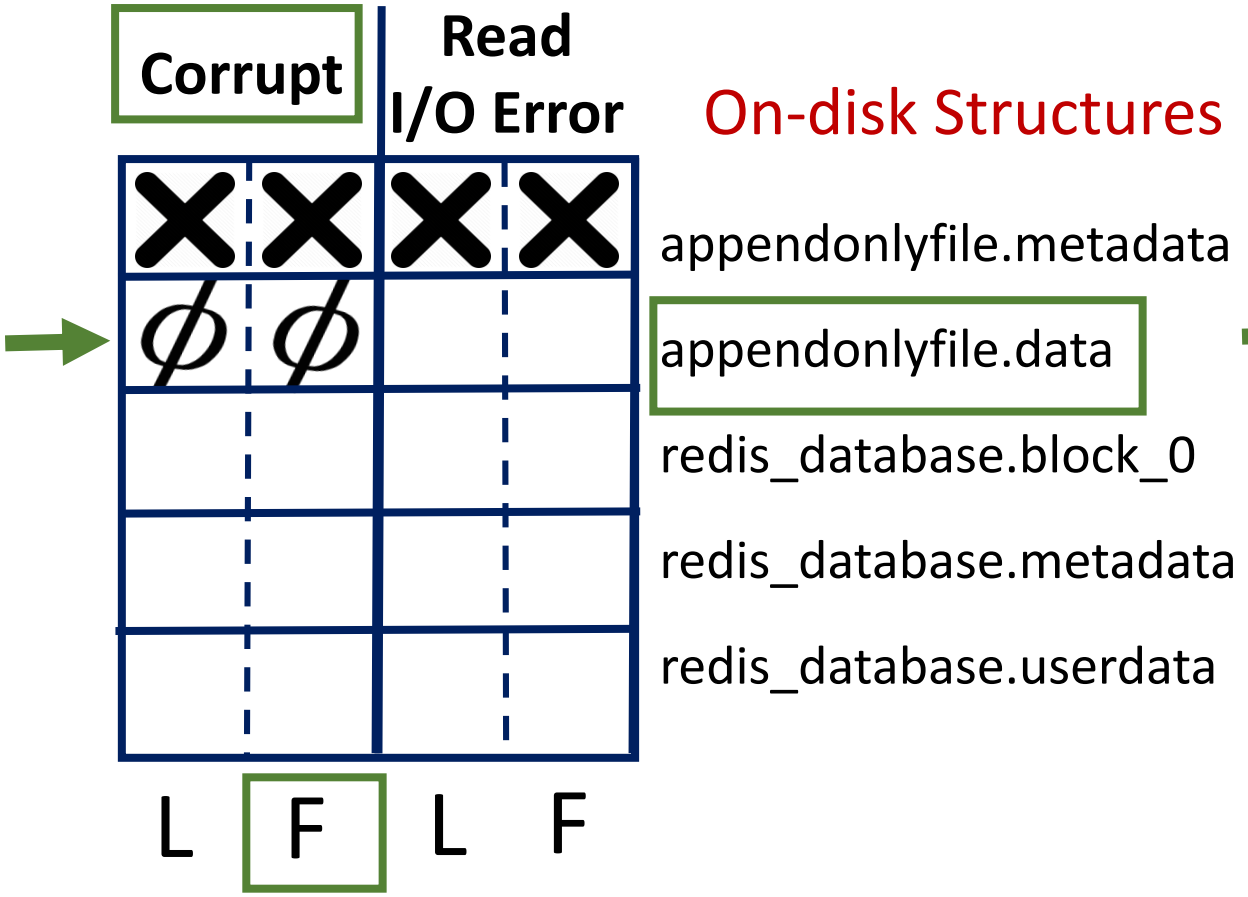
Redis: Behavior Analysis

Read Workload

L Leader

F Follower

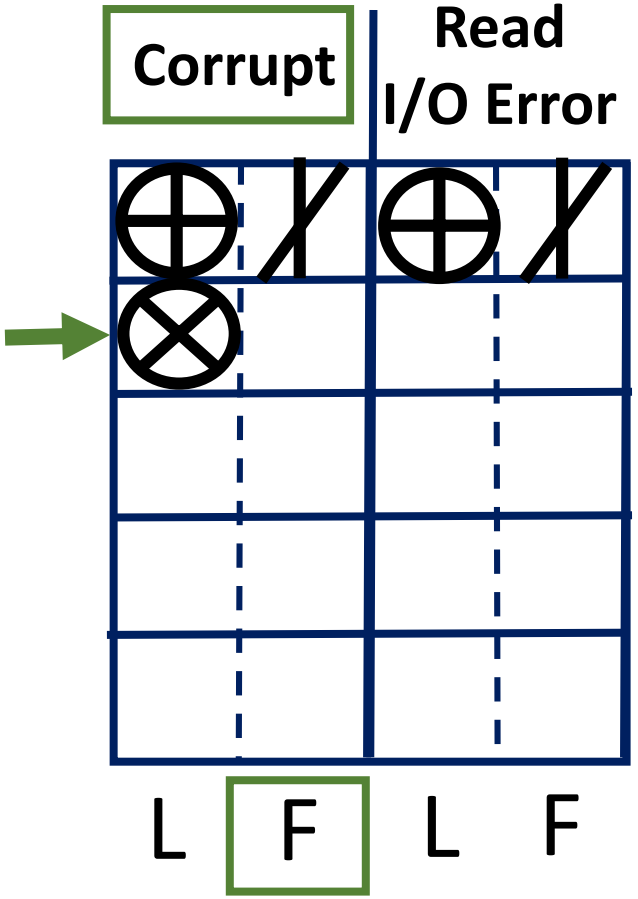
Local Behavior



On-disk Structures

- appendonlyfile.metadata
- appendonlyfile.data
- redis_database.block_0
- redis_database.metadata
- redis_database.userdata

Global Effect



Local Behavior

- Crash
- No Detection/ No Recovery

Global Effect

- Unavailability
- Reduced Redundancy
- Corruption

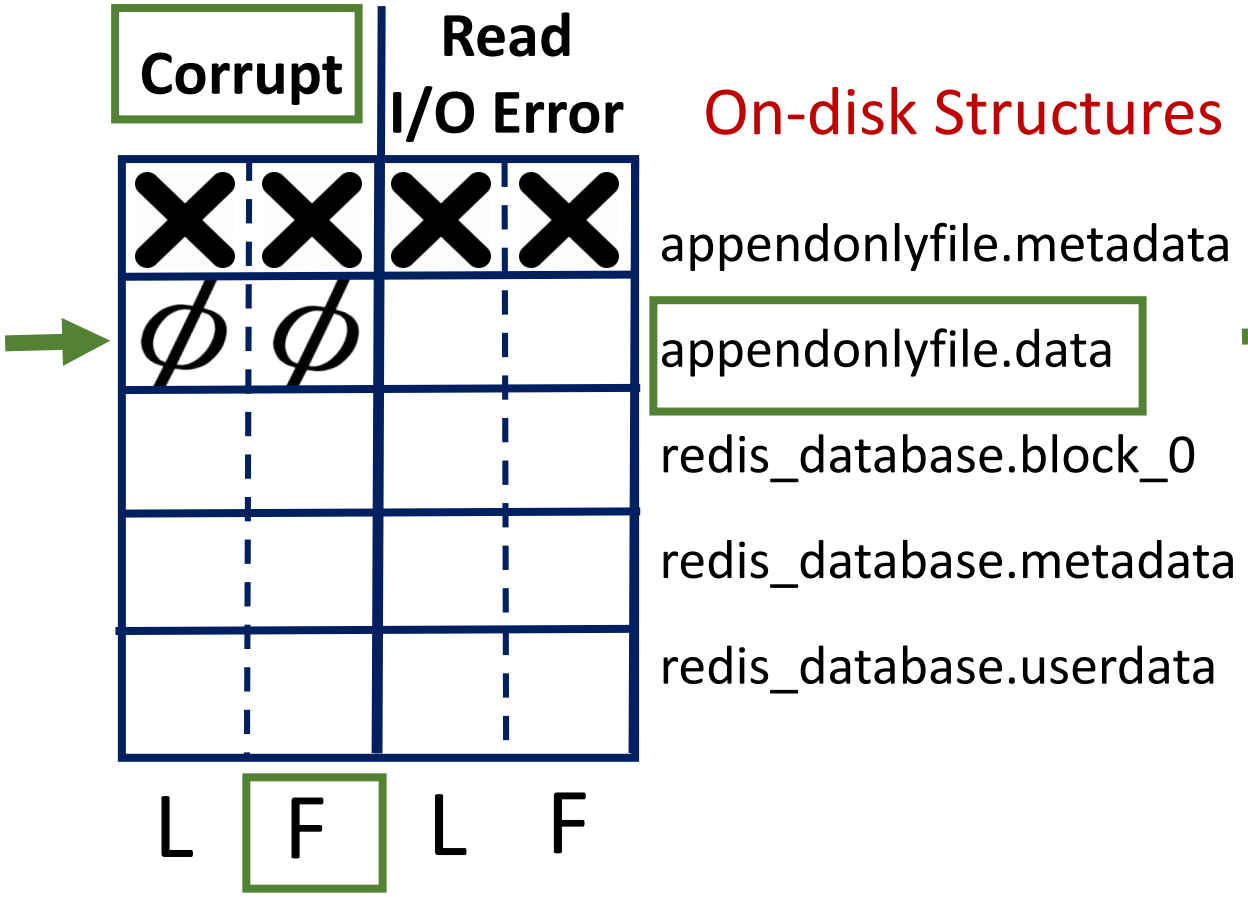
Redis: Behavior Analysis

Read Workload

L Leader

F Follower

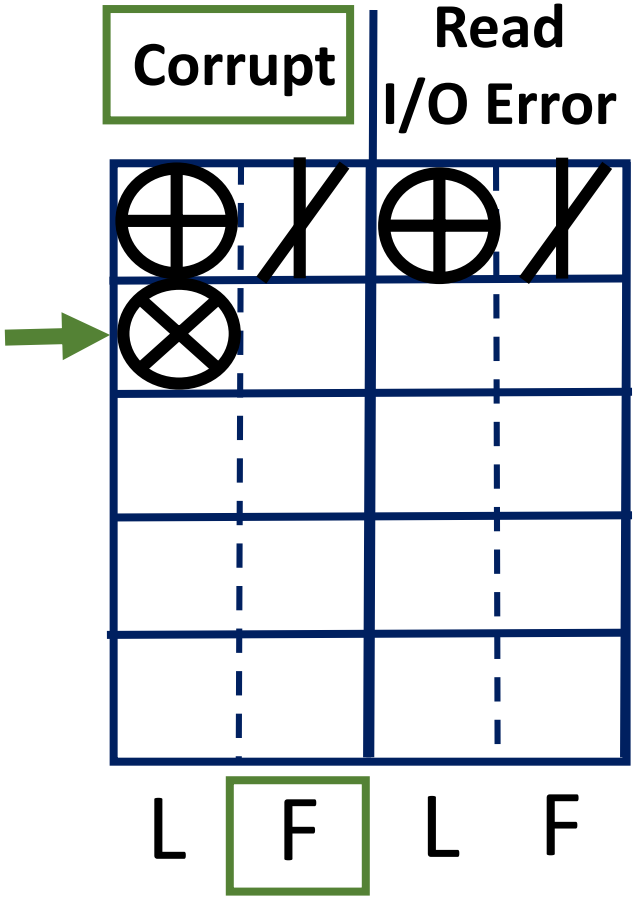
Local Behavior



On-disk Structures

- appendonlyfile.metadata
- appendonlyfile.data
- redis_database.block_0
- redis_database.metadata
- redis_database.userdata

Global Effect



Local Behavior

- Crash
- No Detection/ No Recovery

Global Effect

- Unavailability
- Reduced Redundancy
- Corruption
- Correct

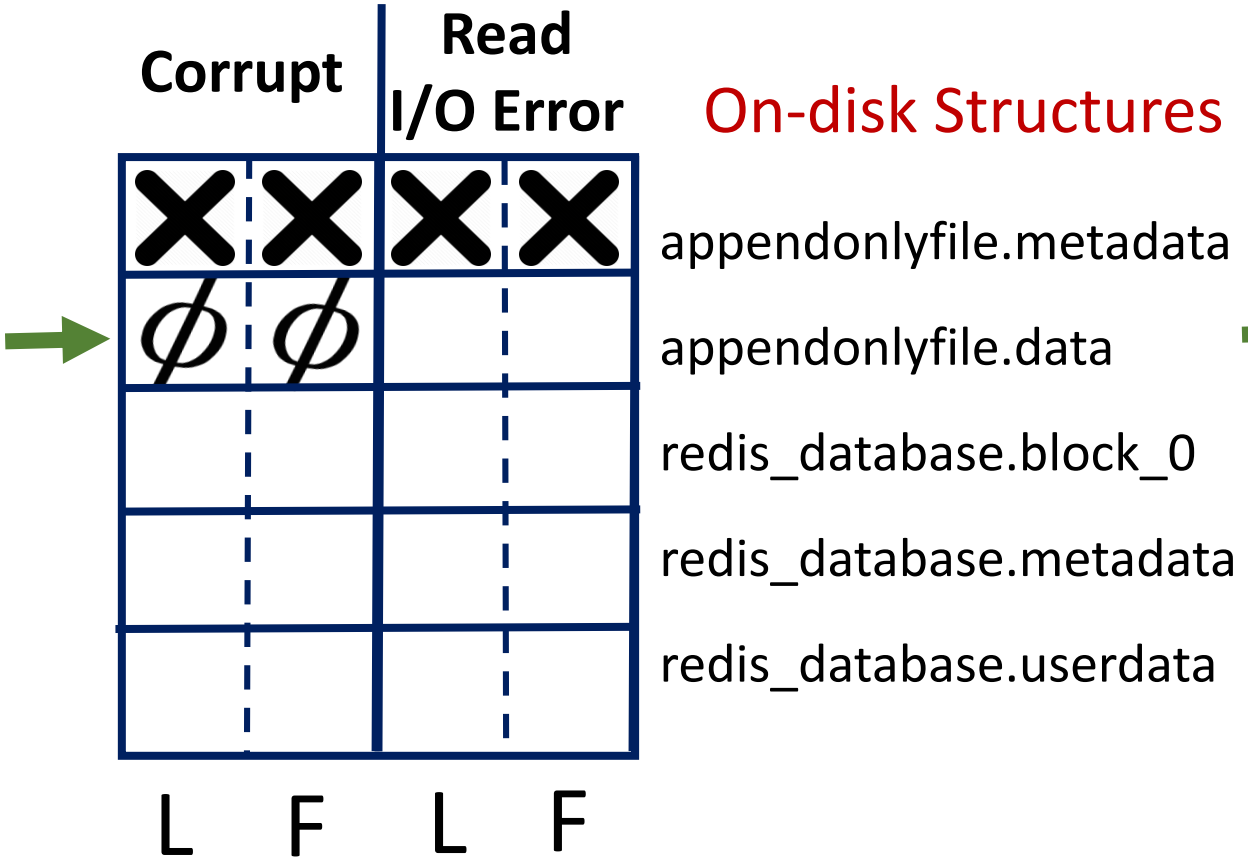
Redis: Behavior Analysis

Read Workload

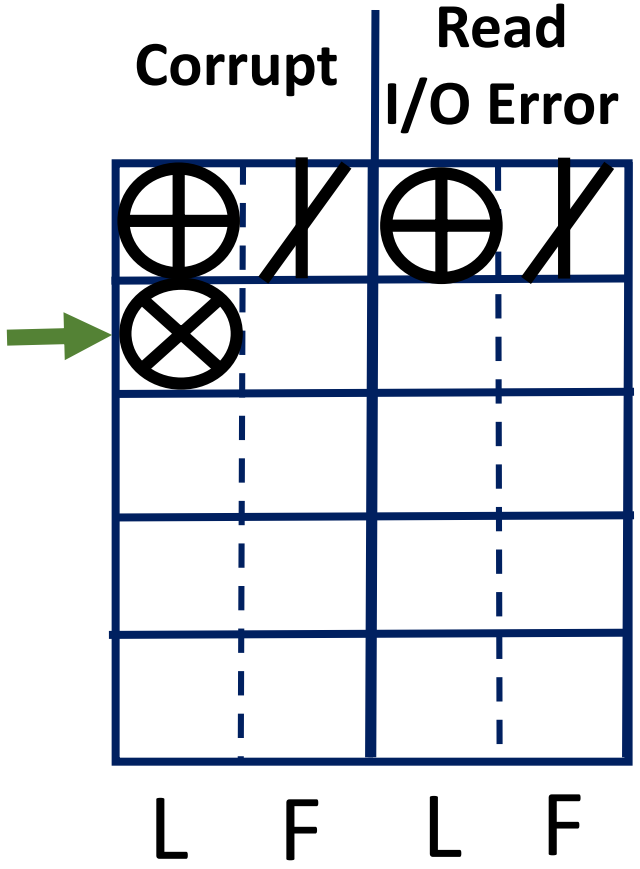
L Leader

F Follower

Local Behavior



Global Effect



Local Behavior

Crash

No Detection/ No Recovery

Global Effect

Unavailability

Reduced Redundancy

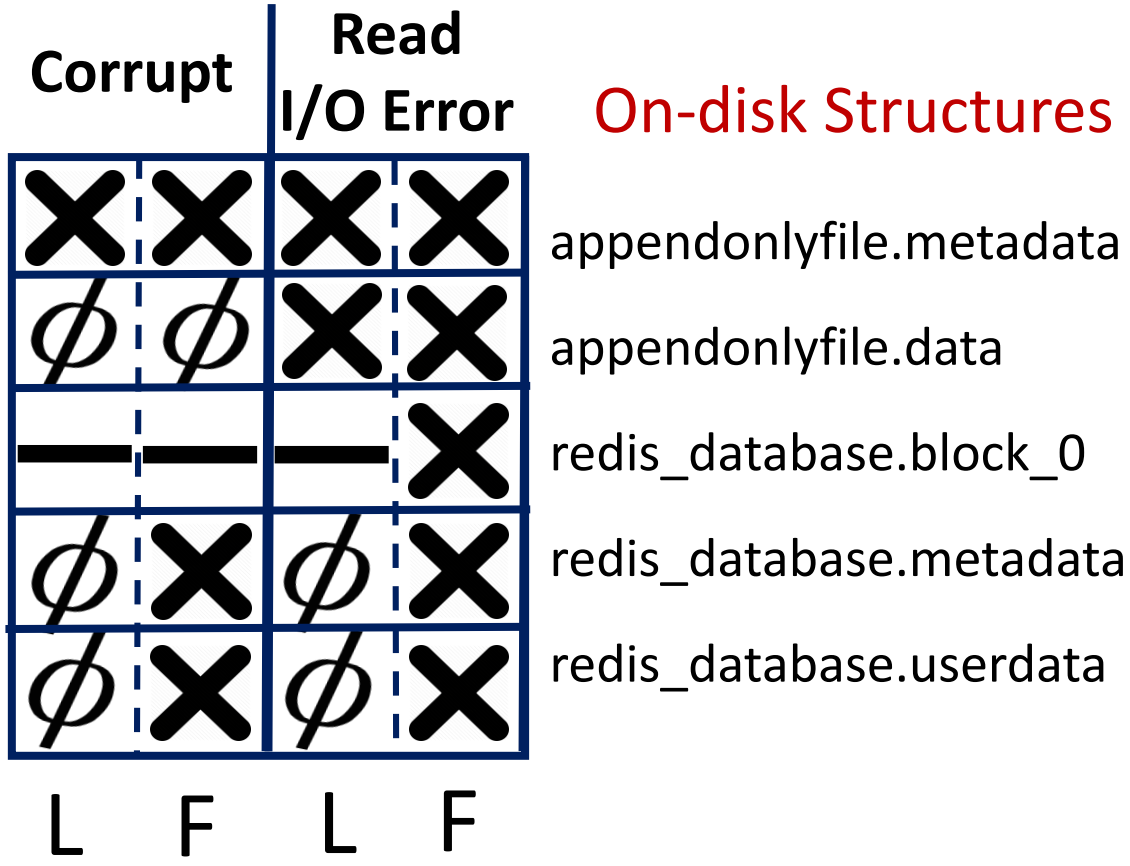
Corruption

Correct

Redis: Behavior Analysis

Read Workload

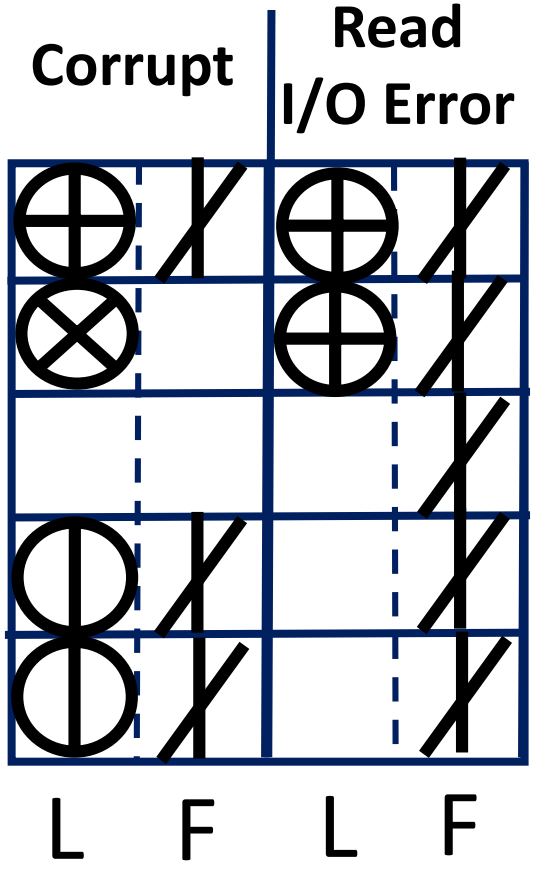
Local Behavior



L Leader

F Follower

Global Effect



Local Behavior

- Crash
- No Detection/ No Recovery
- Retry

Global Effect

- Unavailability
- Reduced Redundancy
- Corruption
- Correct
- Write Unavailability

Other Systems

Other Systems

Metadata stores: ZooKeeper, LogCabin

Wide column store: Cassandra

Document stores: MongoDB

Distributed databases: RethinkDB, CockroachDB

Message Queues: Kafka

Outline

Introduction

Fault Injection

System Behavior Analysis

Major Results

- Redundancy Does not Provide Fault Tolerance

Observations Across Systems

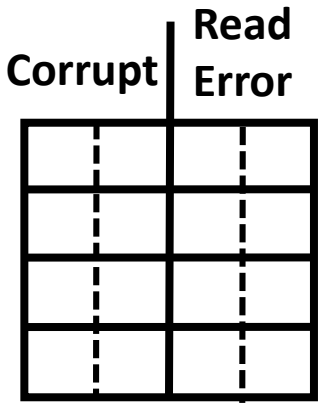
Conclusion

Redundancy Does not Provide Fault Tolerance

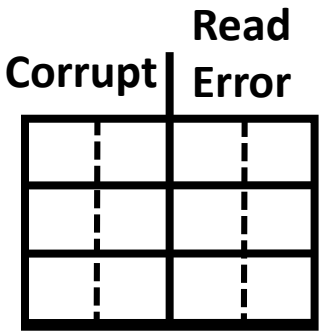
Redundancy Does not Provide Fault Tolerance

Redundancy Does not Provide Fault Tolerance

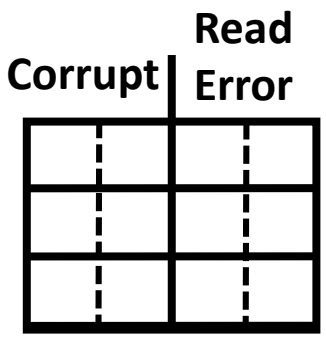
Redis Read



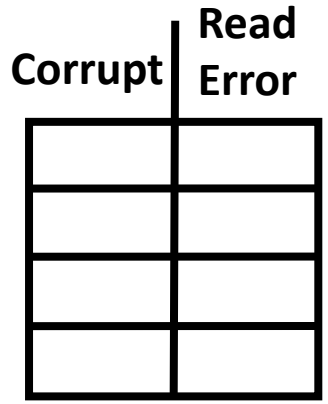
Kafka Read



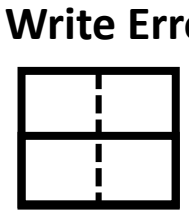
Kafka Write



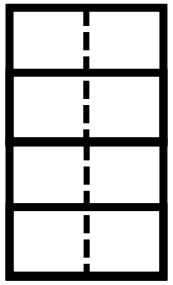
Cassandra Read



ZooKeeper Write

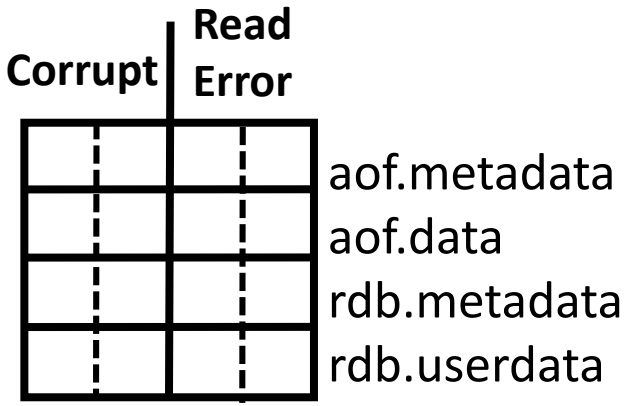


RethinkDB Read

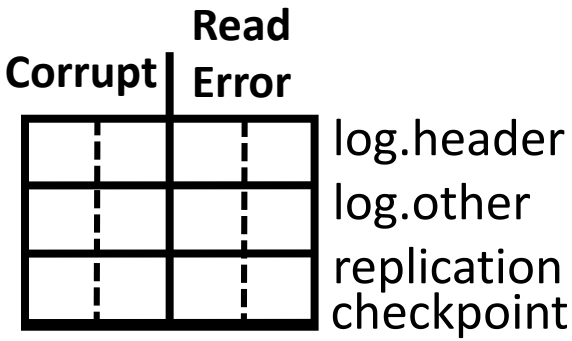


Redundancy Does not Provide Fault Tolerance

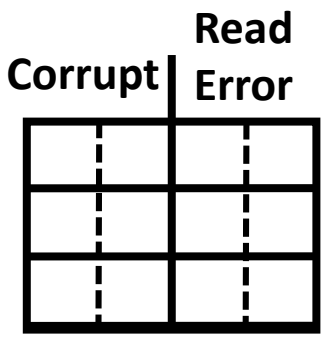
Redis Read



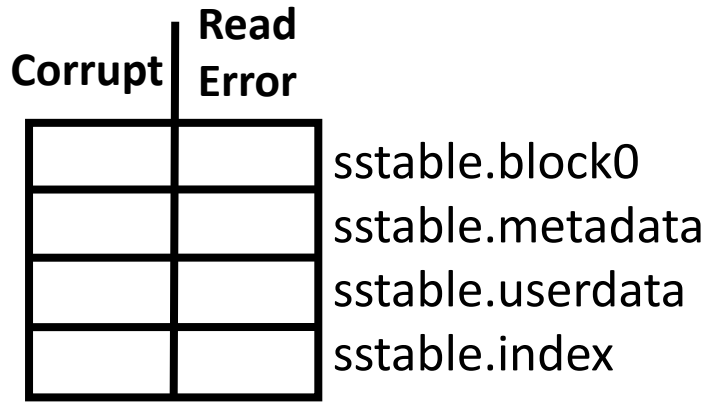
Kafka Read



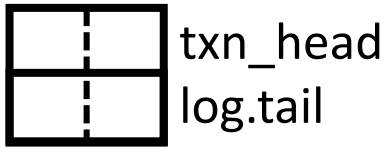
Kafka Write



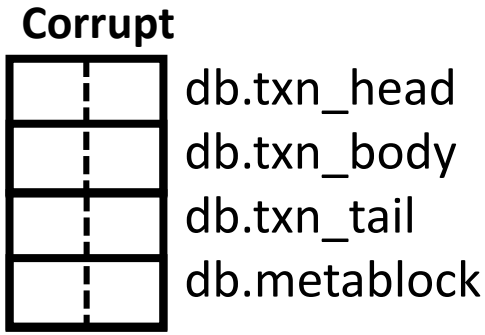
Cassandra Read



ZooKeeper Write

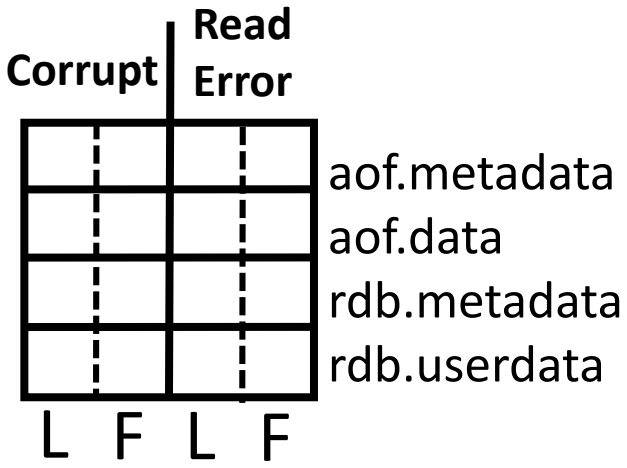


RethinkDB Read

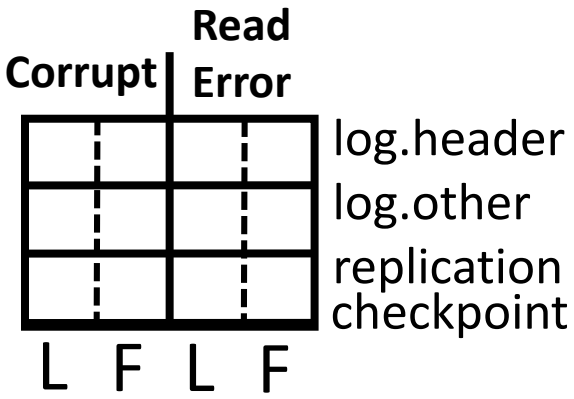


Redundancy Does not Provide Fault Tolerance

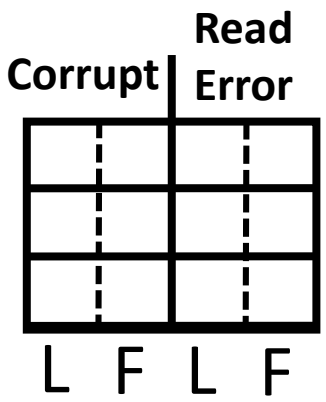
Redis Read



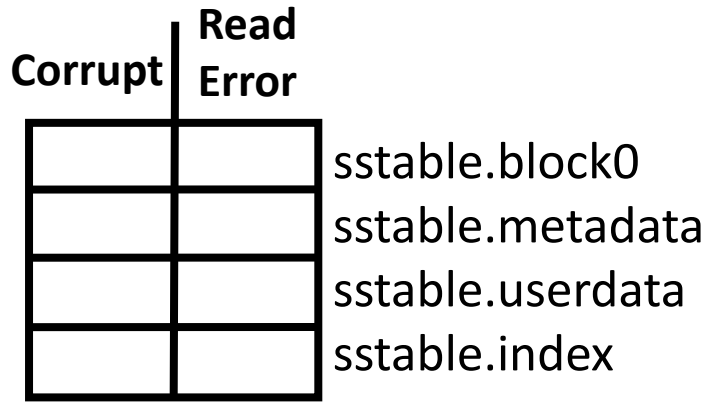
Kafka Read



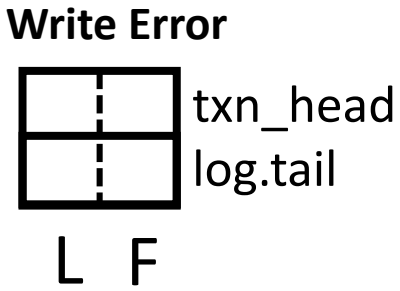
Kafka Write



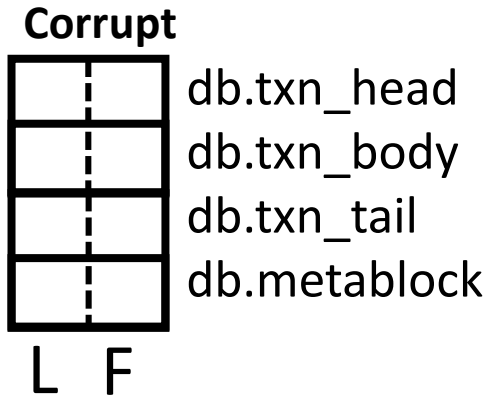
Cassandra Read



ZooKeeper Write

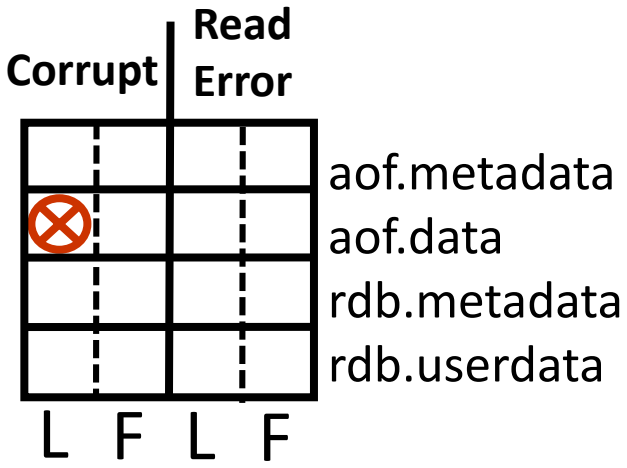


RethinkDB Read

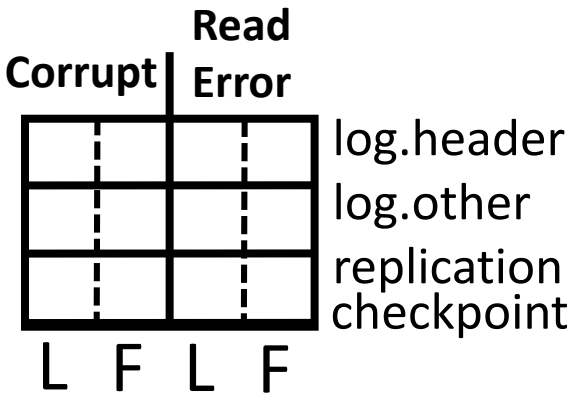


Redundancy Does not Provide Fault Tolerance

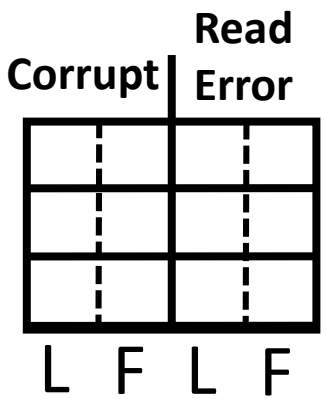
Redis Read



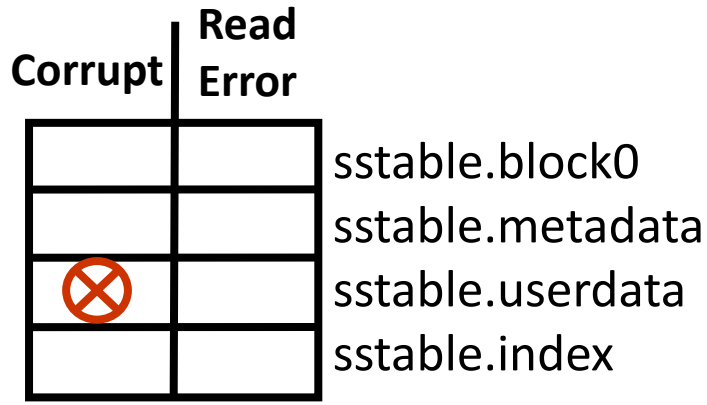
Kafka Read



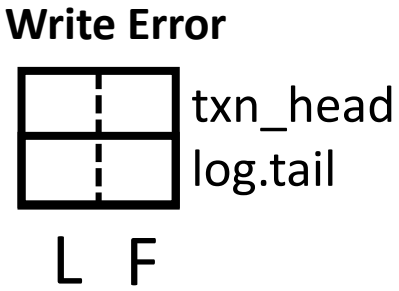
Kafka Write



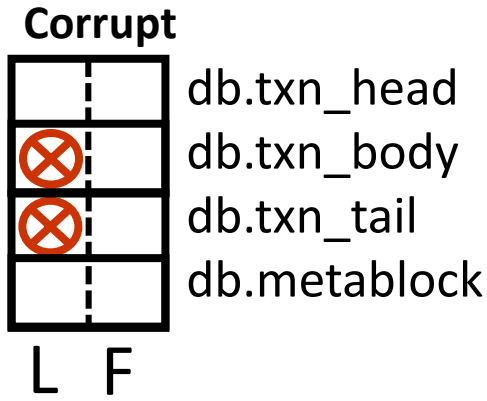
Cassandra Read



ZooKeeper Write



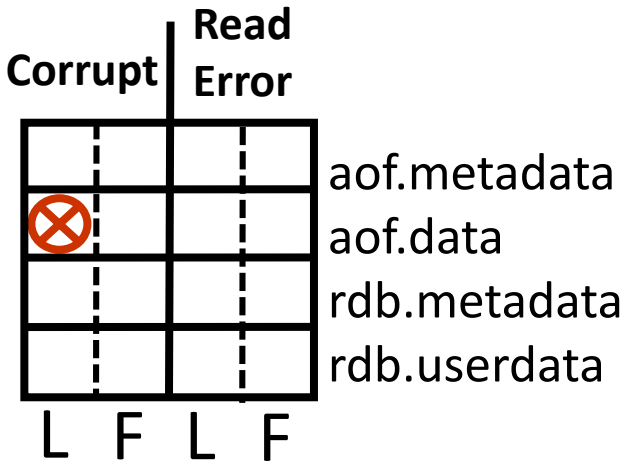
RethinkDB Read



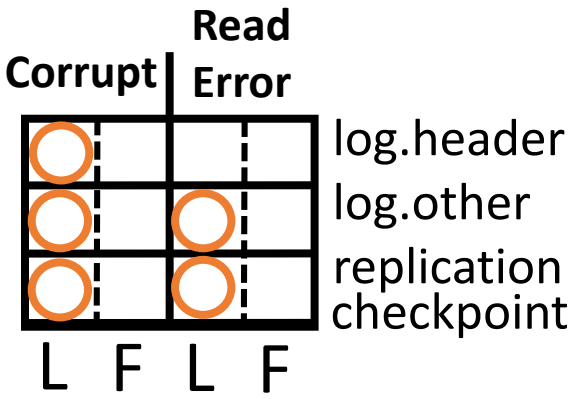
Corruption

Redundancy Does not Provide Fault Tolerance

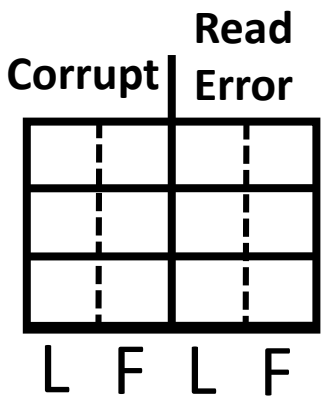
Redis Read



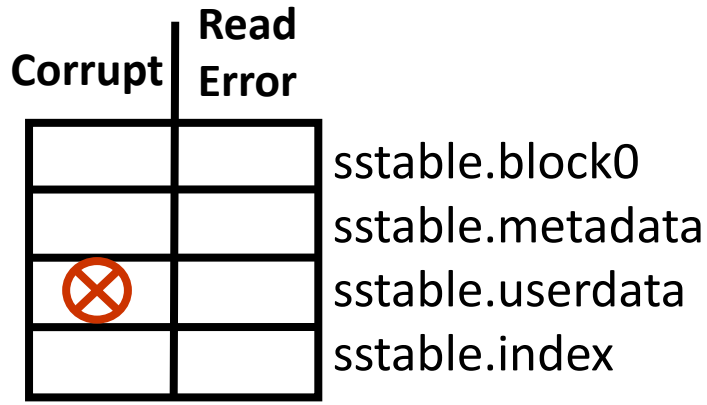
Kafka Read



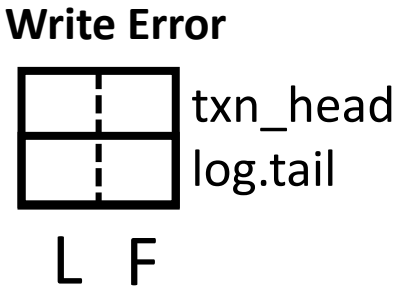
Kafka Write



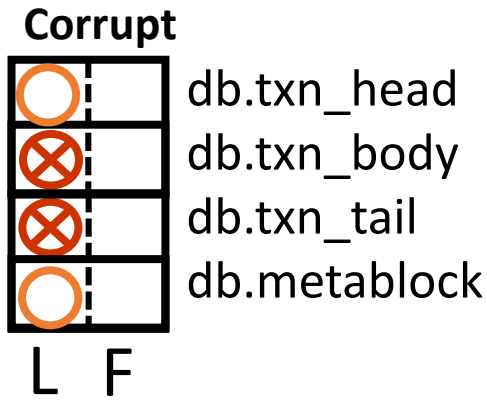
Cassandra Read



ZooKeeper Write



RethinkDB Read

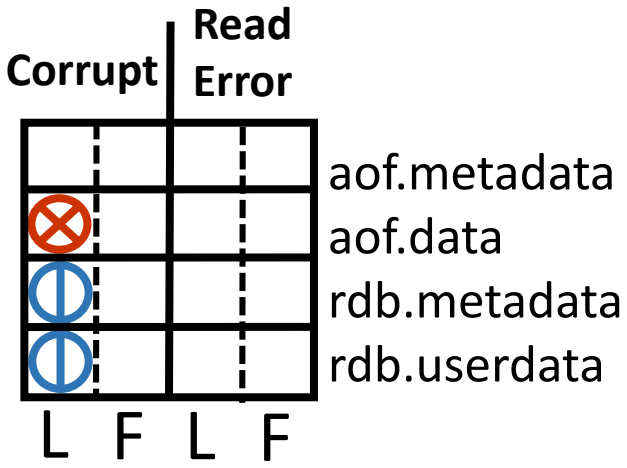


Corruption

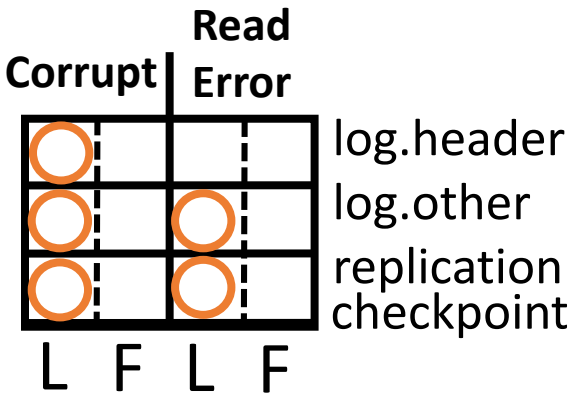
Data Loss

Redundancy Does not Provide Fault Tolerance

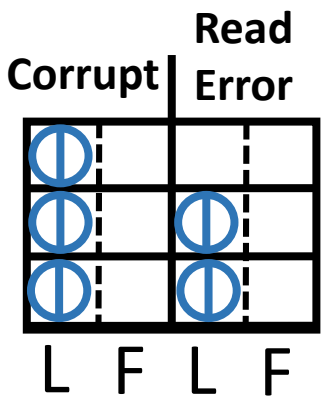
Redis Read



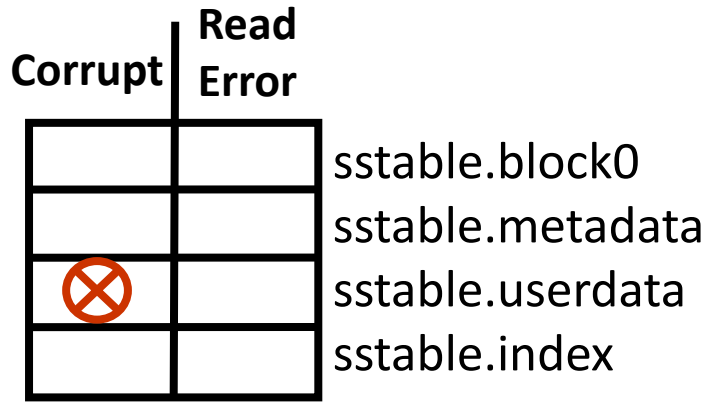
Kafka Read



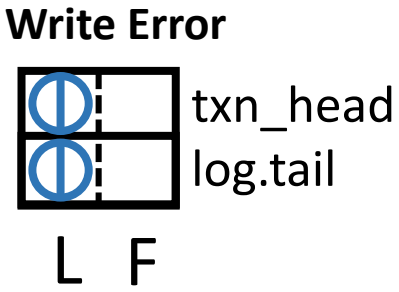
Kafka Write



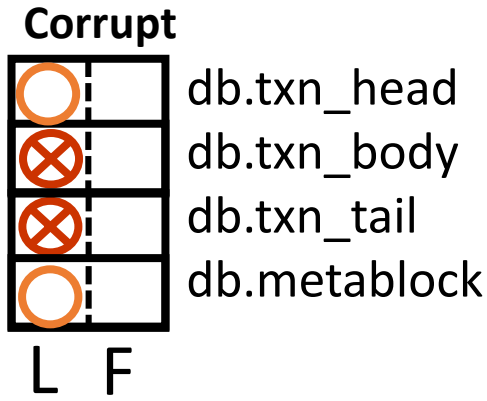
Cassandra Read

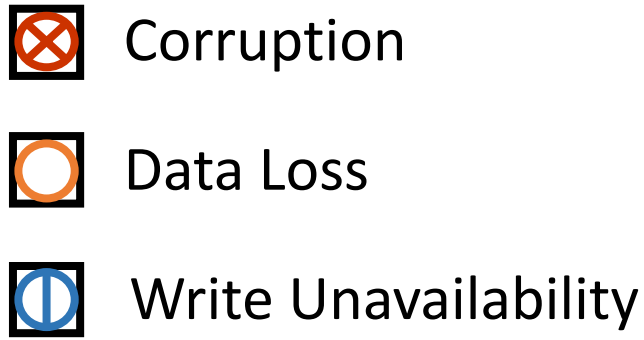


ZooKeeper Write



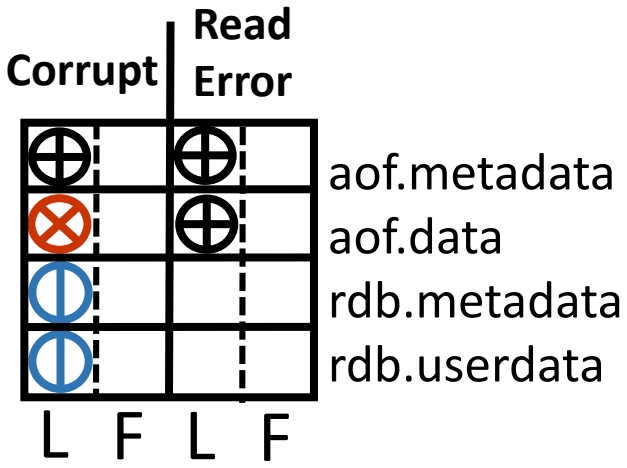
RethinkDB Read



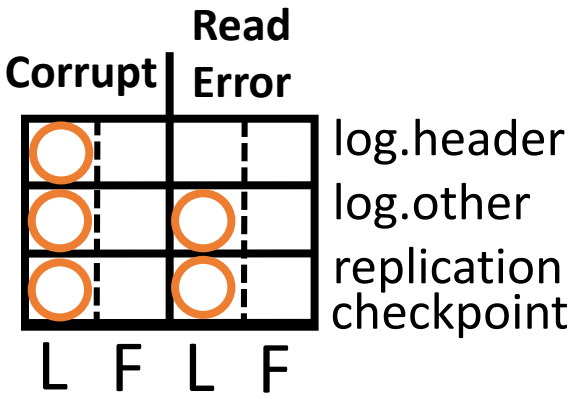


Redundancy Does not Provide Fault Tolerance

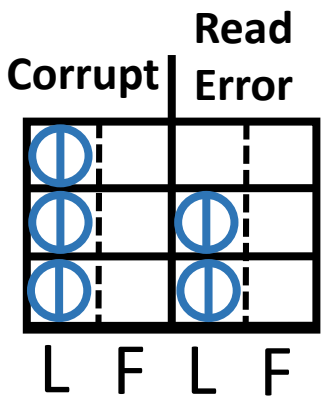
Redis Read



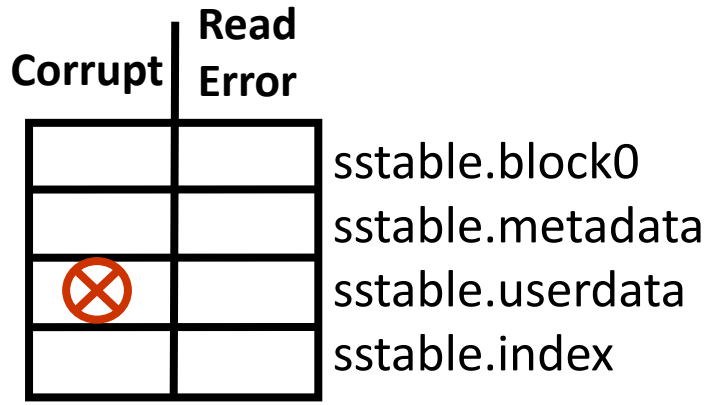
Kafka Read



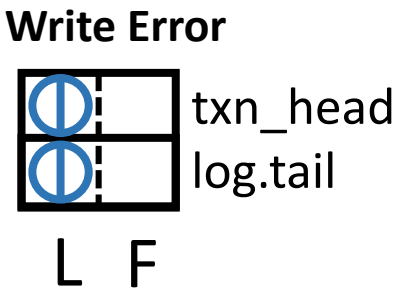
Kafka Write



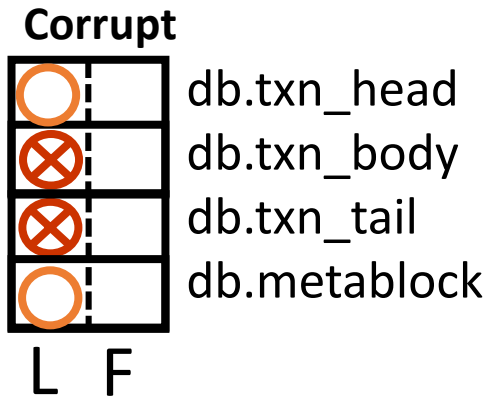
Cassandra Read



ZooKeeper Write



RethinkDB Read



Corruption

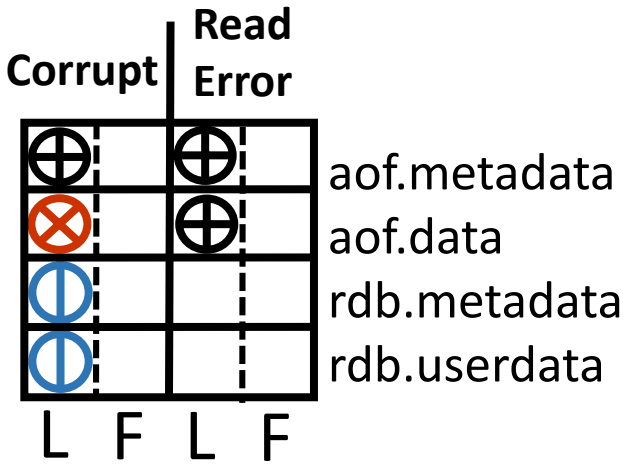
Unavailability

Data Loss

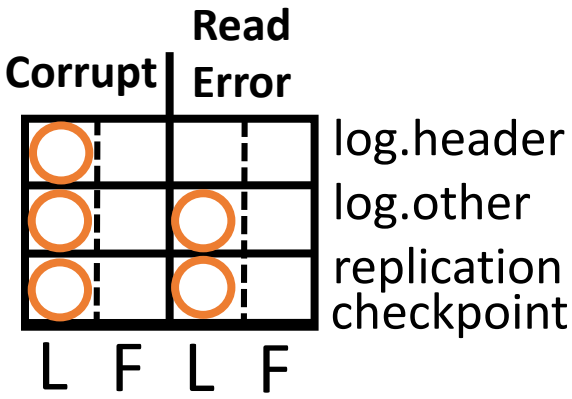
Write Unavailability

Redundancy Does not Provide Fault Tolerance

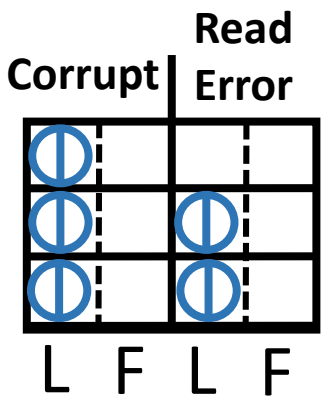
Redis Read



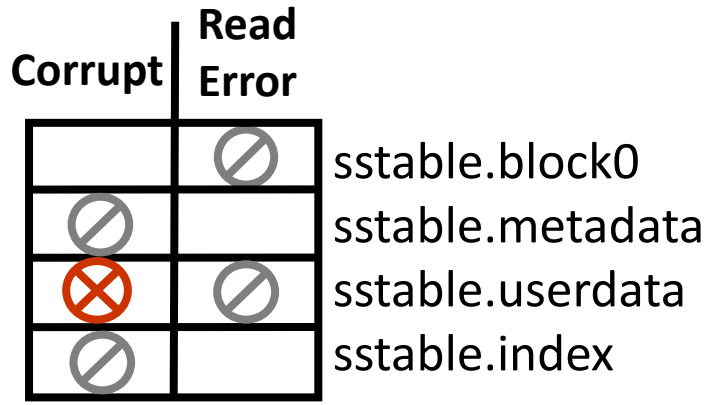
Kafka Read



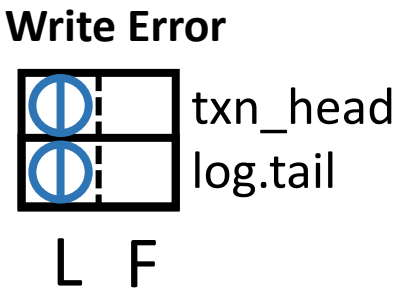
Kafka Write



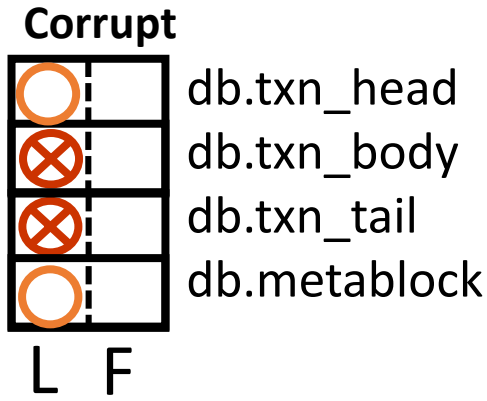
Cassandra Read

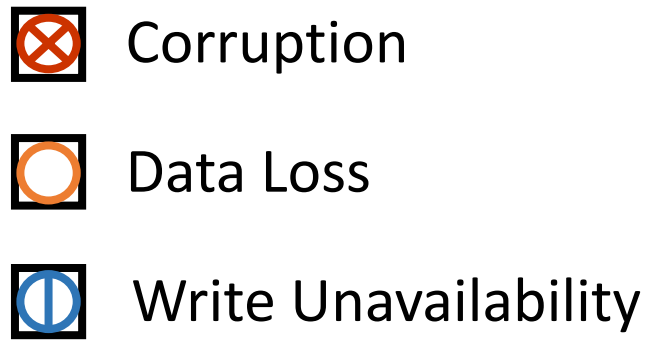


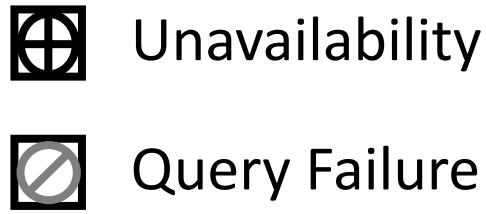
ZooKeeper Write



RethinkDB Read

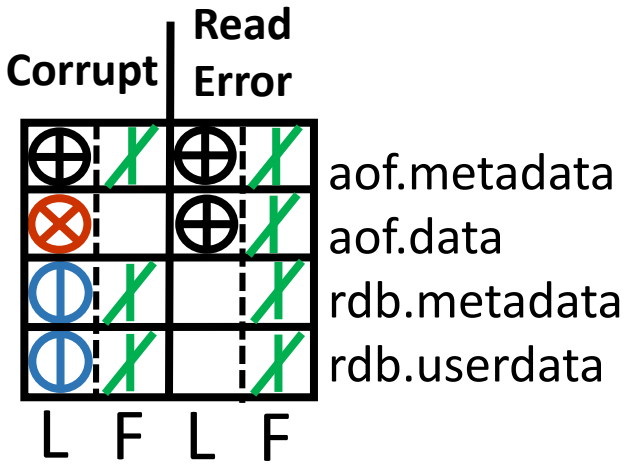




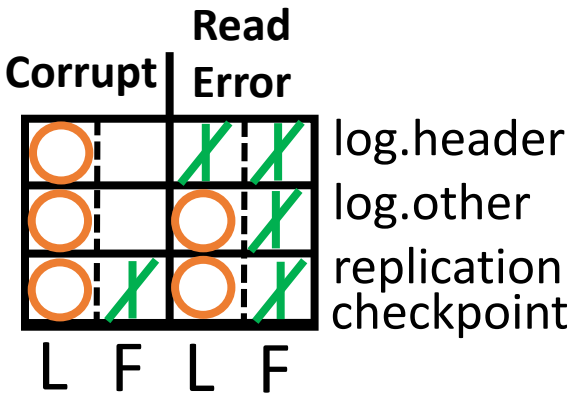


Redundancy Does not Provide Fault Tolerance

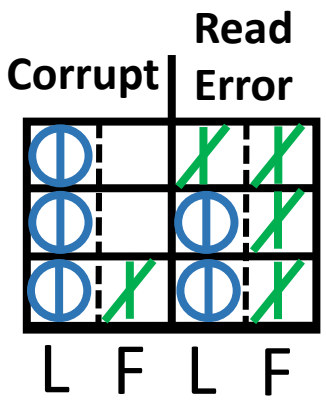
Redis Read



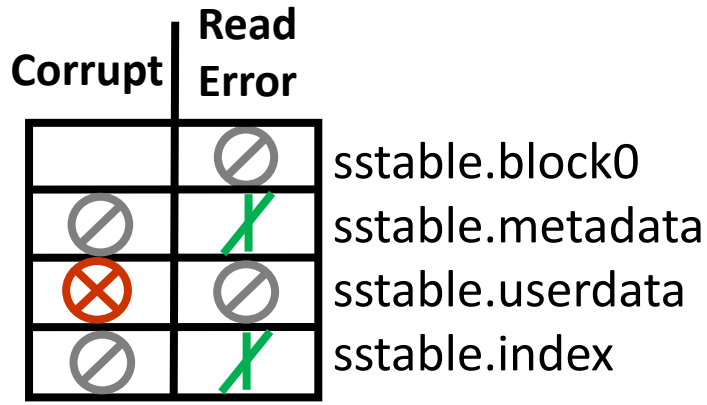
Kafka Read



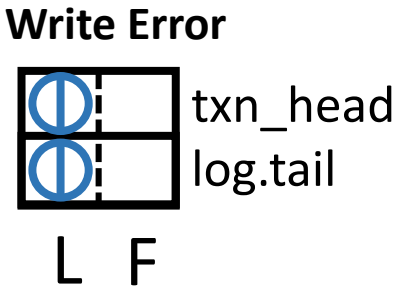
Kafka Write



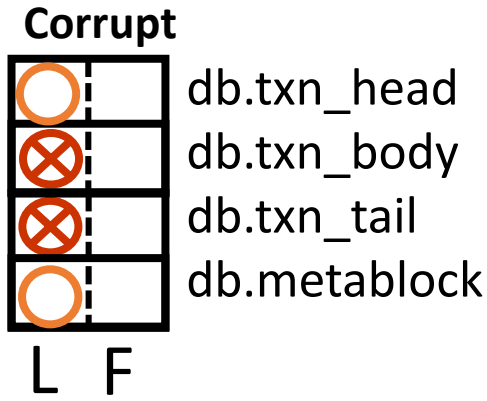
Cassandra Read

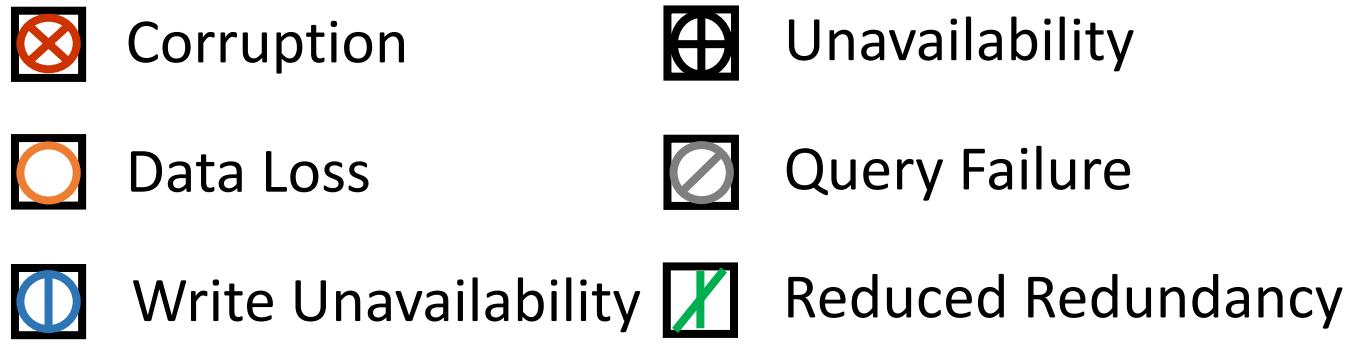


ZooKeeper Write



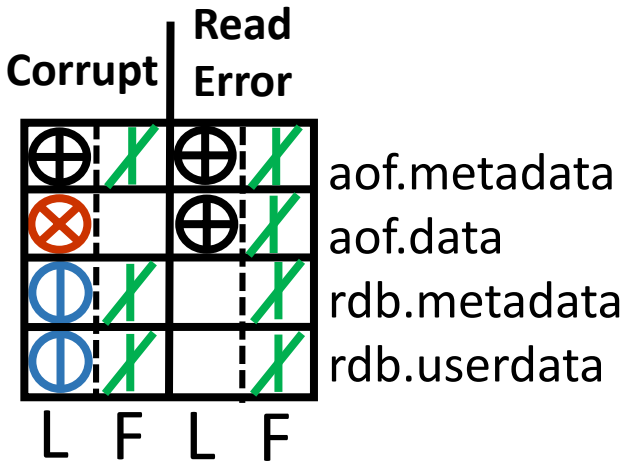
RethinkDB Read



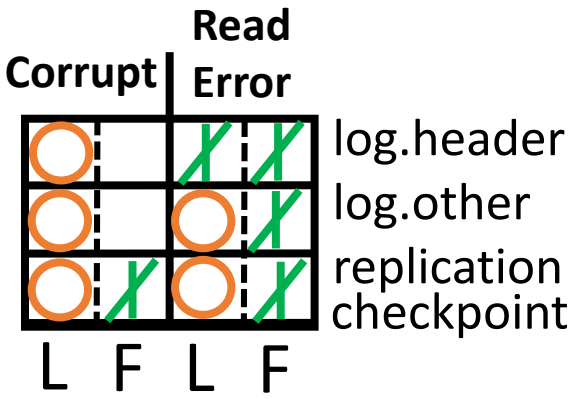


Redundancy Does not Provide Fault Tolerance

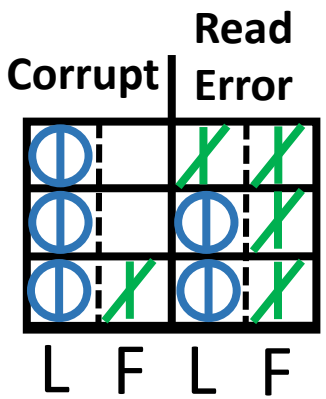
Redis Read



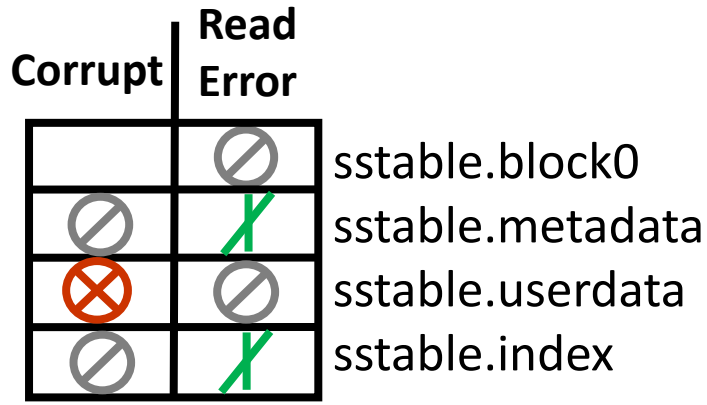
Kafka Read



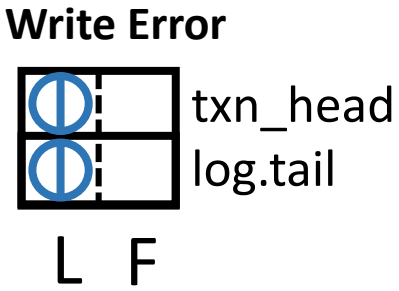
Kafka Write



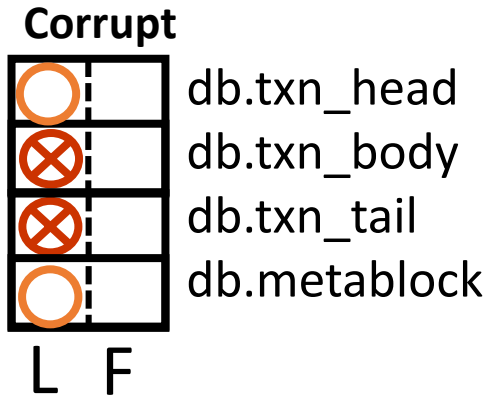
Cassandra Read

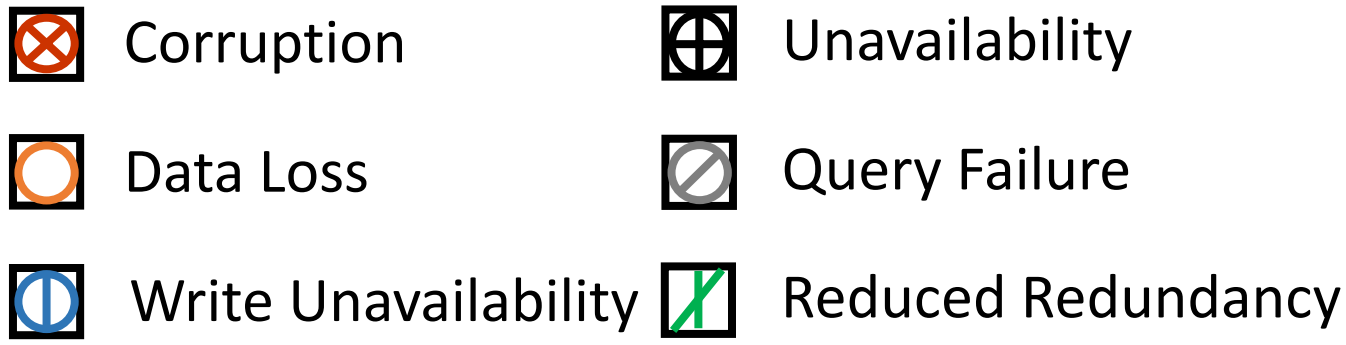


ZooKeeper Write



RethinkDB Read

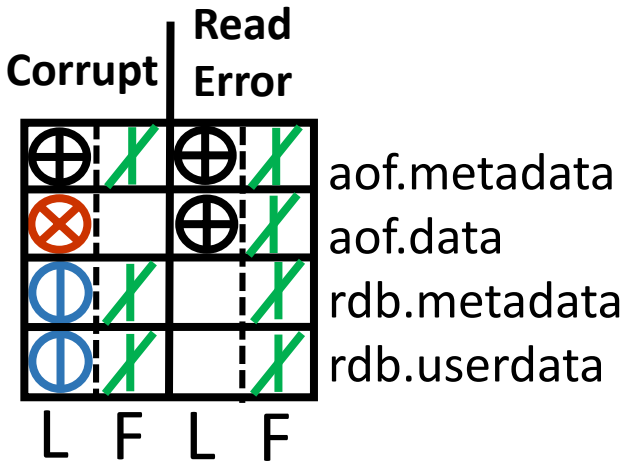




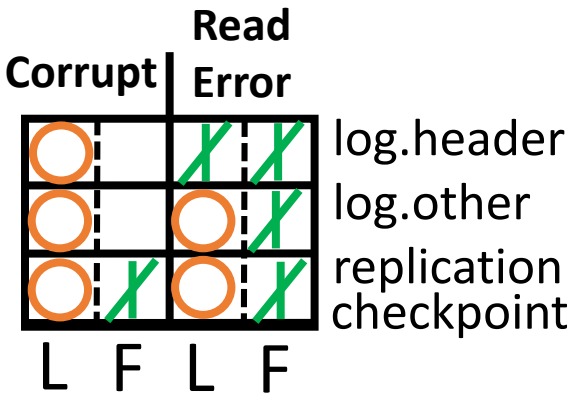
Harmful global effects despite redundancy

Redundancy Does not Provide Fault Tolerance

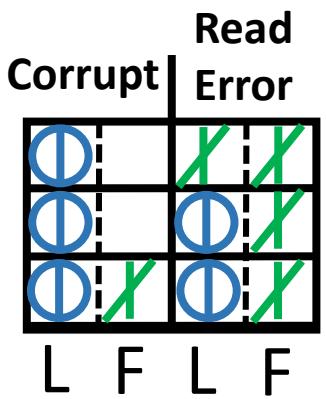
Redis Read



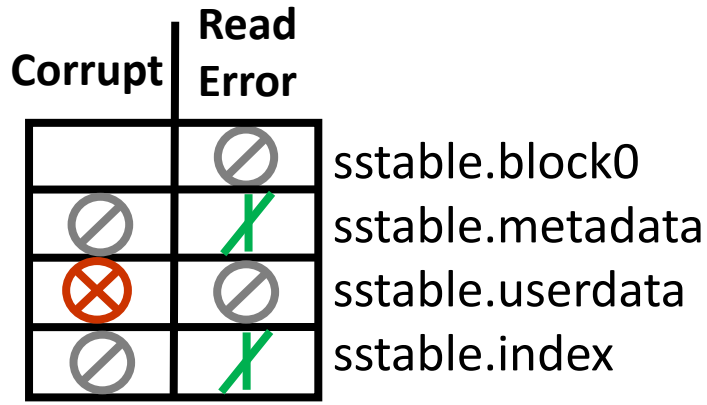
Kafka Read



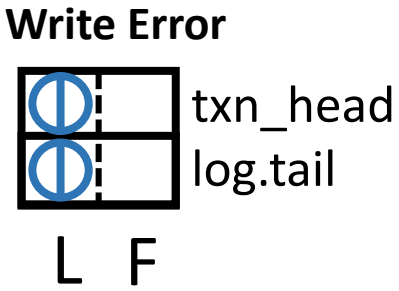
Kafka Write



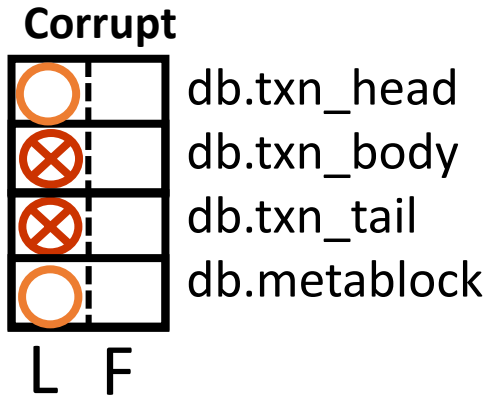
Cassandra Read

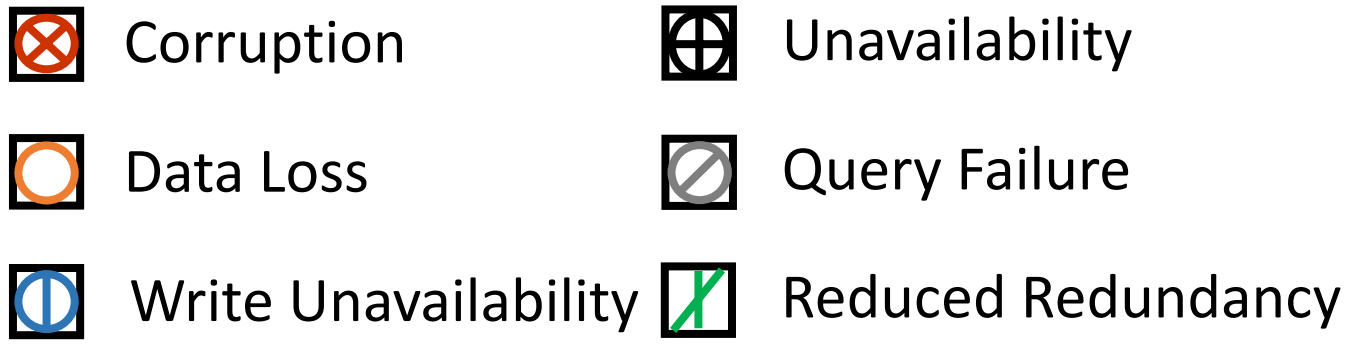


ZooKeeper Write



RethinkDB Read





Harmful global effects despite redundancy

Not simple implementation bugs - fundamental problems across multiple systems!

Outline

Introduction

Fault Injection

System Behavior Analysis

Major Results

Observations Across Systems

- Faults are Often Undetected Locally
- Crashing: Common Local Reaction
- Crash and Corruption Handling are Entangled
- Unsafe interaction between local and global protocols

Conclusion

Faults are Often Undetected Locally

Faults are Often Undetected Locally

Undetected faults may lead to harmful global effect

Faults are Often Undetected Locally

Undetected faults may lead to harmful global effect

- Locally undetected corruption → global silent corruption

Faults are Often Undetected Locally

Undetected faults may lead to harmful global effect

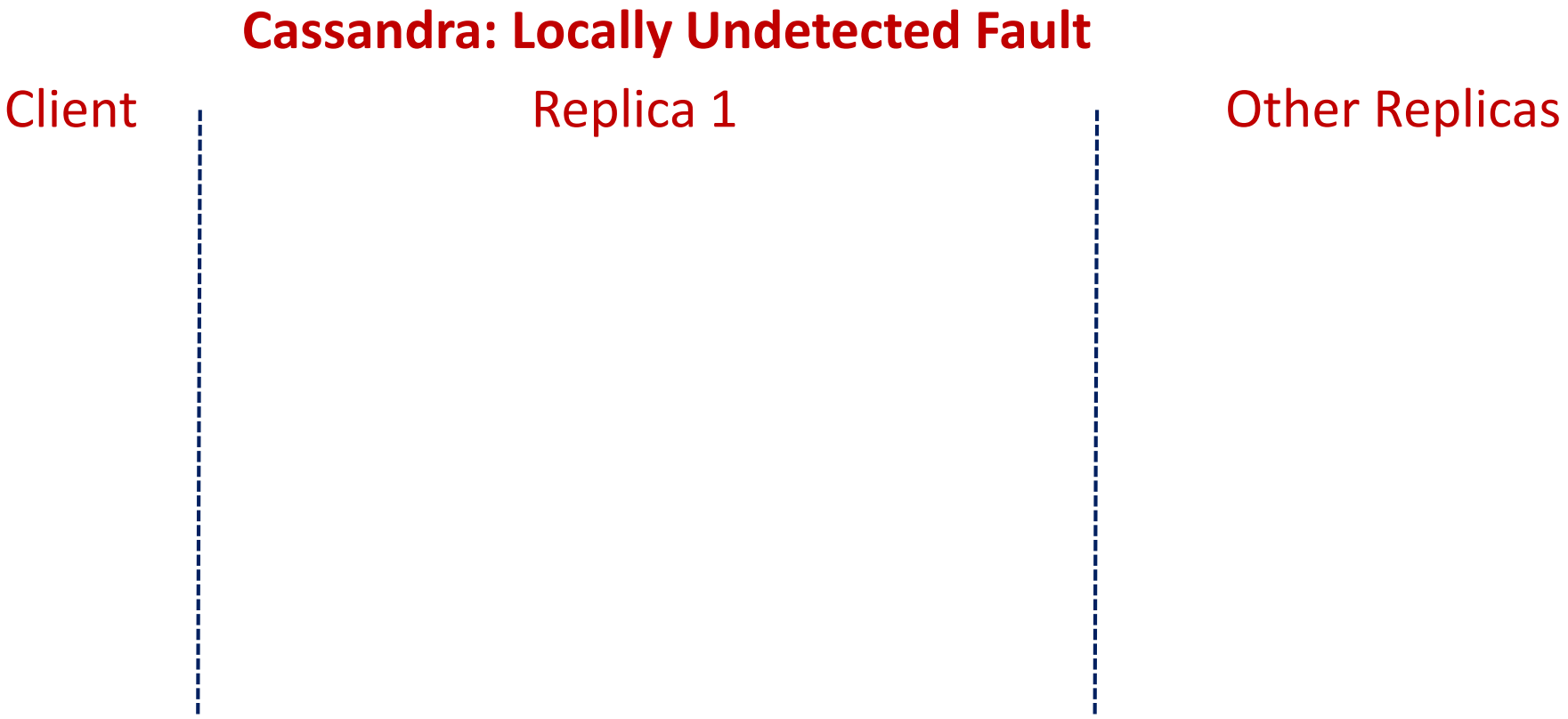
- Locally undetected corruption → global silent corruption

Cassandra: Locally Undetected Fault

Faults are Often Undetected Locally

Undetected faults may lead to harmful global effect

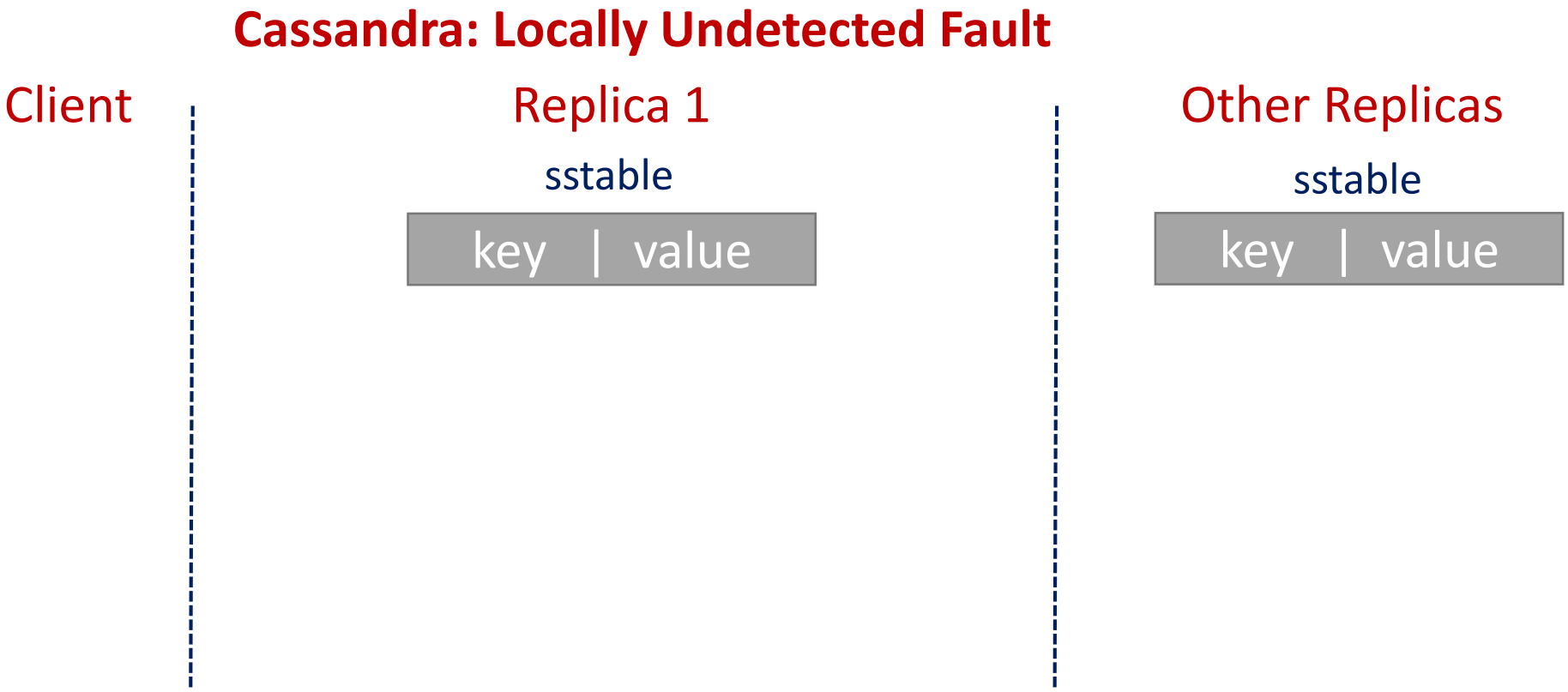
- Locally undetected corruption → global silent corruption



Faults are Often Undetected Locally

Undetected faults may lead to harmful global effect

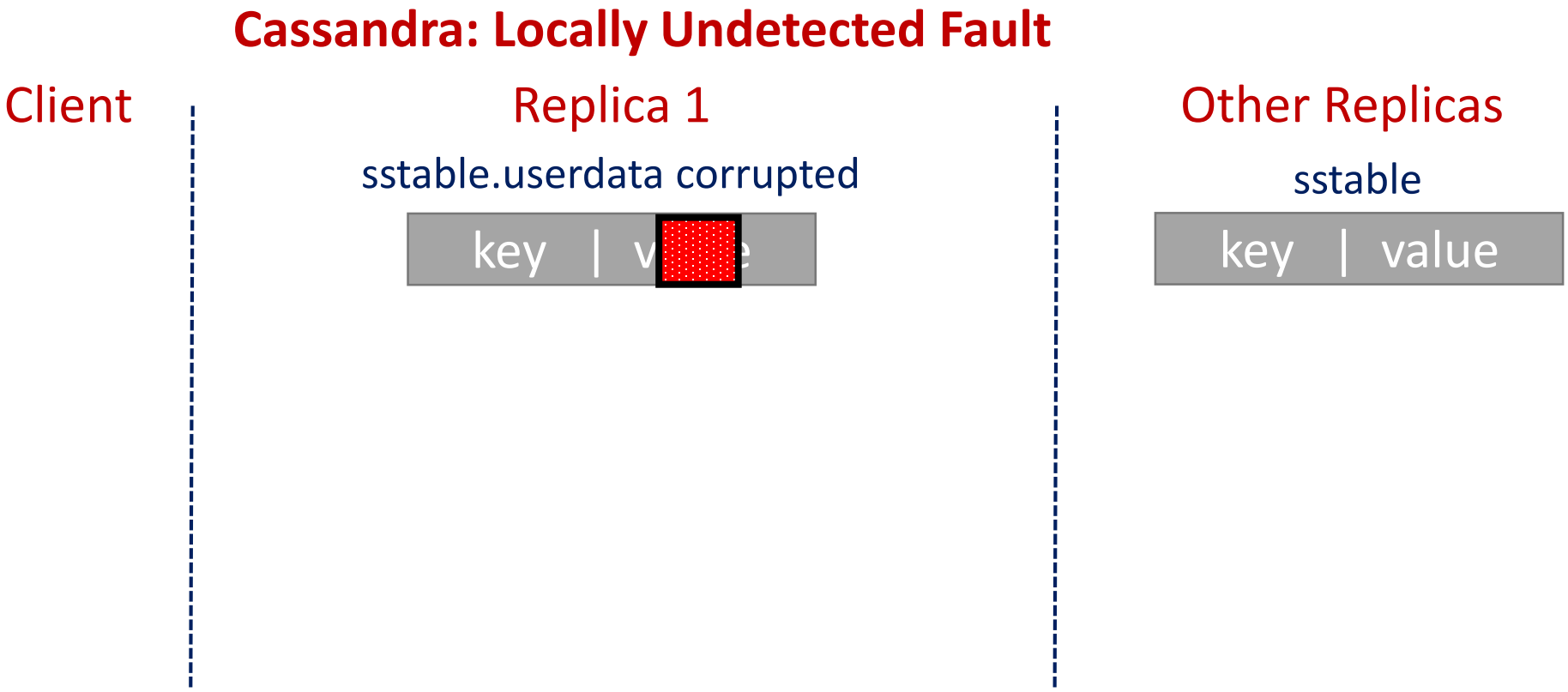
- Locally undetected corruption → global silent corruption



Faults are Often Undetected Locally

Undetected faults may lead to harmful global effect

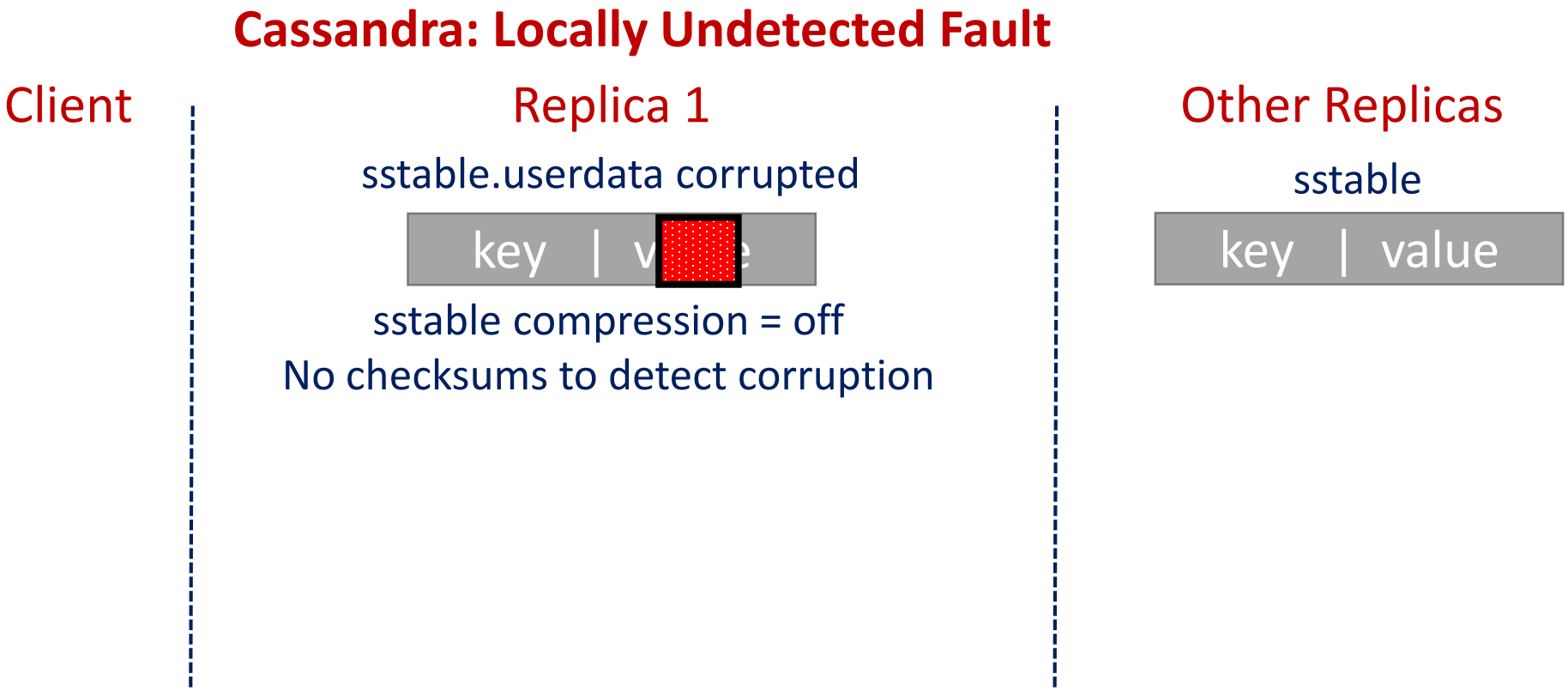
- Locally undetected corruption → global silent corruption



Faults are Often Undetected Locally

Undetected faults may lead to harmful global effect

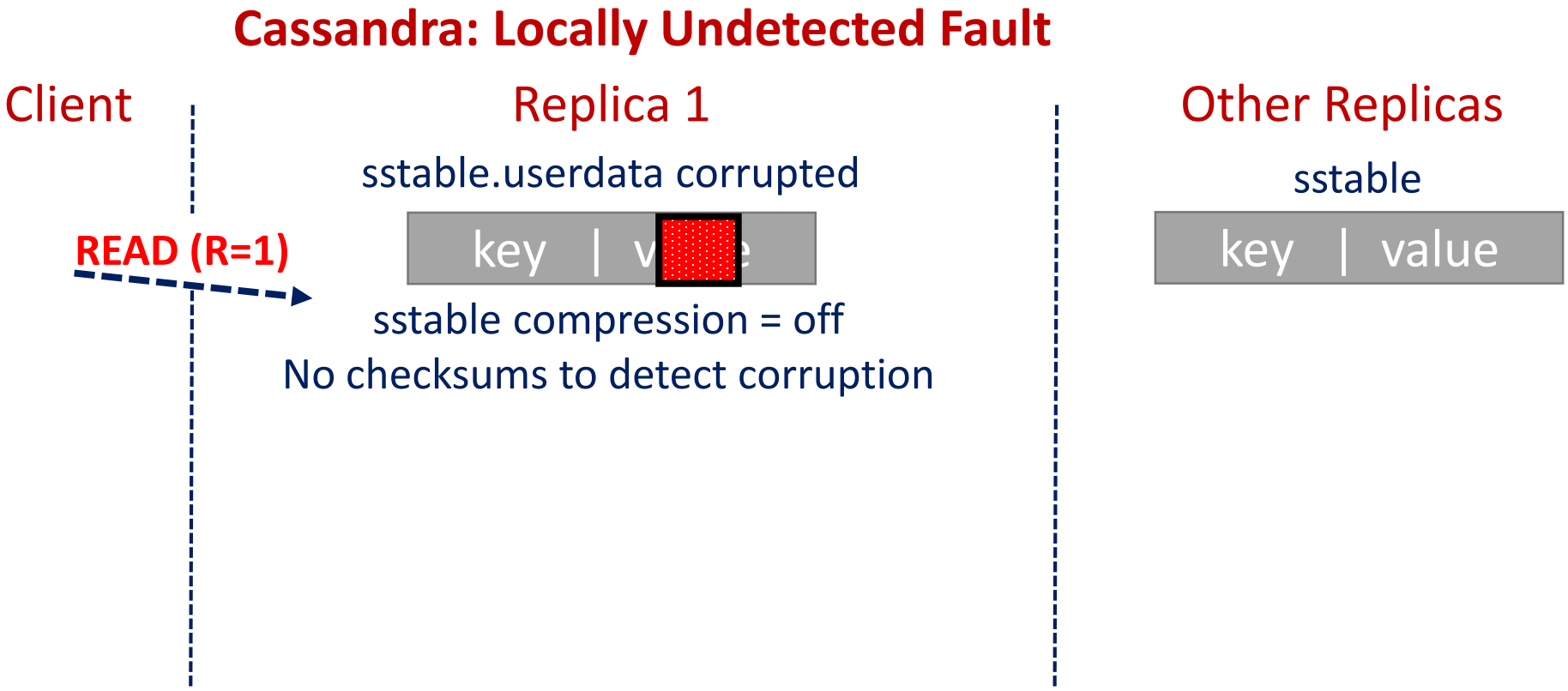
- Locally undetected corruption → global silent corruption



Faults are Often Undetected Locally

Undetected faults may lead to harmful global effect

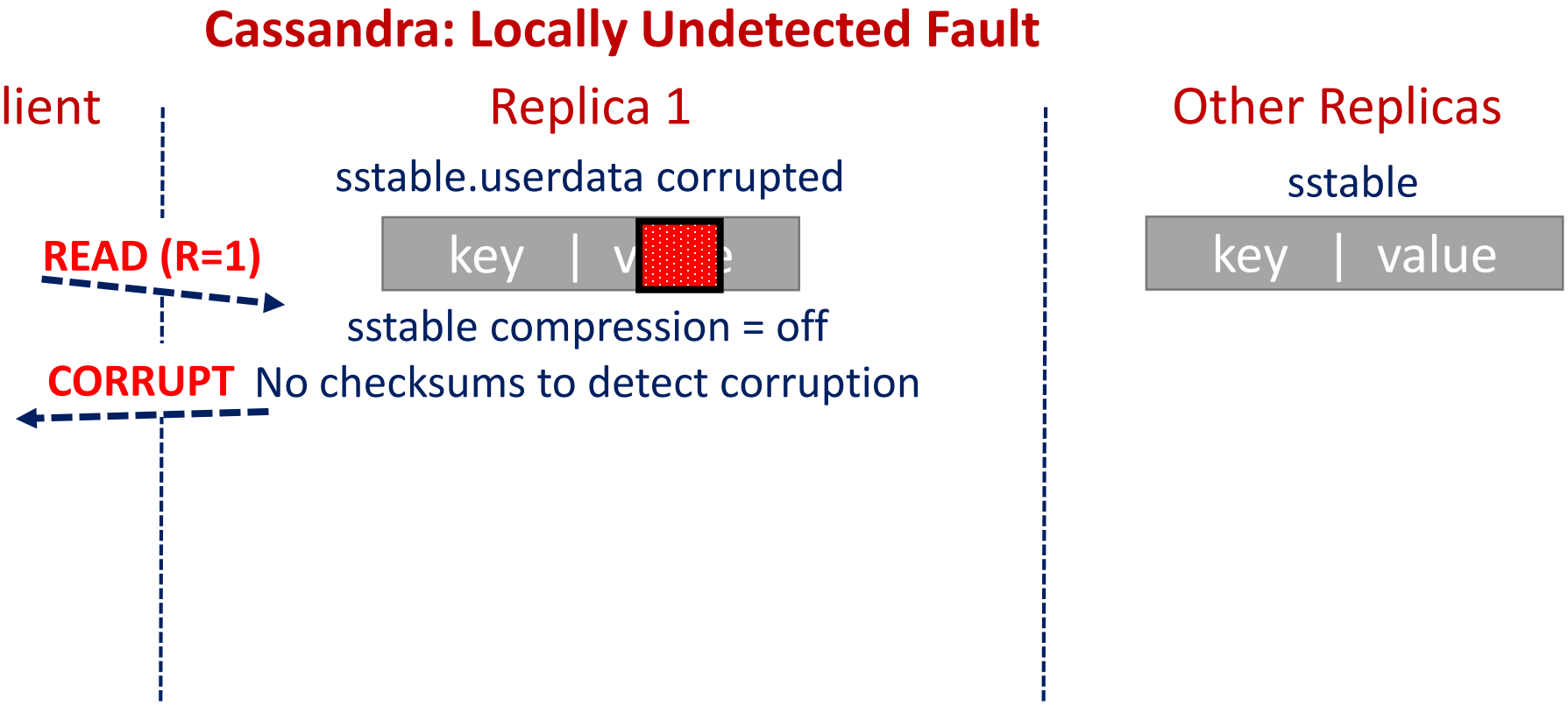
- Locally undetected corruption → global silent corruption



Faults are Often Undetected Locally

Undetected faults may lead to harmful global effect

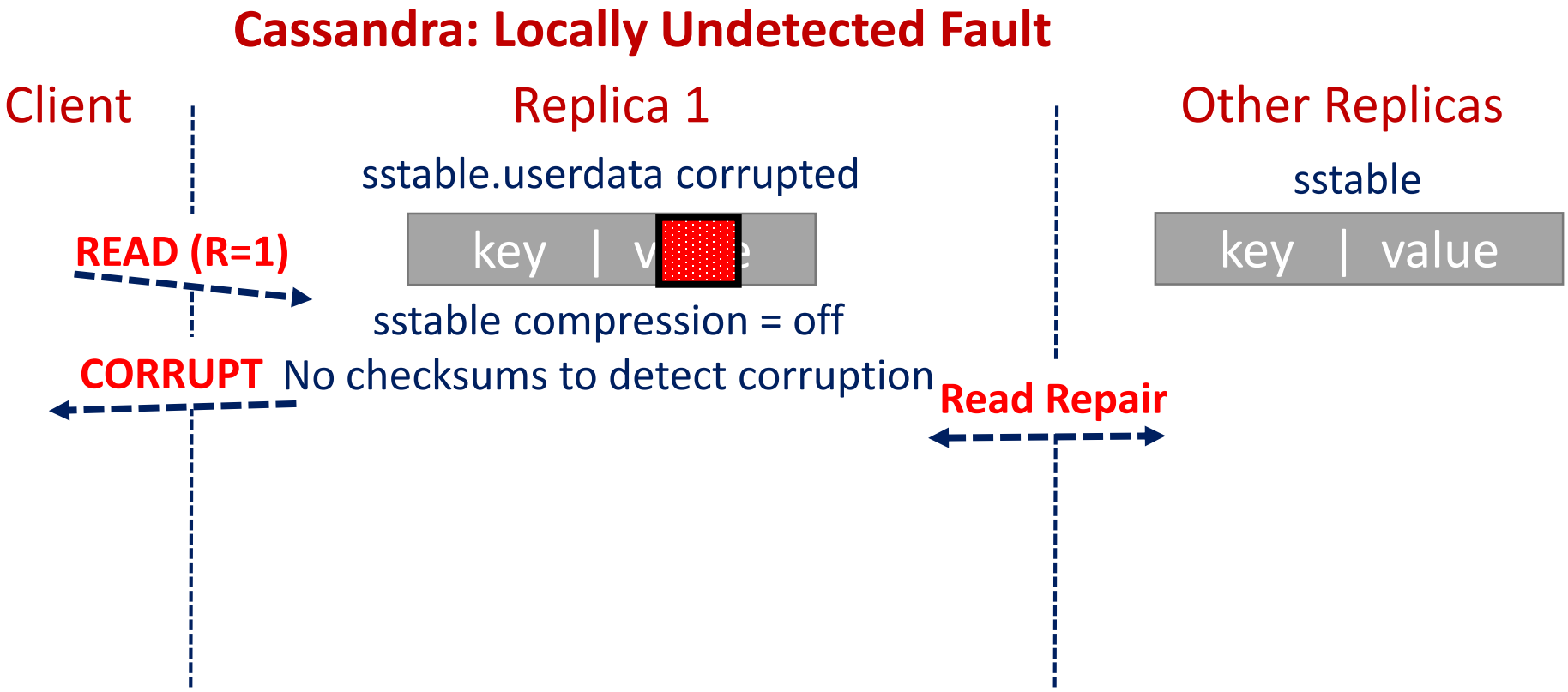
- Locally undetected corruption → global silent corruption



Faults are Often Undetected Locally

Undetected faults may lead to harmful global effect

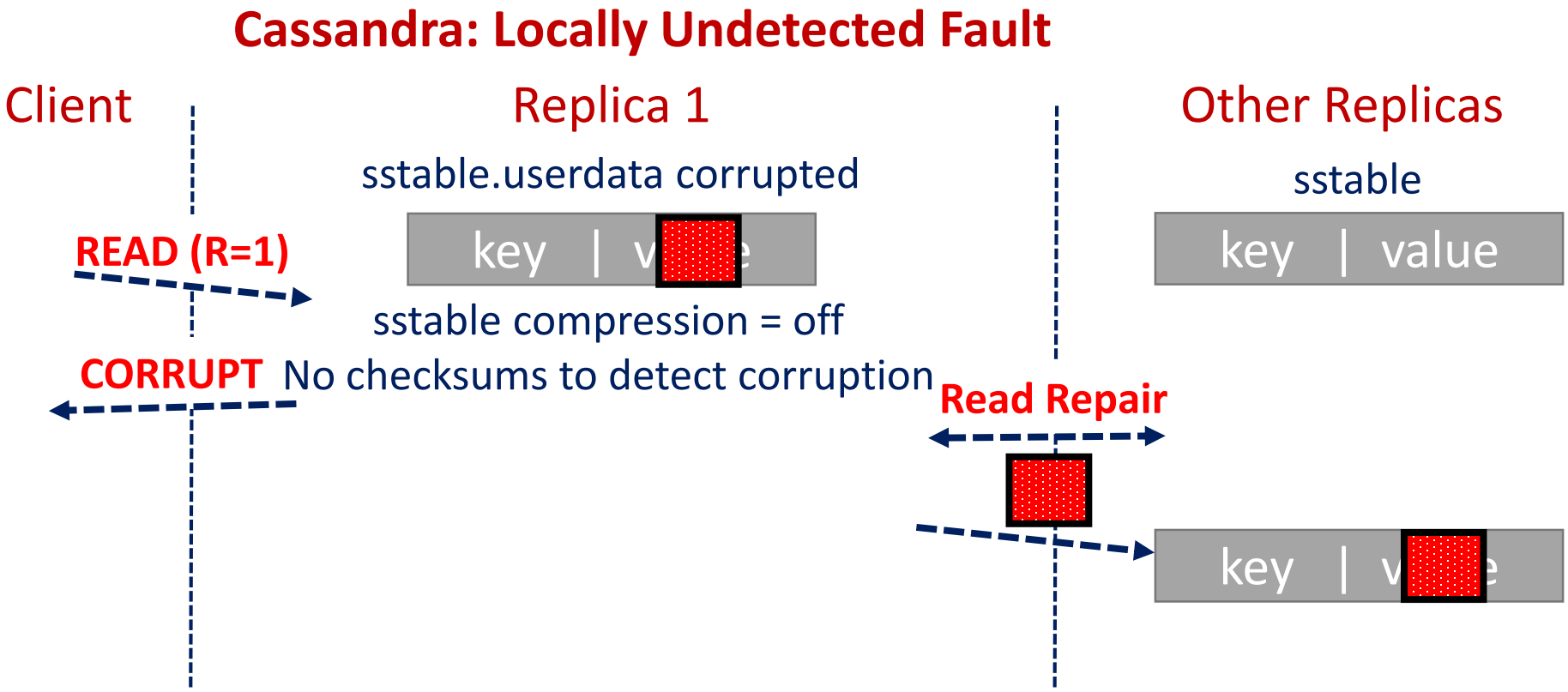
- Locally undetected corruption → global silent corruption



Faults are Often Undetected Locally

Undetected faults may lead to harmful global effect

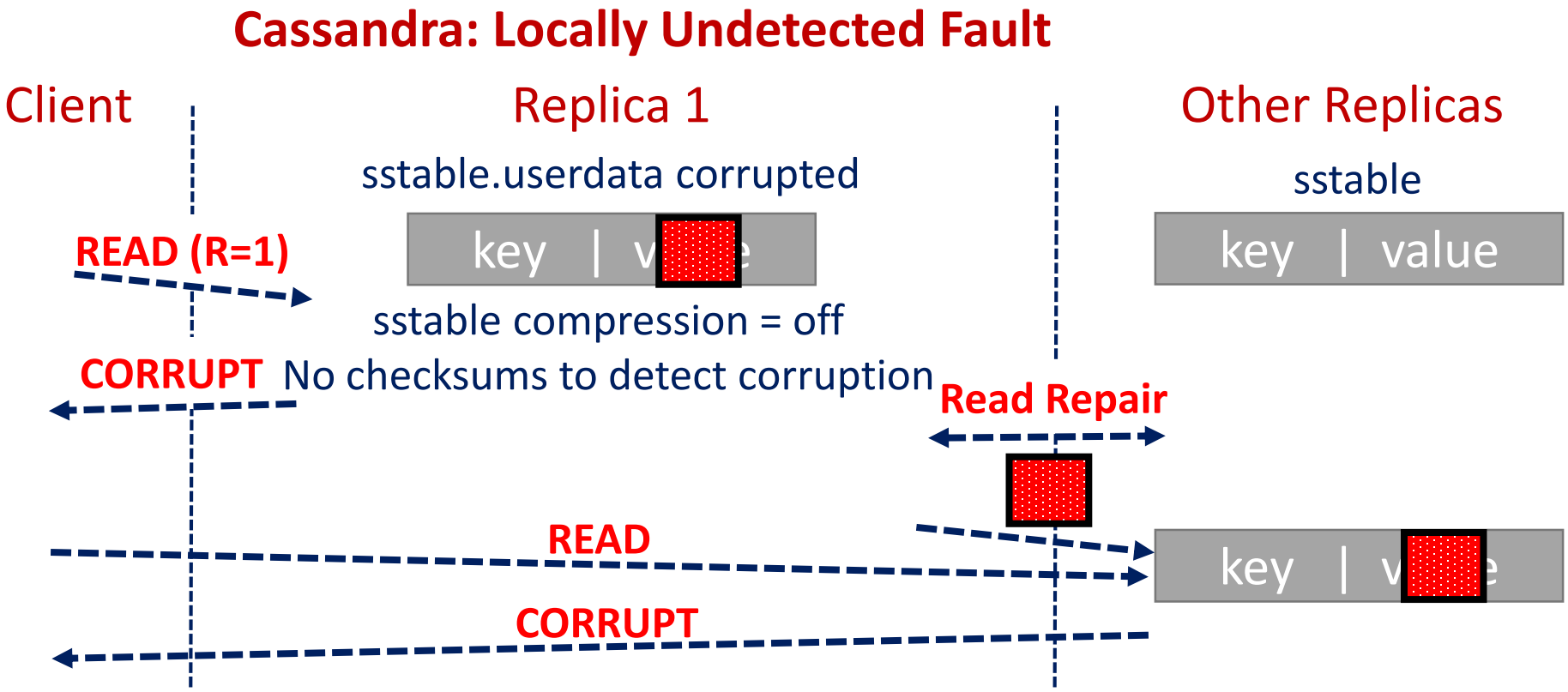
- Locally undetected corruption → global silent corruption



Faults are Often Undetected Locally

Undetected faults may lead to harmful global effect

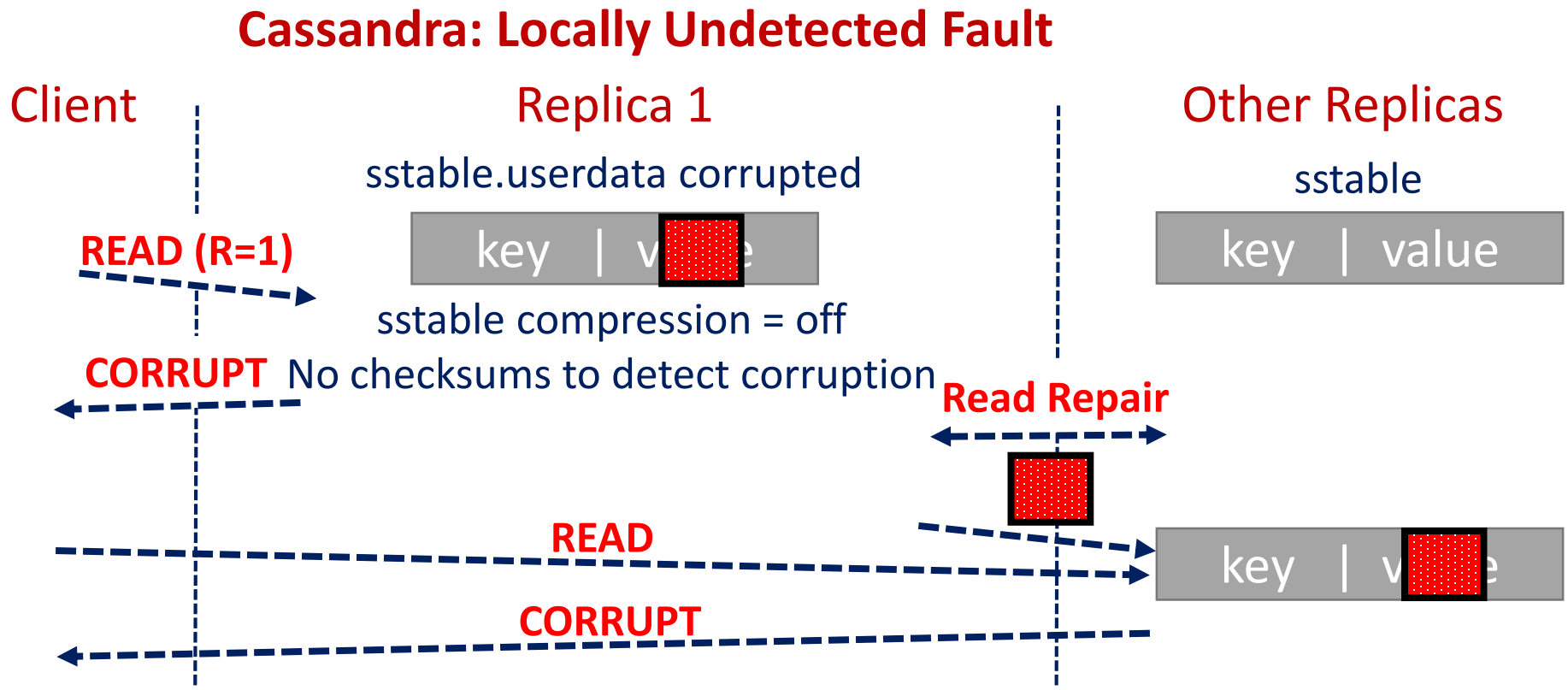
- Locally undetected corruption → global silent corruption



Faults are Often Undetected Locally

Undetected faults may lead to harmful global effect

- Locally undetected corruption → global silent corruption



Need for end-to-end integrity and error handling

Crashing - Common Local Reaction

Crashing - Common Local Reaction

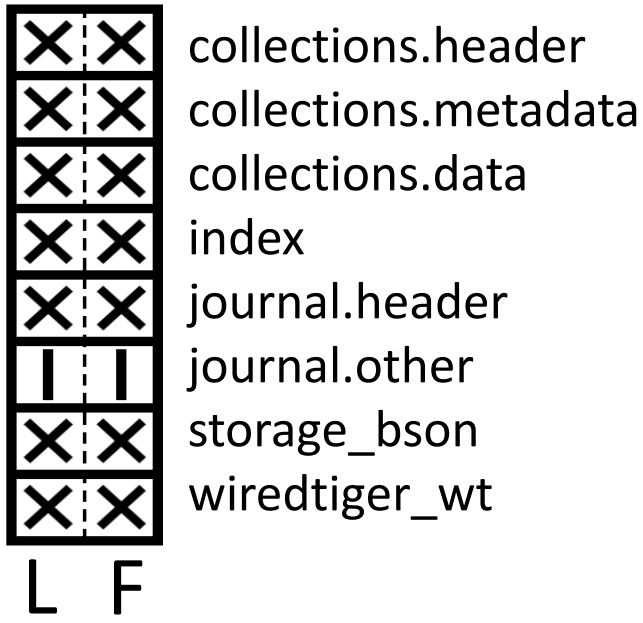
Many systems that reliably detect fault simply crash on encountering faults

Crashing - Common Local Reaction

Many systems that reliably detect fault simply crash on encountering faults

Block Corruption during Read Workloads

MongoDB



☒ Crash

L Leader

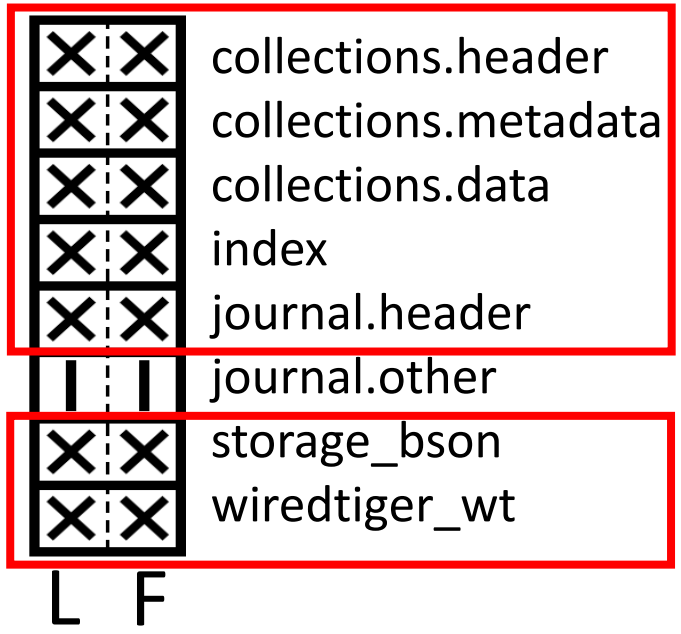
F Follower

Crashing - Common Local Reaction

Many systems that reliably detect fault simply crash on encountering faults

Block Corruption during Read Workloads

MongoDB



Crash

L Leader

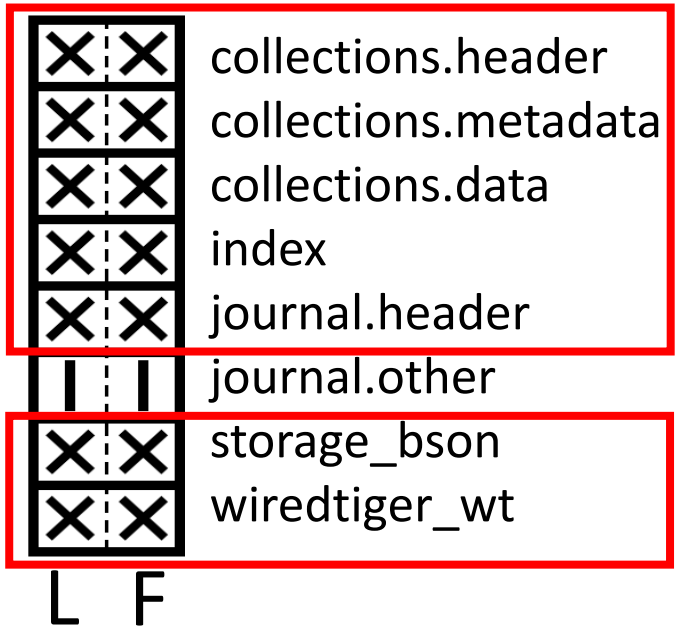
F Follower

Crashing - Common Local Reaction

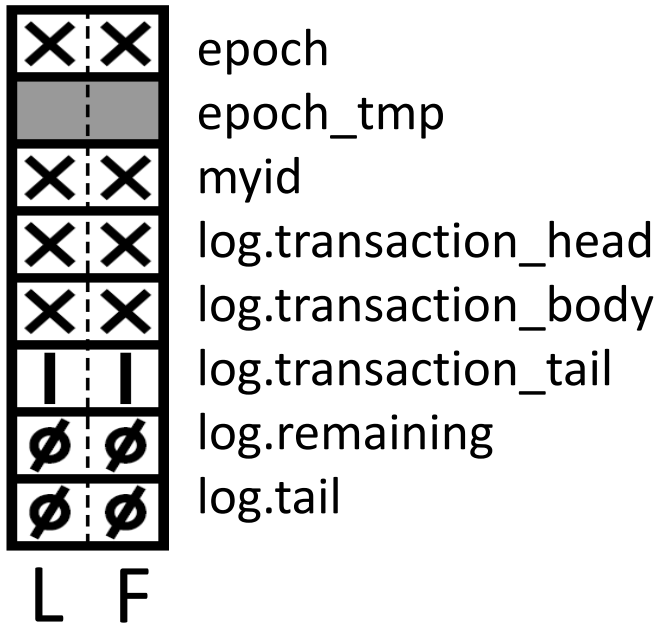
Many systems that reliably detect fault simply crash on encountering faults

Block Corruption during Read Workloads

MongoDB



ZooKeeper



Crash

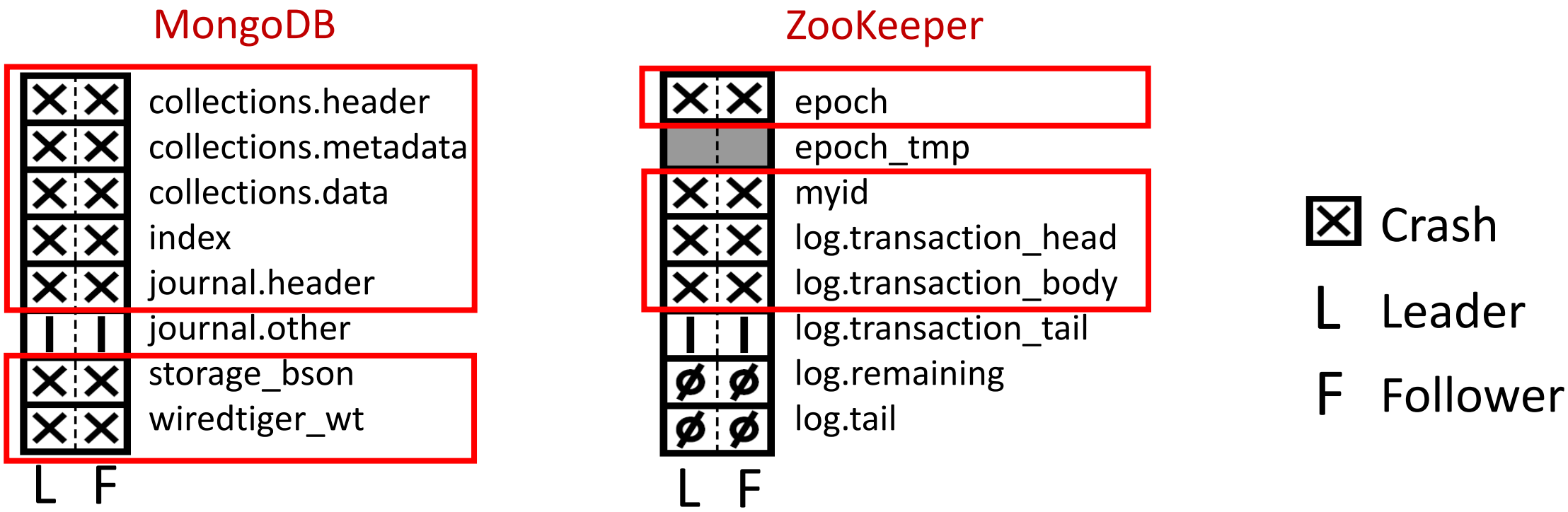
L Leader

F Follower

Crashing - Common Local Reaction

Many systems that reliably detect fault simply crash on encountering faults

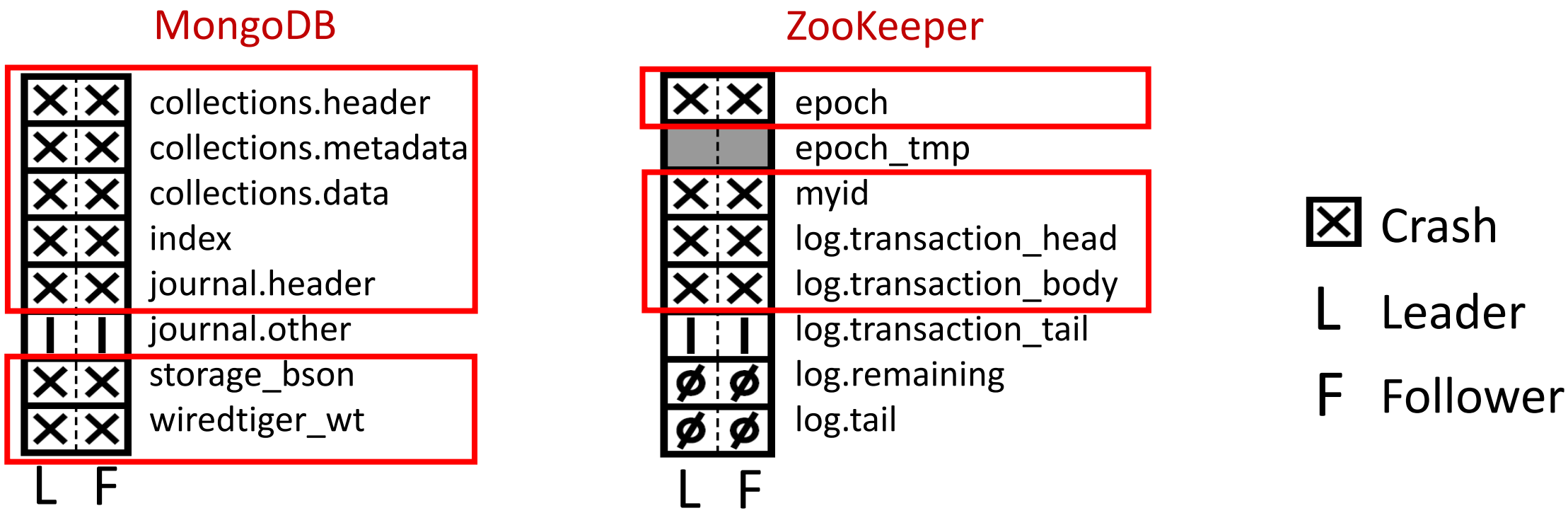
Block Corruption during Read Workloads



Crashing - Common Local Reaction

Many systems that reliably detect fault simply crash on encountering faults

Block Corruption during Read Workloads

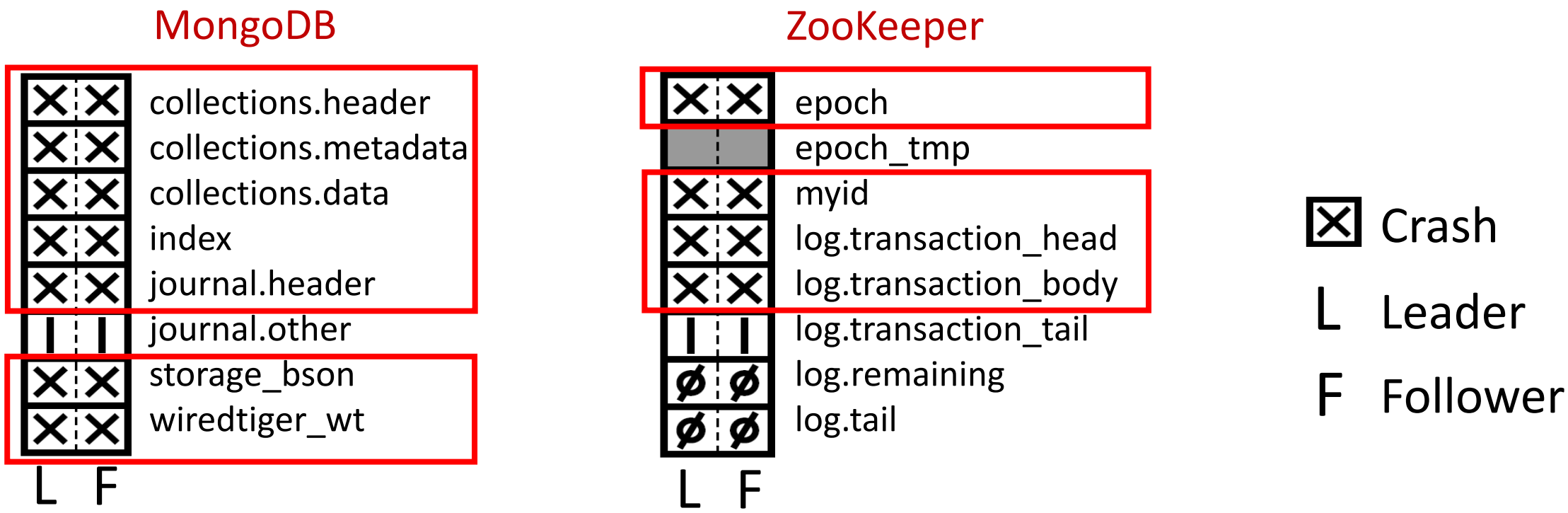


Crashing leads to reduced redundancy and imminent unavailability

Crashing - Common Local Reaction

Many systems that reliably detect fault simply crash on encountering faults

Block Corruption during Read Workloads



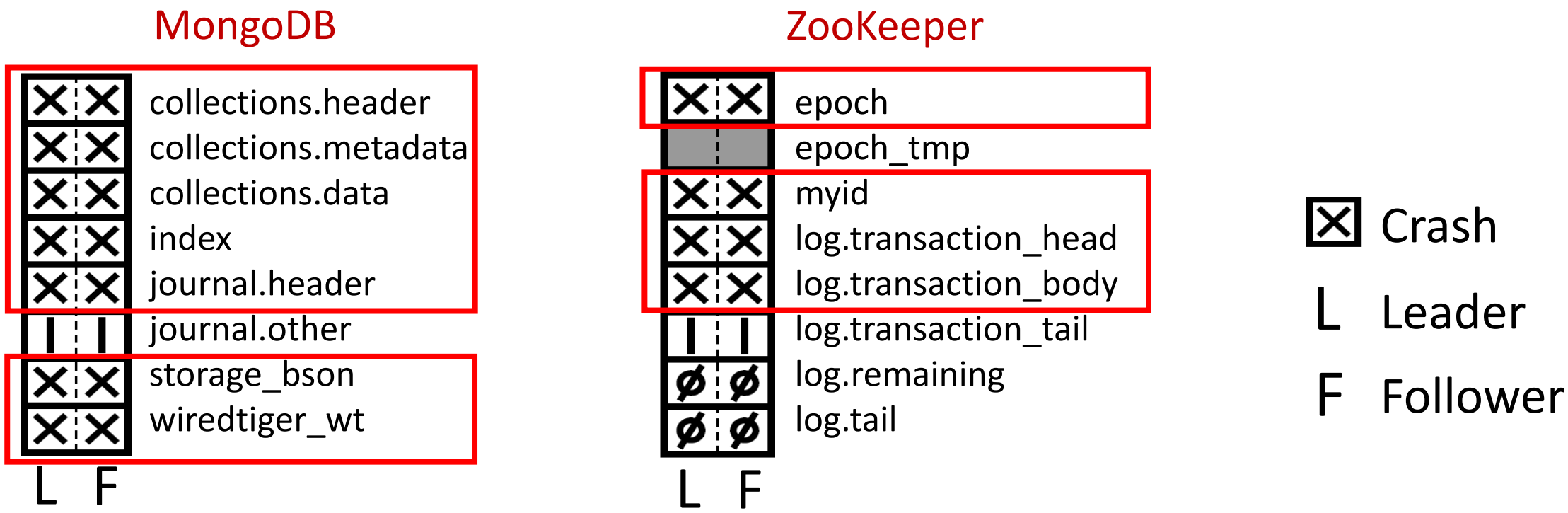
Crashing leads to reduced redundancy and imminent unavailability

Persistent fault -- Requires manual intervention

Crashing - Common Local Reaction

Many systems that reliably detect fault simply crash on encountering faults

Block Corruption during Read Workloads



Crashing leads to reduced redundancy and imminent unavailability

Persistent fault -- Requires manual intervention

Redundancy underutilized!

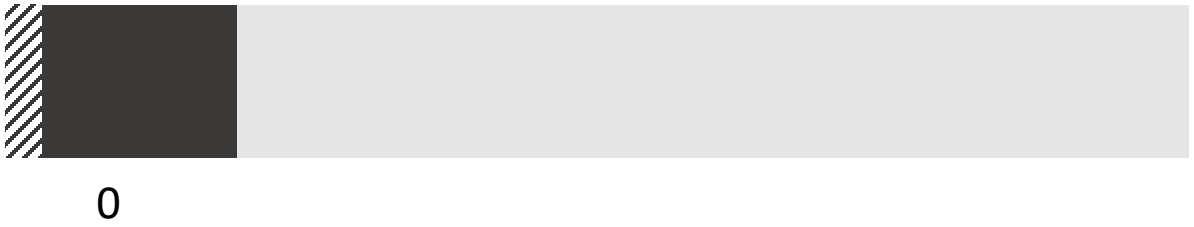
Crash and Corruption Handling are Entangled

Crash and Corruption Handling are Entangled

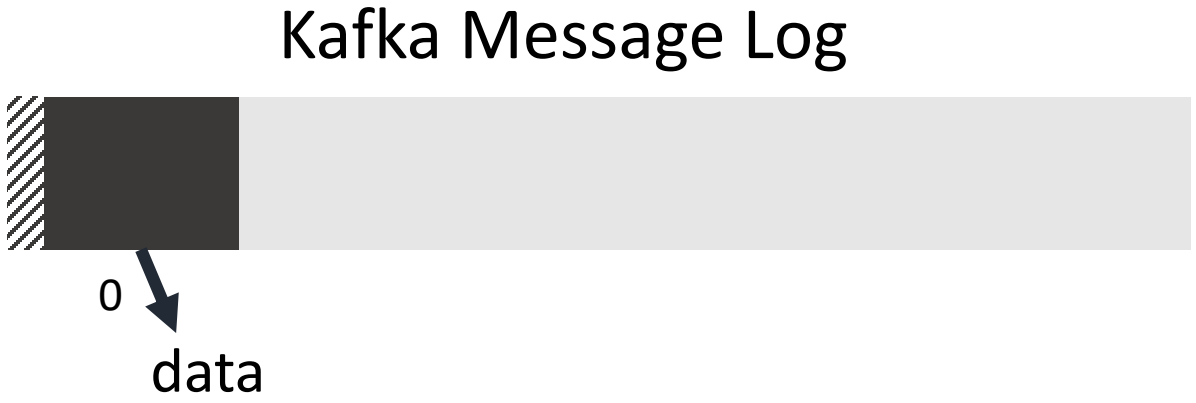
Kafka Message Log

Crash and Corruption Handling are Entangled

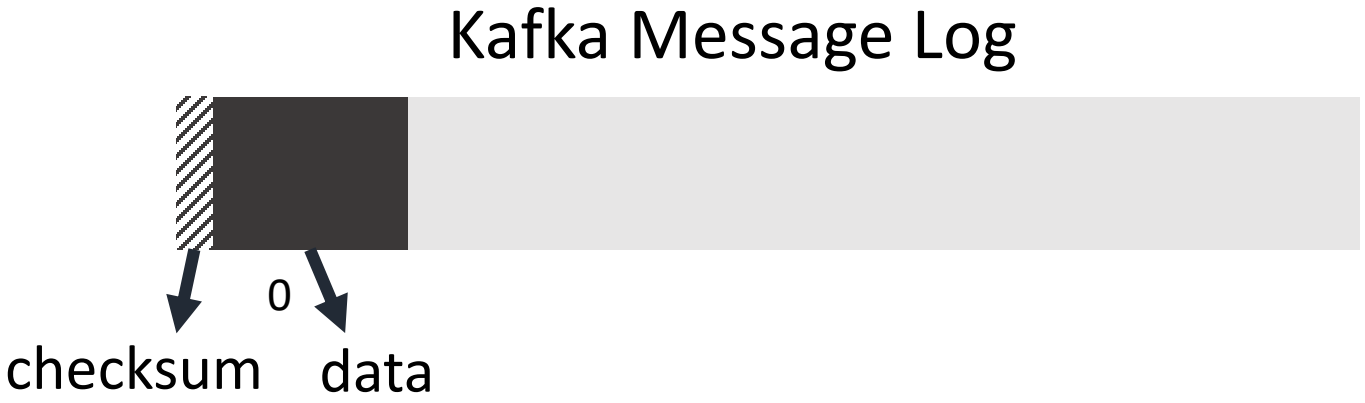
Kafka Message Log



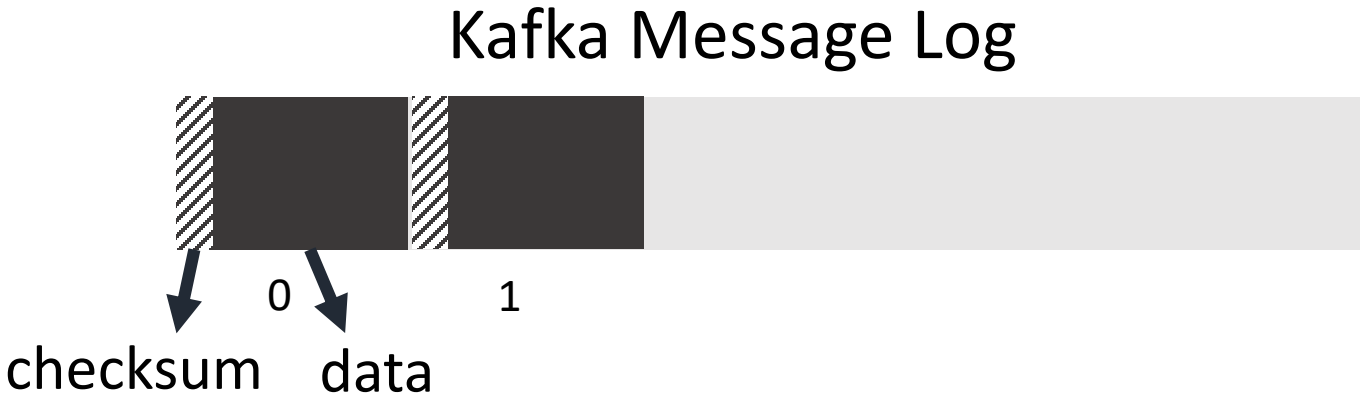
Crash and Corruption Handling are Entangled



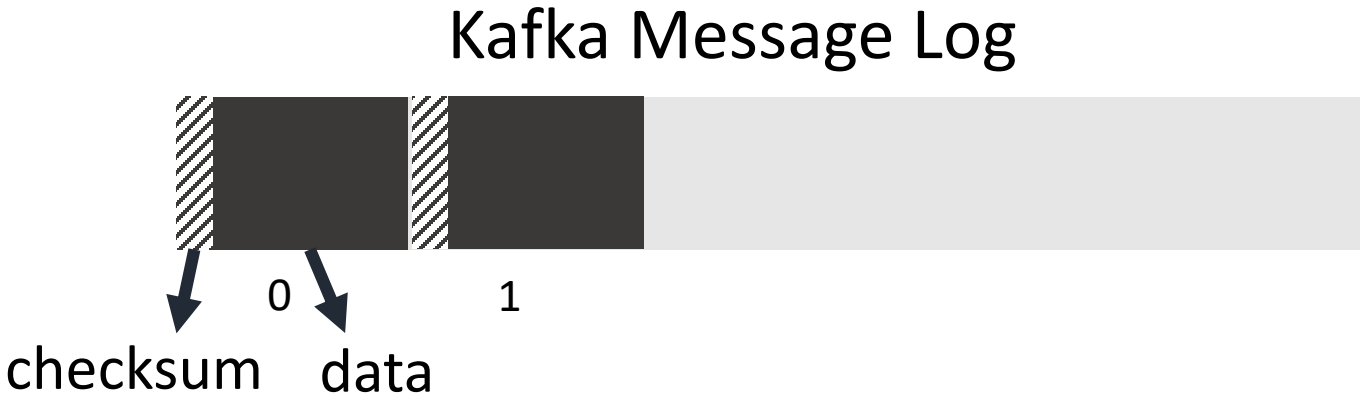
Crash and Corruption Handling are Entangled



Crash and Corruption Handling are Entangled

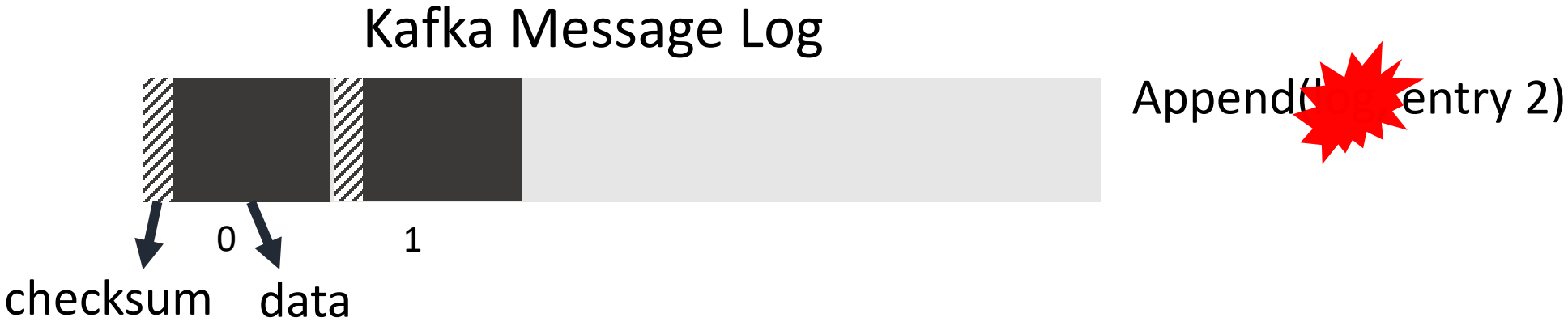


Crash and Corruption Handling are Entangled

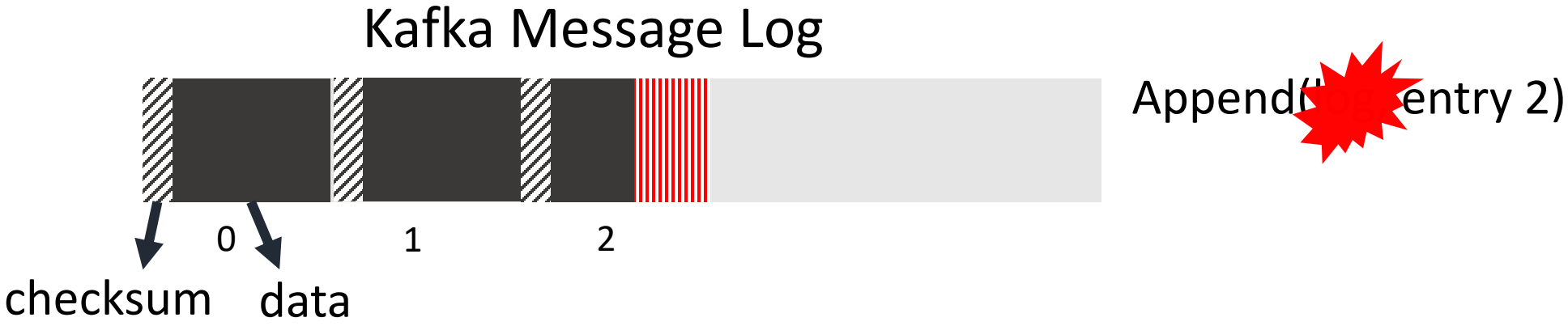


Append(log, entry 2)

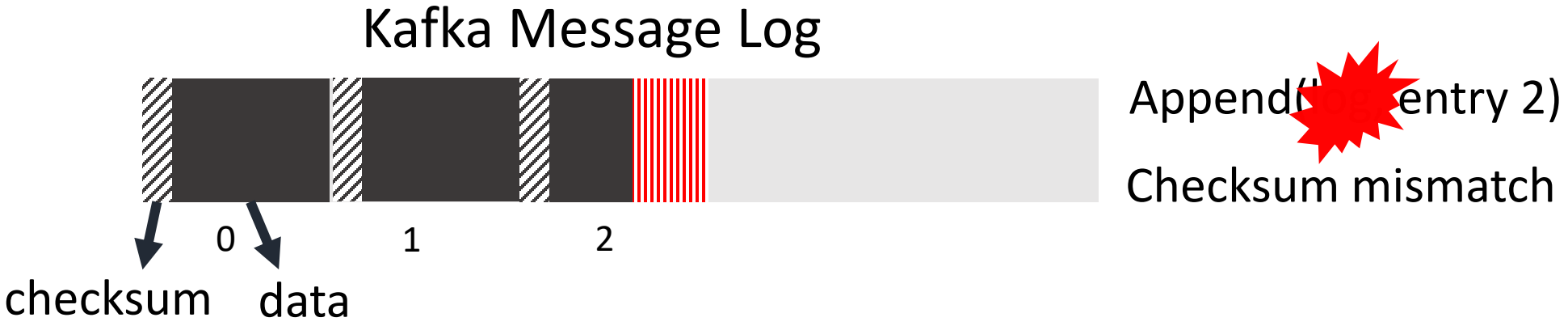
Crash and Corruption Handling are Entangled



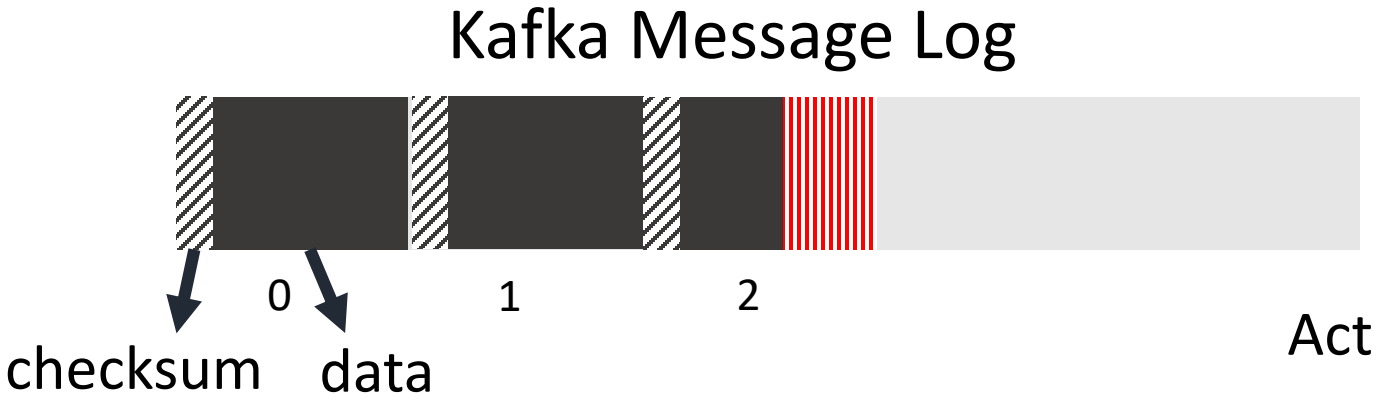
Crash and Corruption Handling are Entangled



Crash and Corruption Handling are Entangled



Crash and Corruption Handling are Entangled

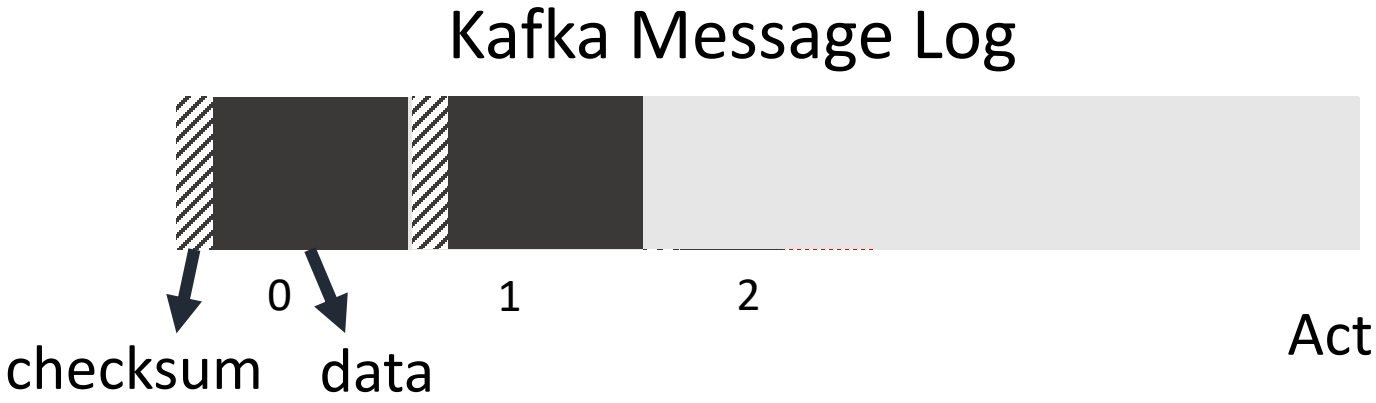


Append([illegible] entry 2)

Checksum mismatch

Action: Truncate log at 1

Crash and Corruption Handling are Entangled

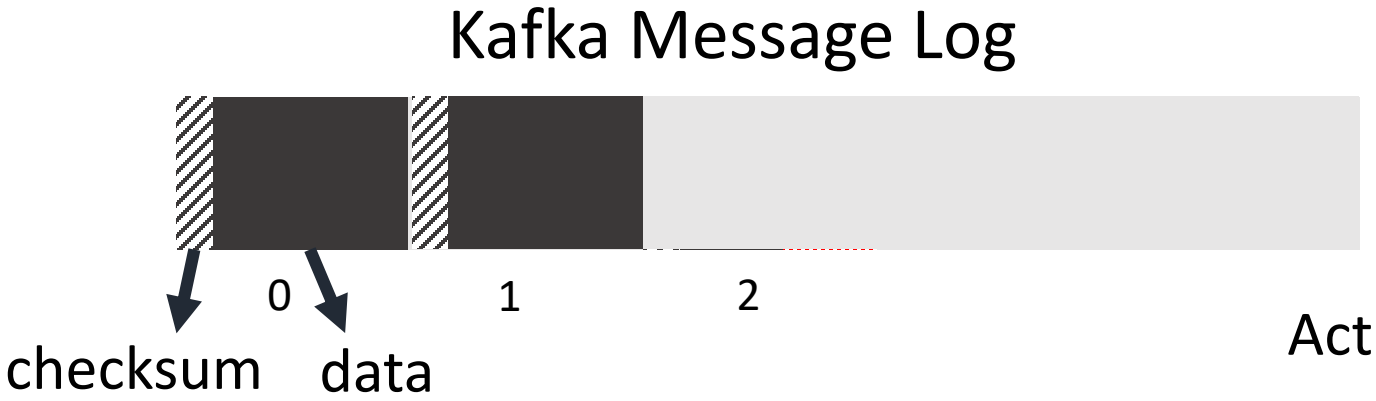


Append([illegible] entry 2)

Checksum mismatch

Action: Truncate log at 1

Crash and Corruption Handling are Entangled



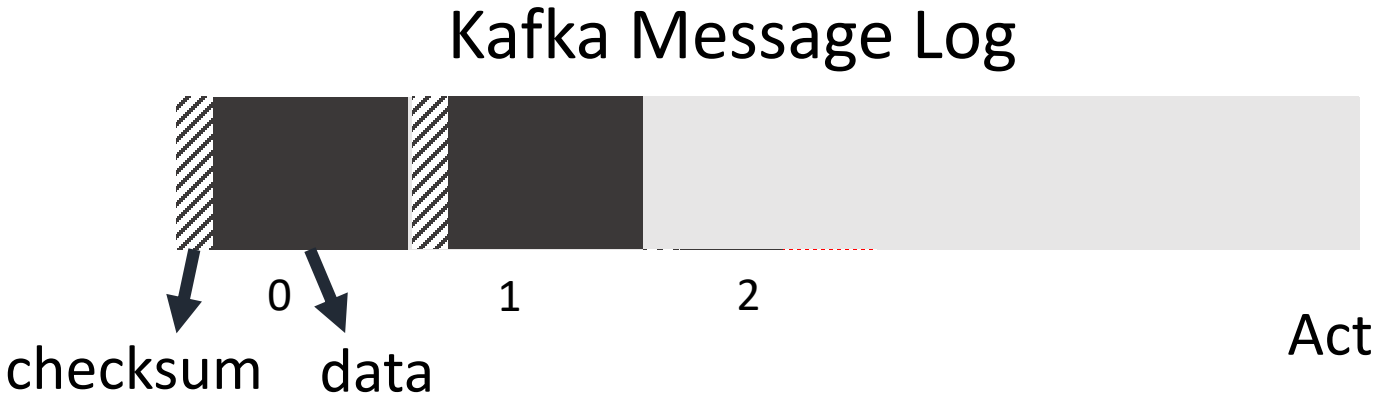
Append([illegible] entry 2)

Checksum mismatch

Action: Truncate log at 1

Lose uncommitted data

Crash and Corruption Handling are Entangled



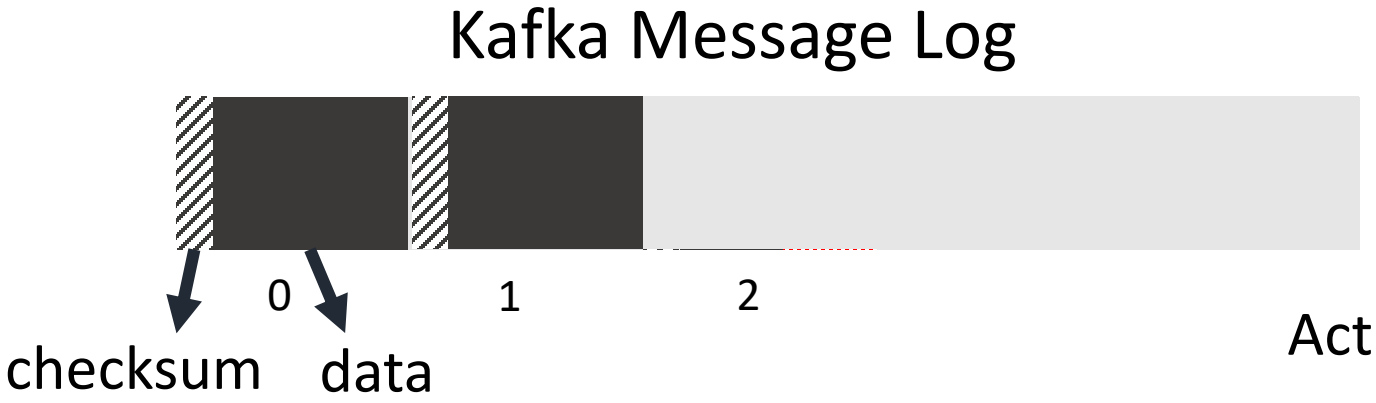
Append(entry 2)

Checksum mismatch

Action: Truncate log at 1

Lose uncommitted data

Crash and Corruption Handling are Entangled

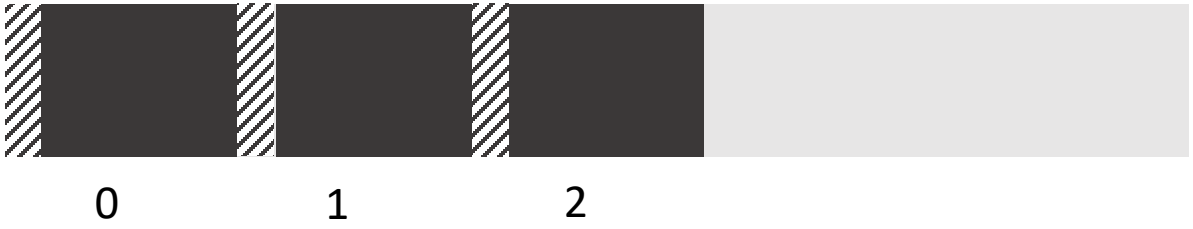


Append([illegible] entry 2)

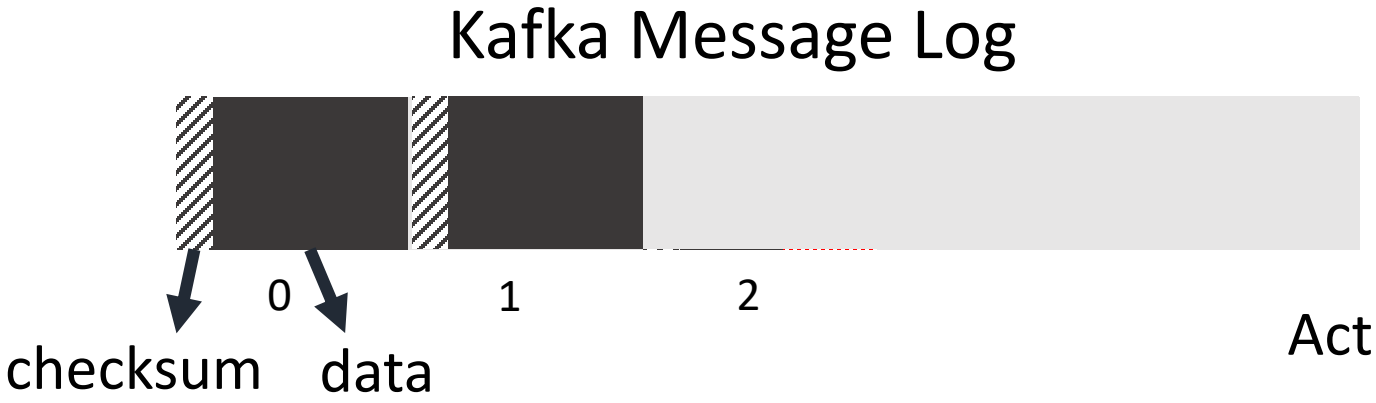
Checksum mismatch

Action: Truncate log at 1

Lose uncommitted data



Crash and Corruption Handling are Entangled

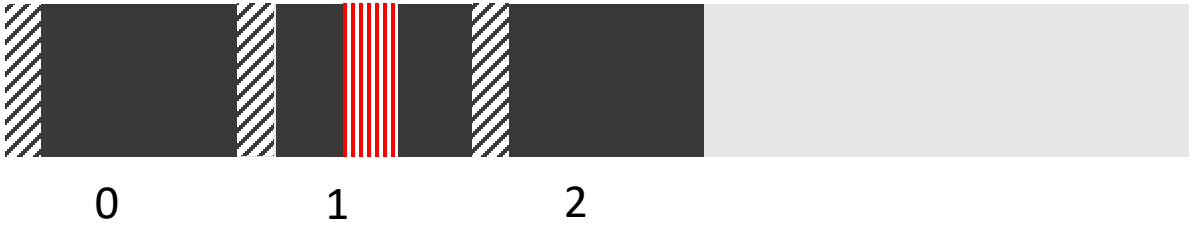


Append(bad entry 2)

Checksum mismatch

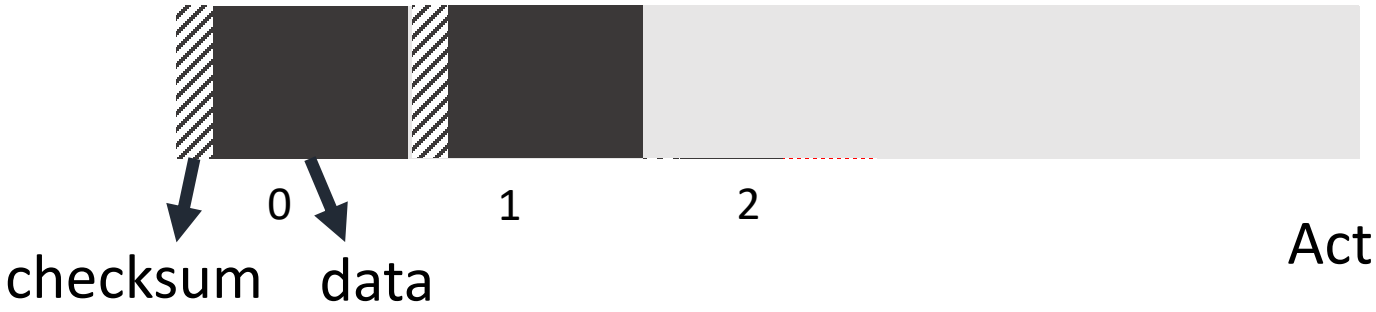
Action: Truncate log at 1

Lose uncommitted data



Crash and Corruption Handling are Entangled

Kafka Message Log

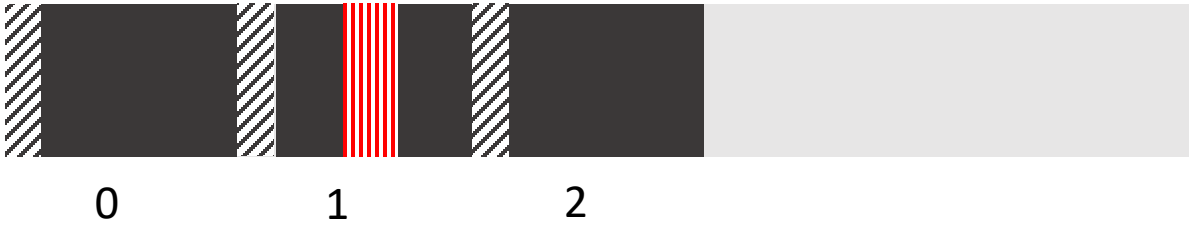


Append(log, entry 2)

Checksum mismatch

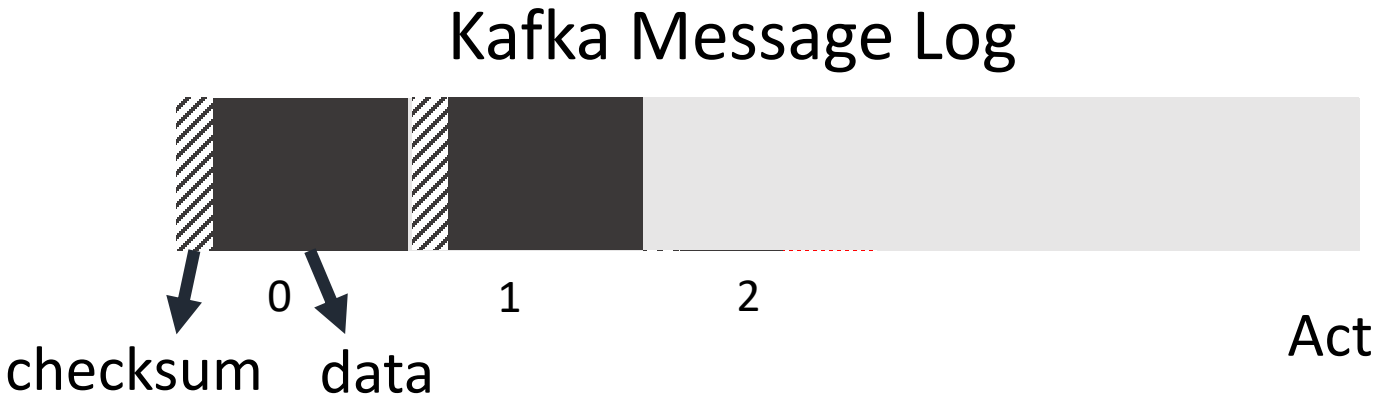
Action: Truncate log at 1

Lose uncommitted data



Disk corruption

Crash and Corruption Handling are Entangled

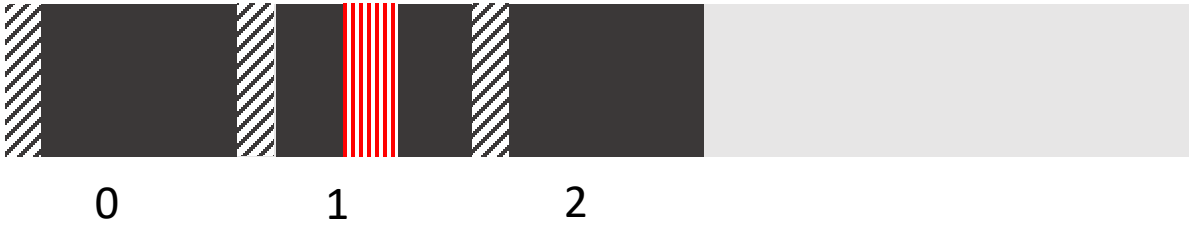


Append(log entry 2)

Checksum mismatch

Action: Truncate log at 1

Lose uncommitted data

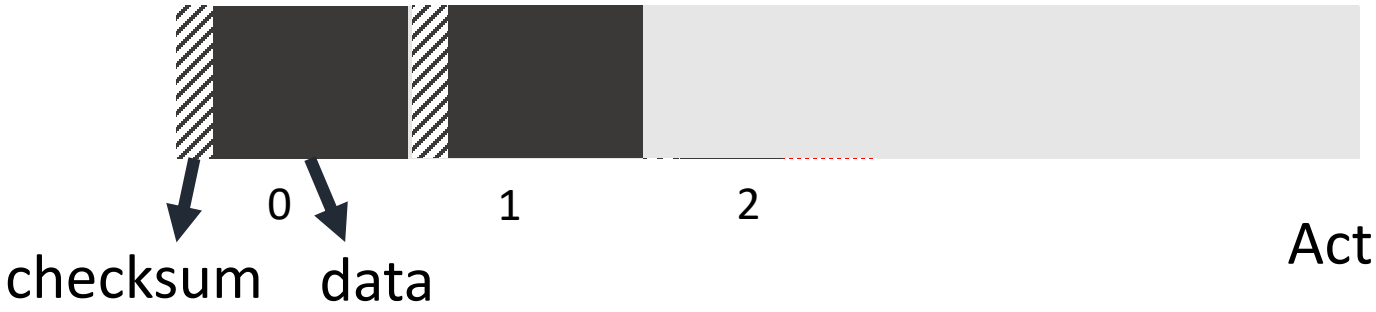


Disk corruption

Checksum mismatch

Crash and Corruption Handling are Entangled

Kafka Message Log

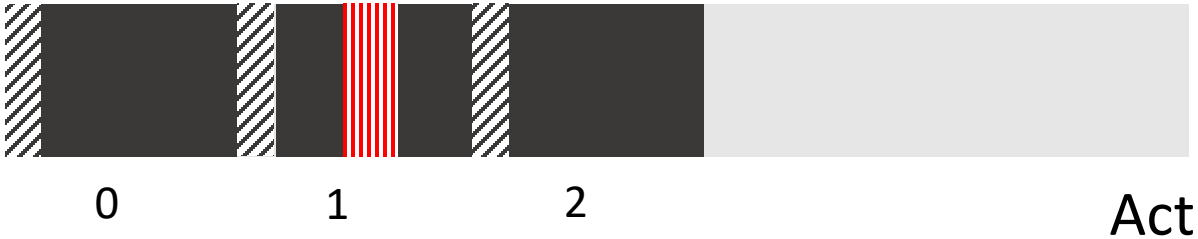


Append(log entry 2)

Checksum mismatch

Action: Truncate log at 1

Lose uncommitted data

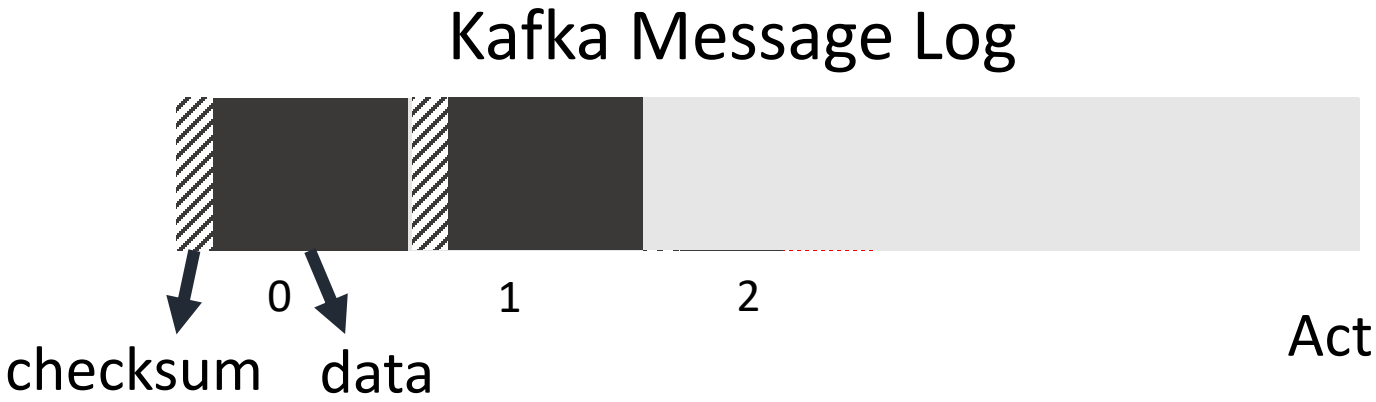


Disk corruption

Checksum mismatch

Action: Truncate log at 0

Crash and Corruption Handling are Entangled



Append(… entry 2)

Checksum mismatch

Action: Truncate log at 1

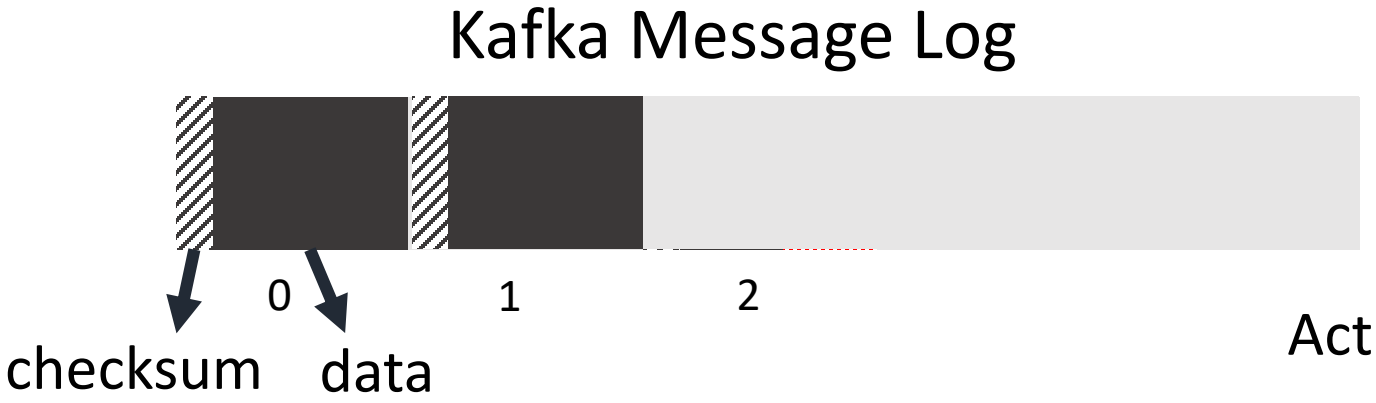
Lose uncommitted data

Disk corruption

Checksum mismatch

Action: Truncate log at 0

Crash and Corruption Handling are Entangled



Append(entry 2)

Checksum mismatch

Action: Truncate log at 1

Lose uncommitted data

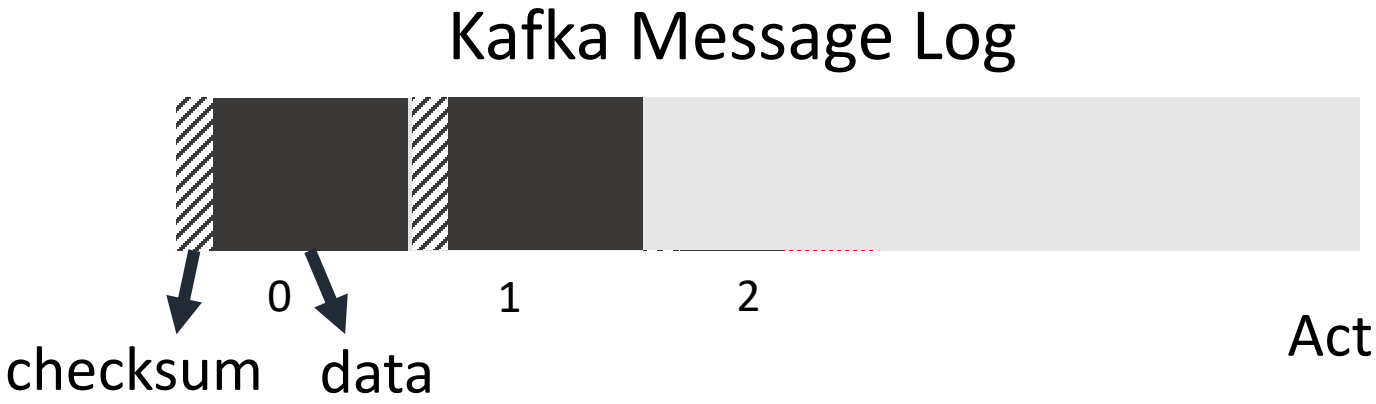
Disk corruption

Checksum mismatch

Action: Truncate log at 0

Lose committed data!

Crash and Corruption Handling are Entangled



Append([illegible] entry 2)

Checksum mismatch

Action: Truncate log at 1

Lose uncommitted data

Disk corruption

Checksum mismatch

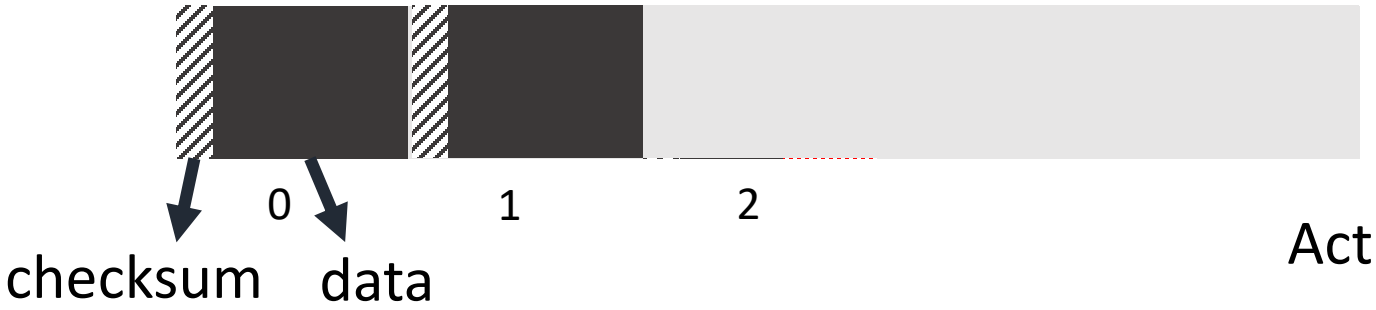
Action: Truncate log at 0

Lose committed data!

Developers of LogCabin and RethinkDB agree entanglement is the problem

Crash and Corruption Handling are Entangled

Kafka Message Log



Append(entry 2)

Checksum mismatch

Action: Truncate log at 1

Lose uncommitted data

Disk corruption

Checksum mismatch

Action: Truncate log at 0

Lose committed data!

Developers of LogCabin and RethinkDB agree entanglement is the problem

Need for discerning corruptions due to crashes from other type of corruptions

Unsafe Interaction between Local & Global Protocols

Unsafe Interaction between Local & Global Protocols

Kafka: Message log at Node 1

Unsafe Interaction between Local & Global Protocols

Kafka: Message log at Node 1

Disk corruption
Checksum mismatch

Unsafe Interaction between Local & Global Protocols

Kafka: Message log at Node 1

Disk corruption
Checksum mismatch
Action: Truncate log at 0

Unsafe Interaction between Local & Global Protocols

Kafka: Message log at Node 1

Disk corruption

Checksum mismatch

Action: Truncate log at 0

Lose committed data!

Unsafe Interaction between Local & Global Protocols

Kafka: Message log at Node 1

Local Behavior

Disk corruption

Checksum mismatch

Action: Truncate log at 0

Lose committed data!

Unsafe Interaction between Local & Global Protocols

Kafka: Message log at Node 1

Local Behavior

Disk corruption

Checksum mismatch

Action: Truncate log at 0

Lose committed data!

Unsafe Interaction between Local & Global Protocols

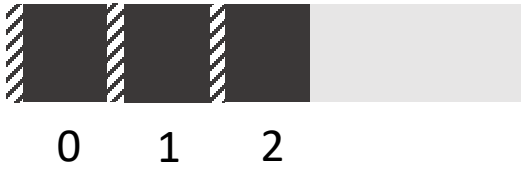
Kafka: Message log at Node 1

Local Behavior

Disk corruption
Checksum mismatch
Action: Truncate log at 0
Lose committed data!

Node1

Other Nodes



Unsafe Interaction between Local & Global Protocols

Kafka: Message log at Node 1

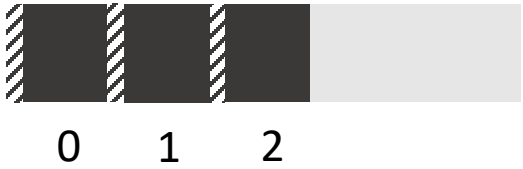
Local Behavior

Disk corruption
Checksum mismatch
Action: Truncate log at 0
Lose committed data!

Set of in-sync replicas

Node1

Other Nodes



Unsafe Interaction between Local & Global Protocols

Kafka: Message log at Node 1

Local Behavior

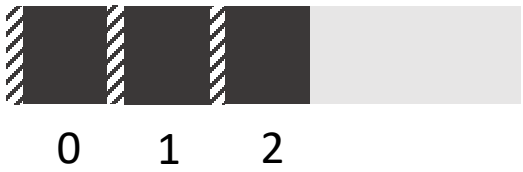
Disk corruption
Checksum mismatch
Action: Truncate log at 0
Lose committed data!

Set of in-sync replicas

Node1 with truncated log not removed from in-sync replicas

Node1

Other Nodes



Unsafe Interaction between Local & Global Protocols

Kafka: Message log at Node 1

Local Behavior

Disk corruption
Checksum mismatch
Action: Truncate log at 0
Lose committed data!

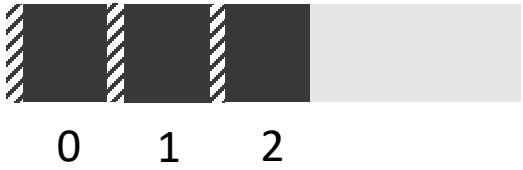
Set of in-sync replicas

Node1 with truncated log not removed from in-sync replicas

Node 1 elected as leader

Leader

Followers



Unsafe Interaction between Local & Global Protocols

Kafka: Message log at Node 1

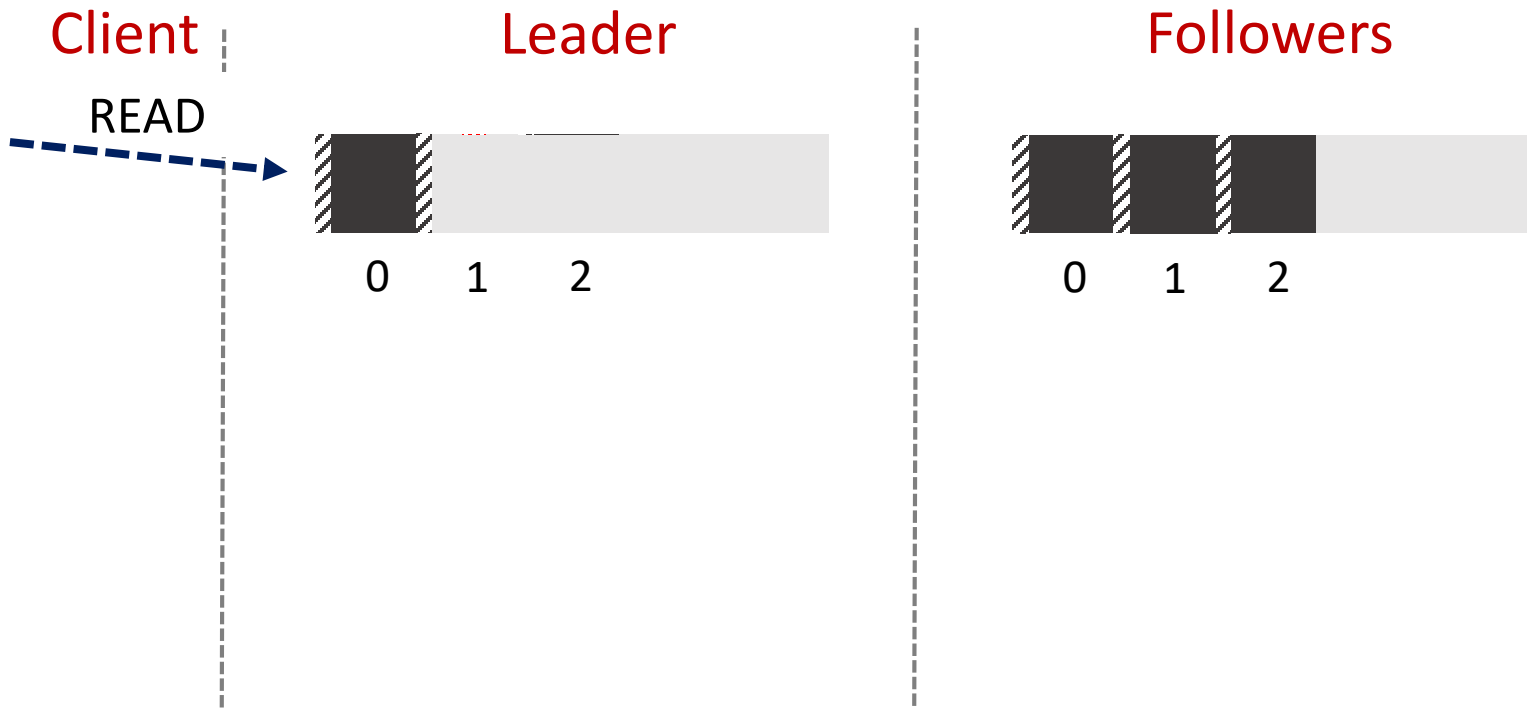
Local Behavior

Disk corruption
Checksum mismatch
Action: Truncate log at 0
Lose committed data!

Set of in-sync replicas

Node1 with truncated log not removed from in-sync replicas

Node 1 elected as leader



Unsafe Interaction between Local & Global Protocols

Kafka: Message log at Node 1

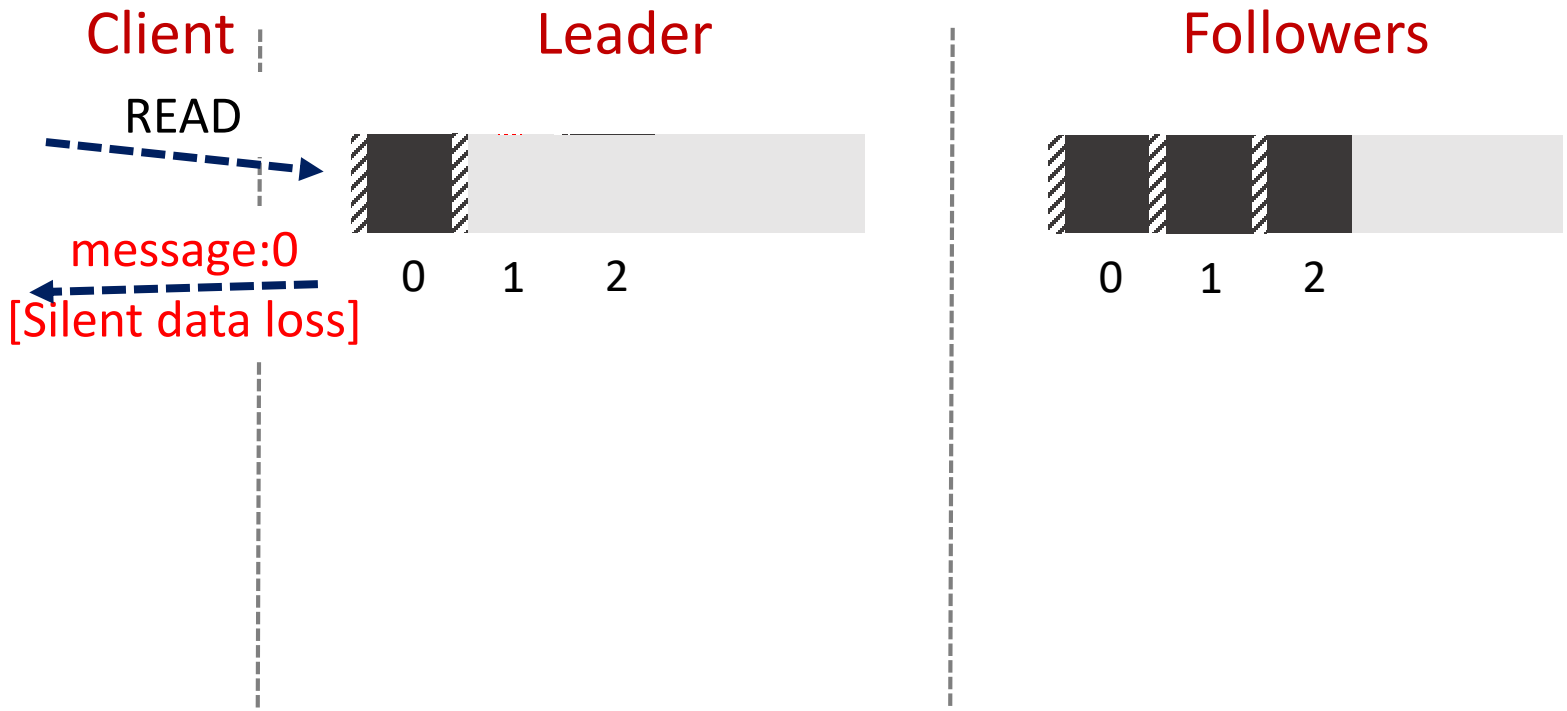
Local Behavior

Disk corruption
Checksum mismatch
Action: Truncate log at 0
Lose committed data!

Set of in-sync replicas

Node1 with truncated log not removed from in-sync replicas

Node 1 elected as leader



Unsafe Interaction between Local & Global Protocols

Kafka: Message log at Node 1

Local Behavior

Disk corruption

Checksum mismatch

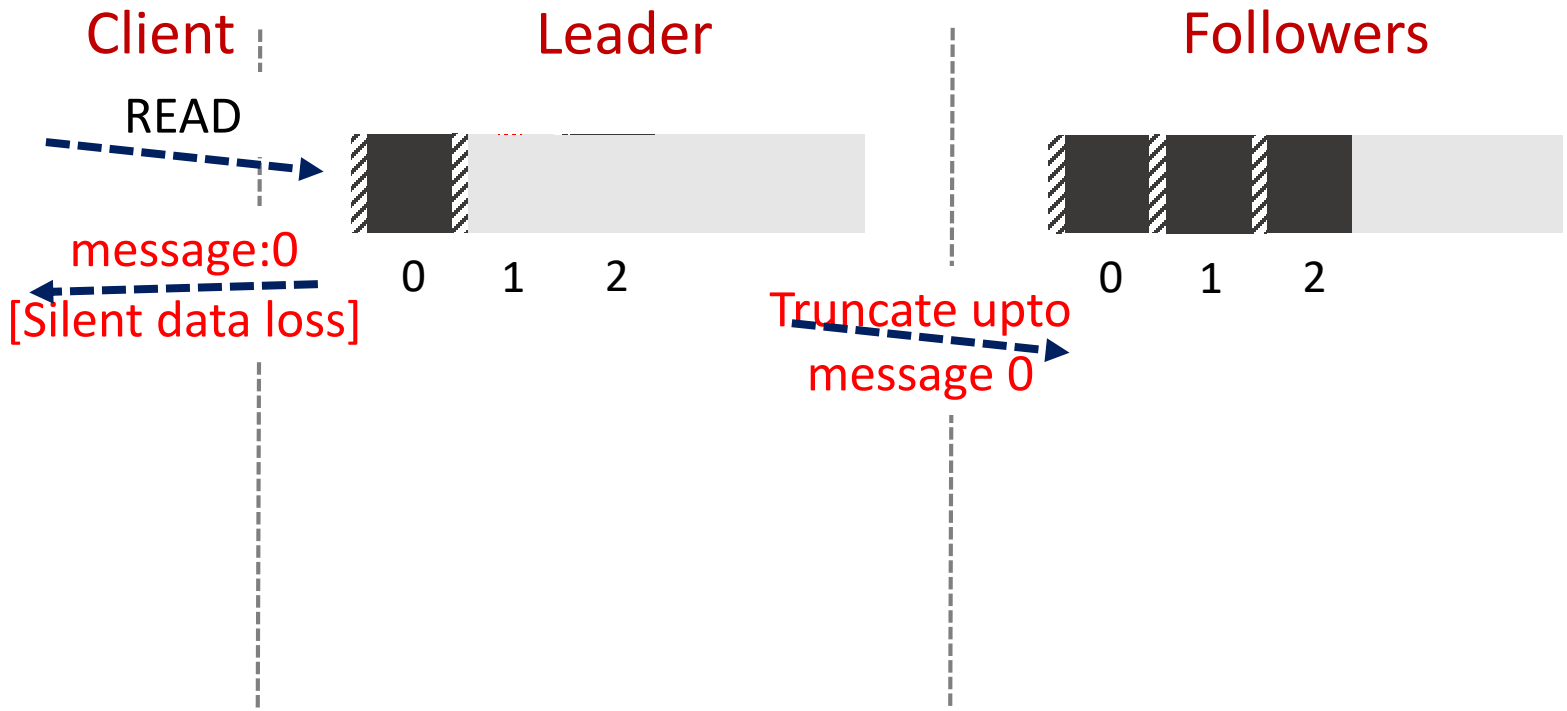
Action: Truncate log at 0

Lose committed data!

Set of in-sync replicas

Node1 with truncated log not removed from in-sync replicas

Node 1 elected as leader



Unsafe Interaction between Local & Global Protocols

Kafka: Message log at Node 1

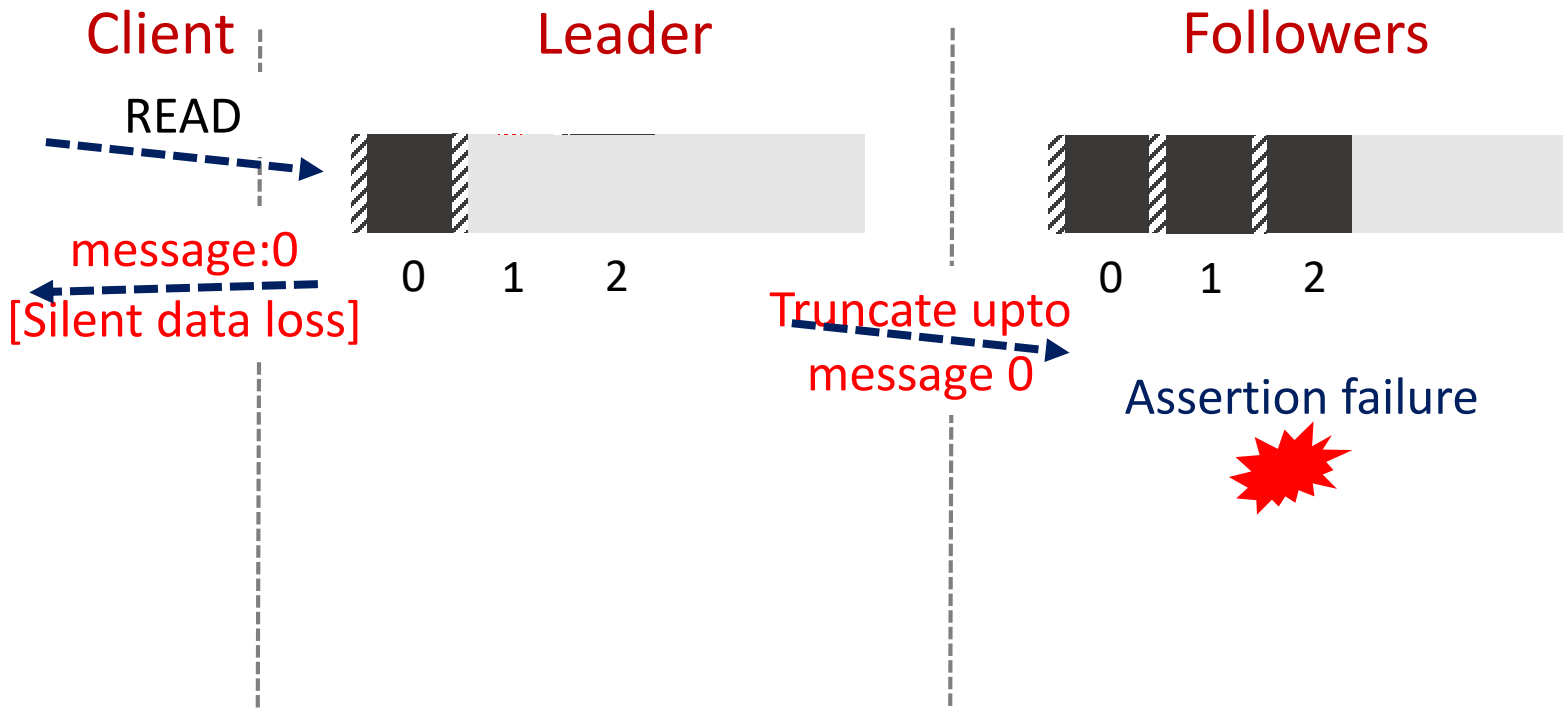
Local Behavior

Disk corruption
Checksum mismatch
Action: Truncate log at 0
Lose committed data!

Set of in-sync replicas

Node1 with truncated log not removed from in-sync replicas

Node 1 elected as leader



Unsafe Interaction between Local & Global Protocols

Kafka: Message log at Node 1

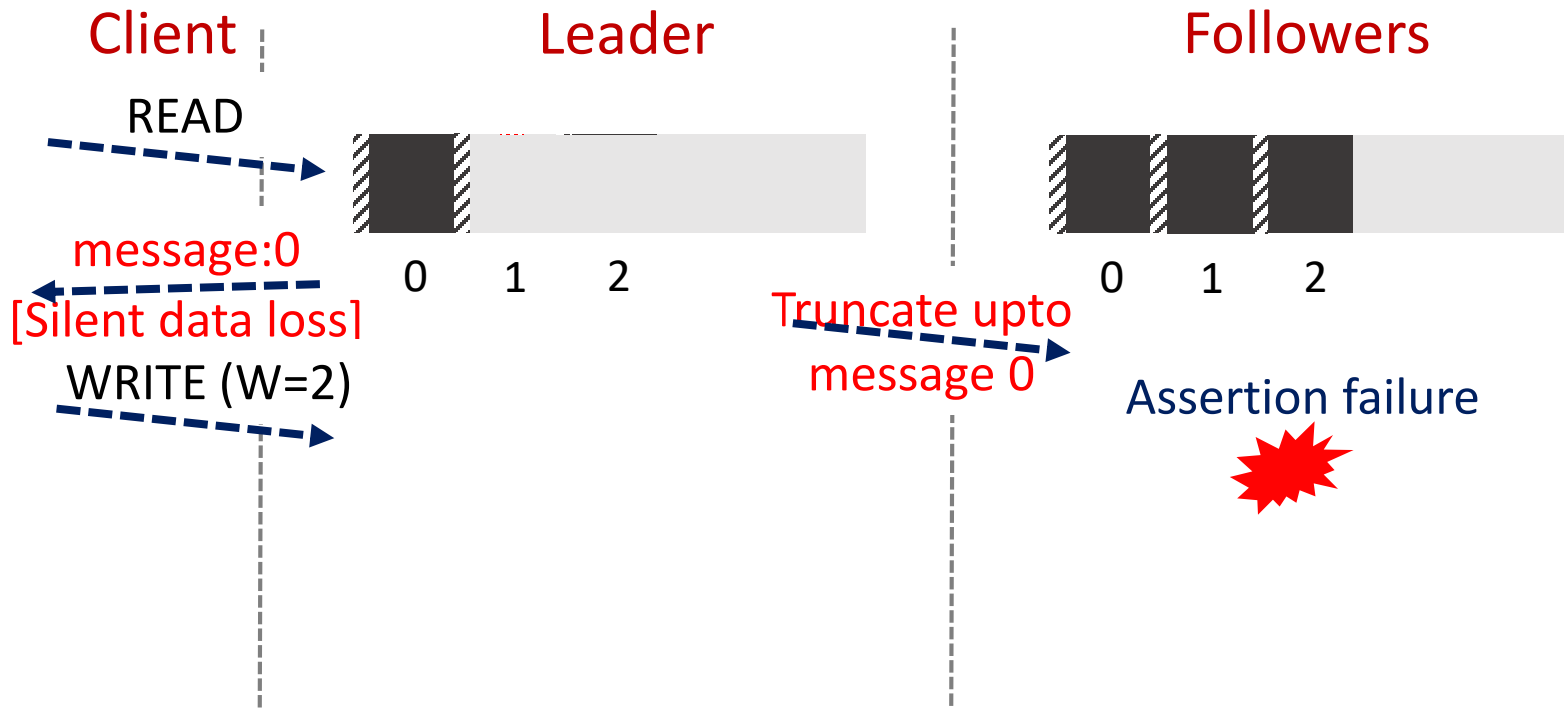
Local Behavior

Disk corruption
Checksum mismatch
Action: Truncate log at 0
Lose committed data!

Set of in-sync replicas

Node1 with truncated log not removed from in-sync replicas

Node 1 elected as leader



Unsafe Interaction between Local & Global Protocols

Kafka: Message log at Node 1

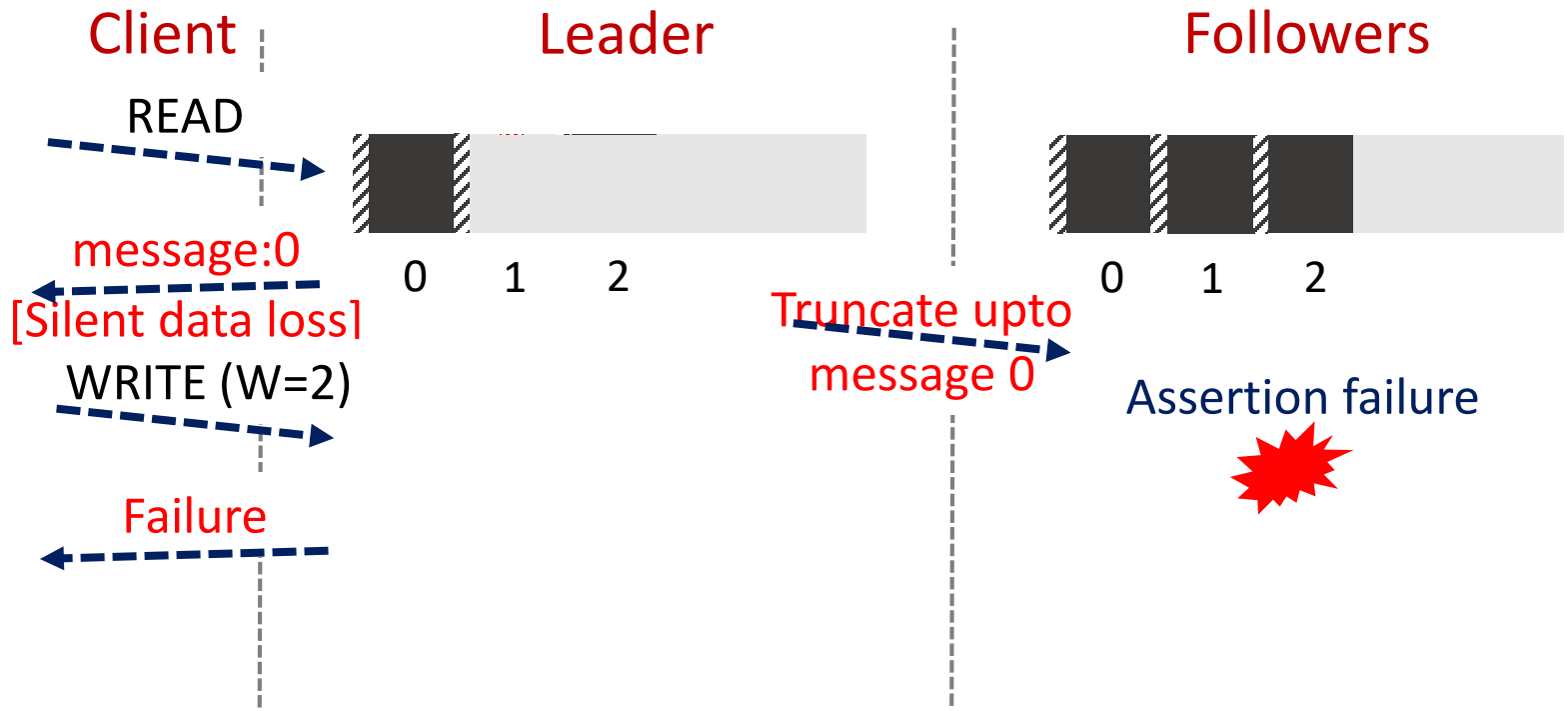
Local Behavior

Disk corruption
Checksum mismatch
Action: Truncate log at 0
Lose committed data!

Set of in-sync replicas

Node1 with truncated log not removed from in-sync replicas

Node 1 elected as leader



Unsafe Interaction between Local & Global Protocols

Kafka: Message log at Node 1

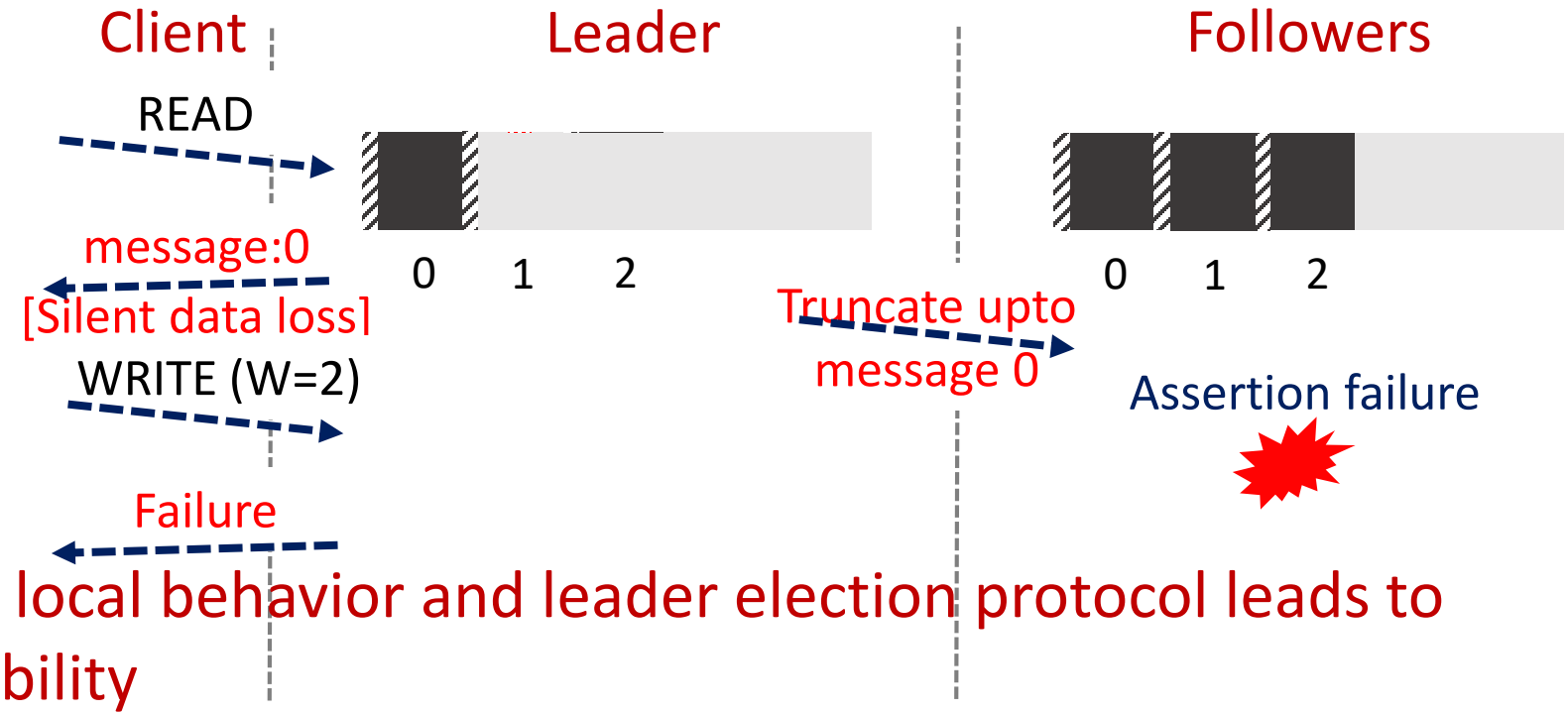
Local Behavior

Disk corruption
Checksum mismatch
Action: Truncate log at 0
Lose committed data!

Set of in-sync replicas

Node1 with truncated log not removed from in-sync replicas

Node 1 elected as leader



Unsafe interaction between local behavior and leader election protocol leads to data loss and write unavailability

Unsafe Interaction between Local & Global Protocols

Kafka: Message log at Node 1

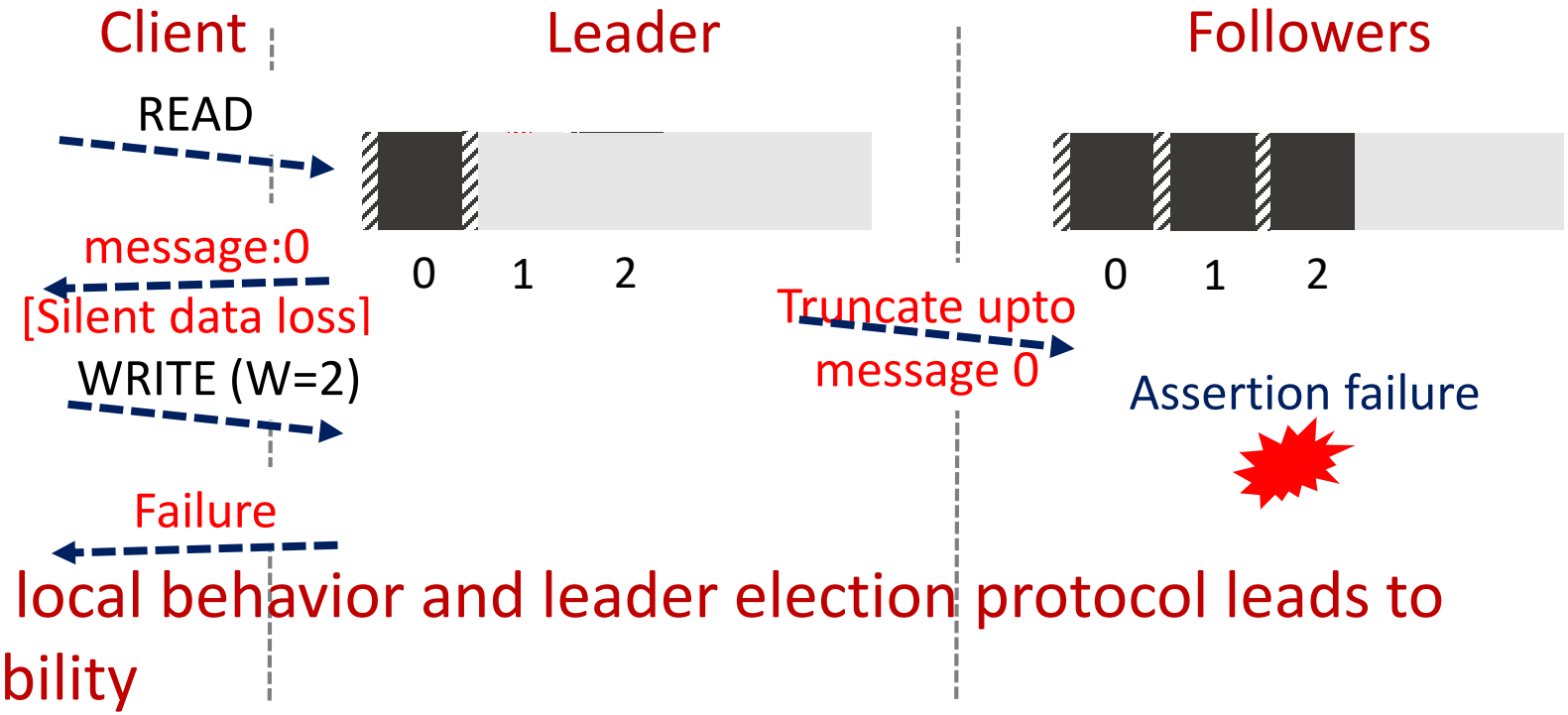
Local Behavior

Disk corruption
Checksum mismatch
Action: Truncate log at 0
Lose committed data!

Set of in-sync replicas

Node1 with truncated log not removed from in-sync replicas

Node 1 elected as leader

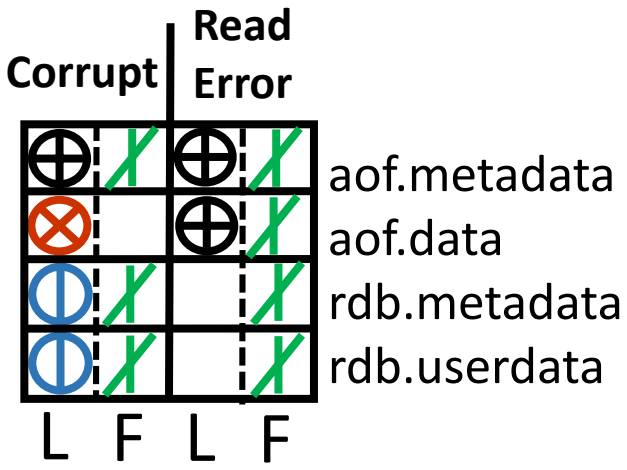


Unsafe interaction between local behavior and leader election protocol leads to data loss and write unavailability

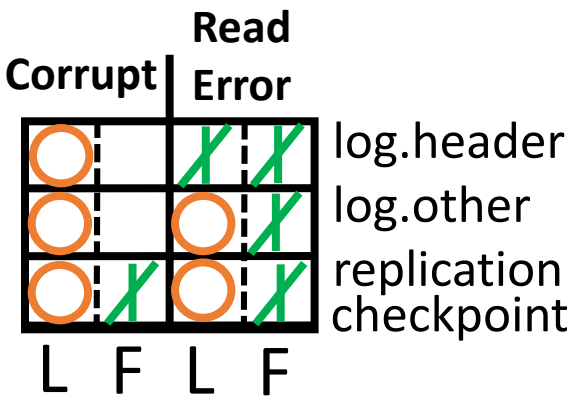
Need for synergy between local behavior and global protocol

Redundancy Does not Provide Fault Tolerance

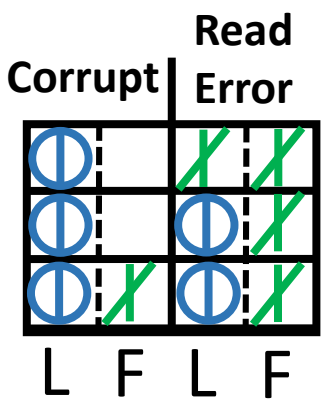
Redis Read



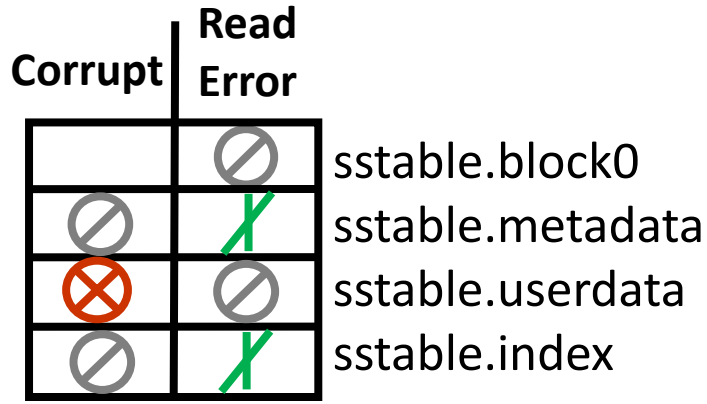
Kafka Read



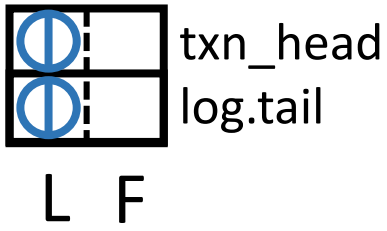
Kafka Write



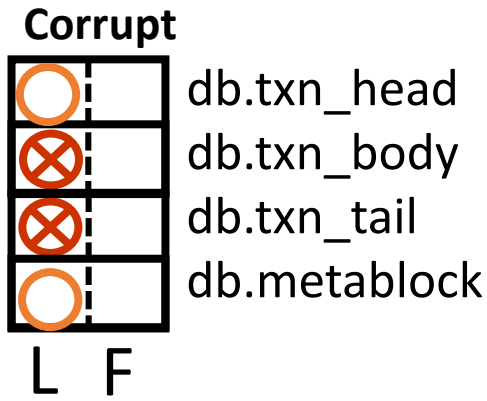
Cassandra Read

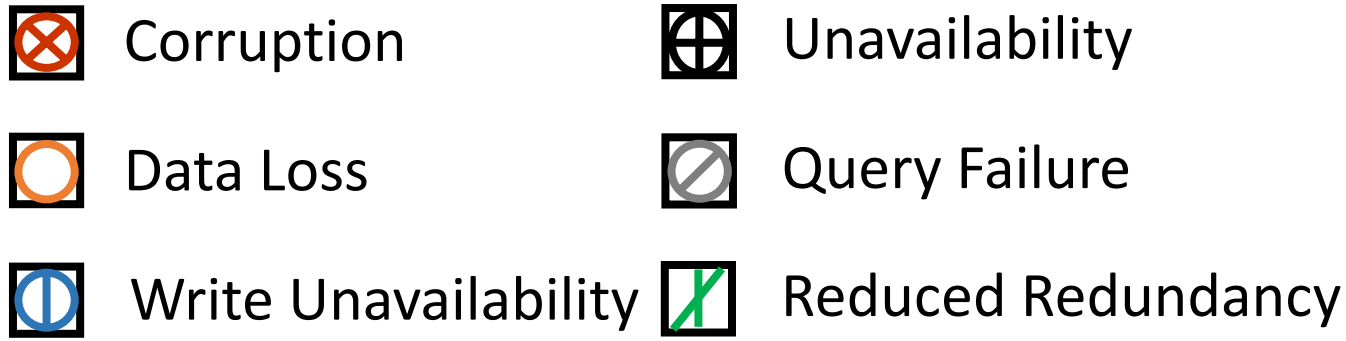


ZooKeeper Write



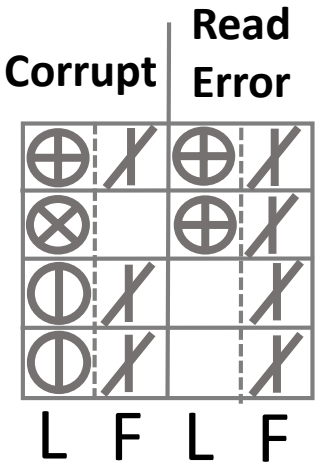
RethinkDB Read



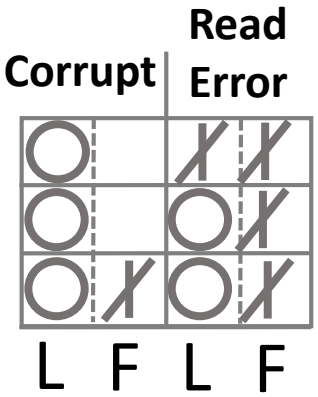


Why does Redundancy Not Imply Fault Tolerance?

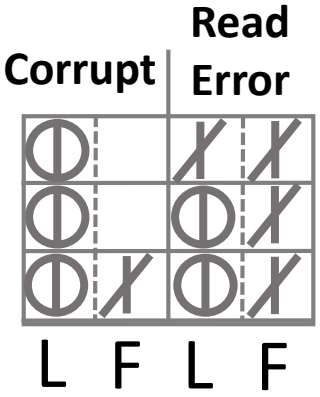
Redis Read



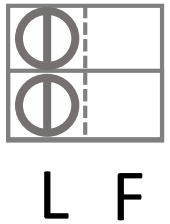
Kafka Read



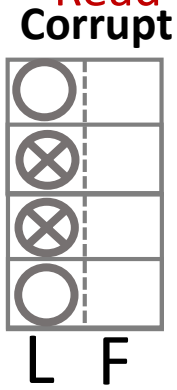
Kafka Write



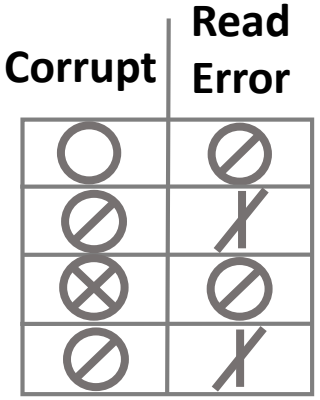
ZooKeeper Write



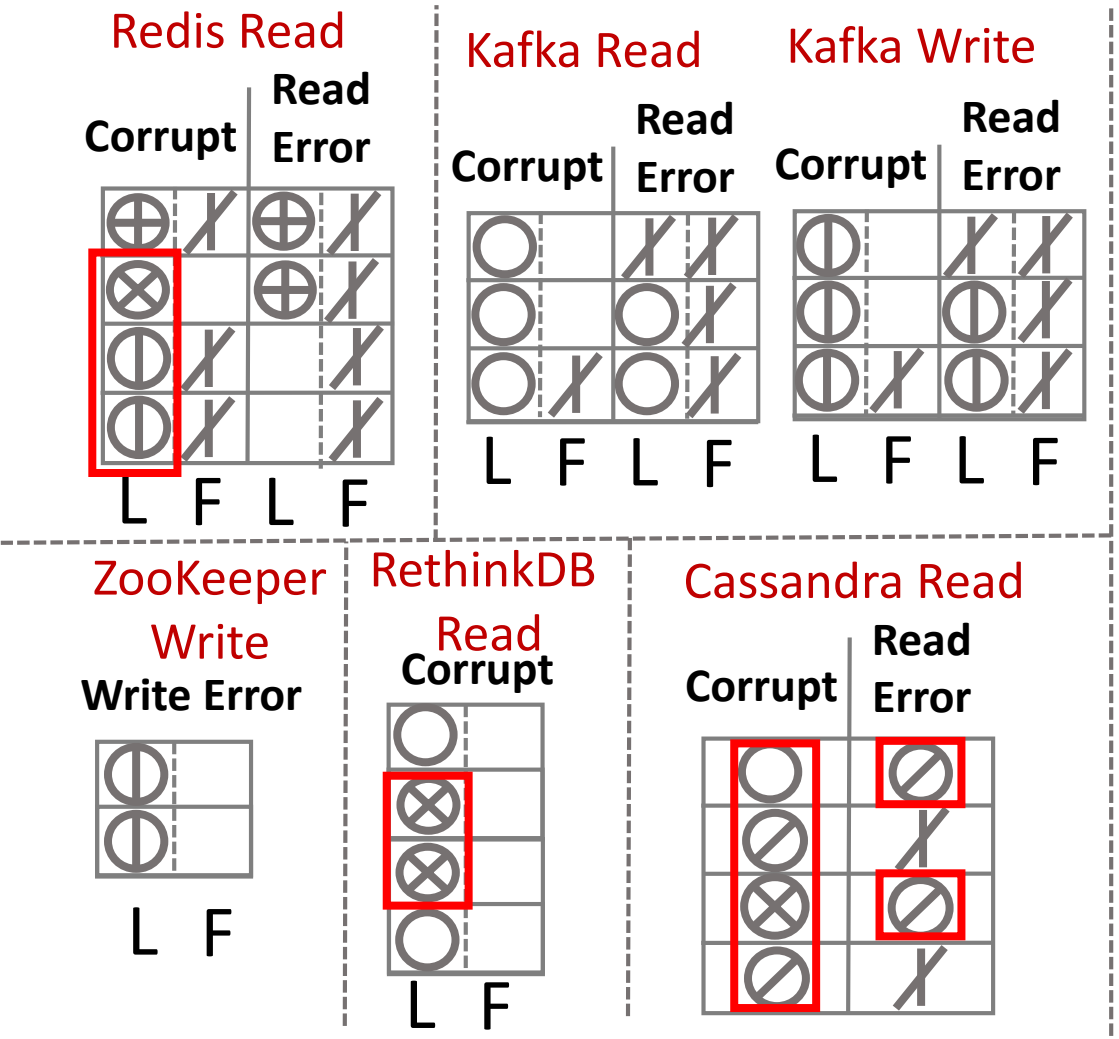
RethinkDB Read



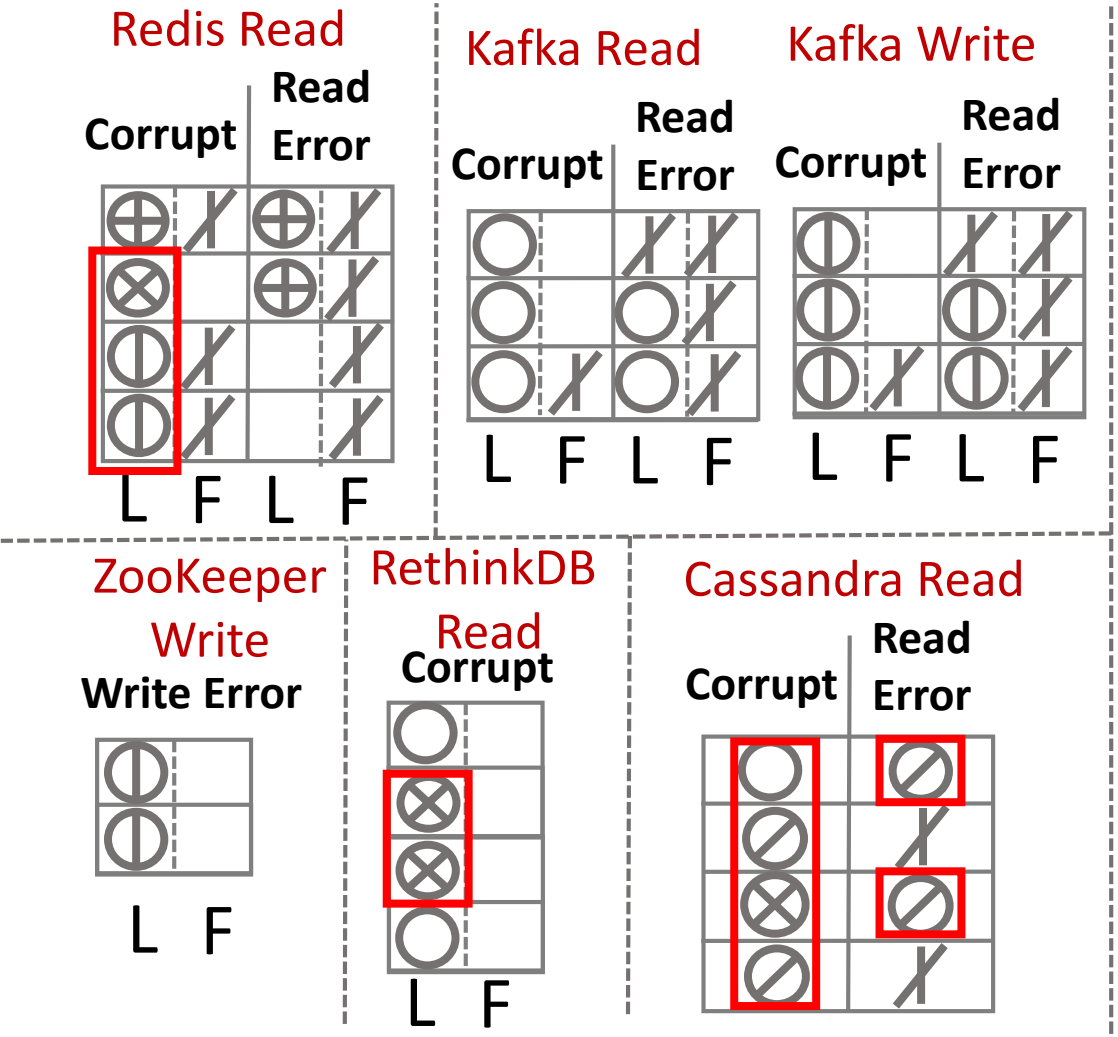
Cassandra Read



Why does Redundancy Not Imply Fault Tolerance?

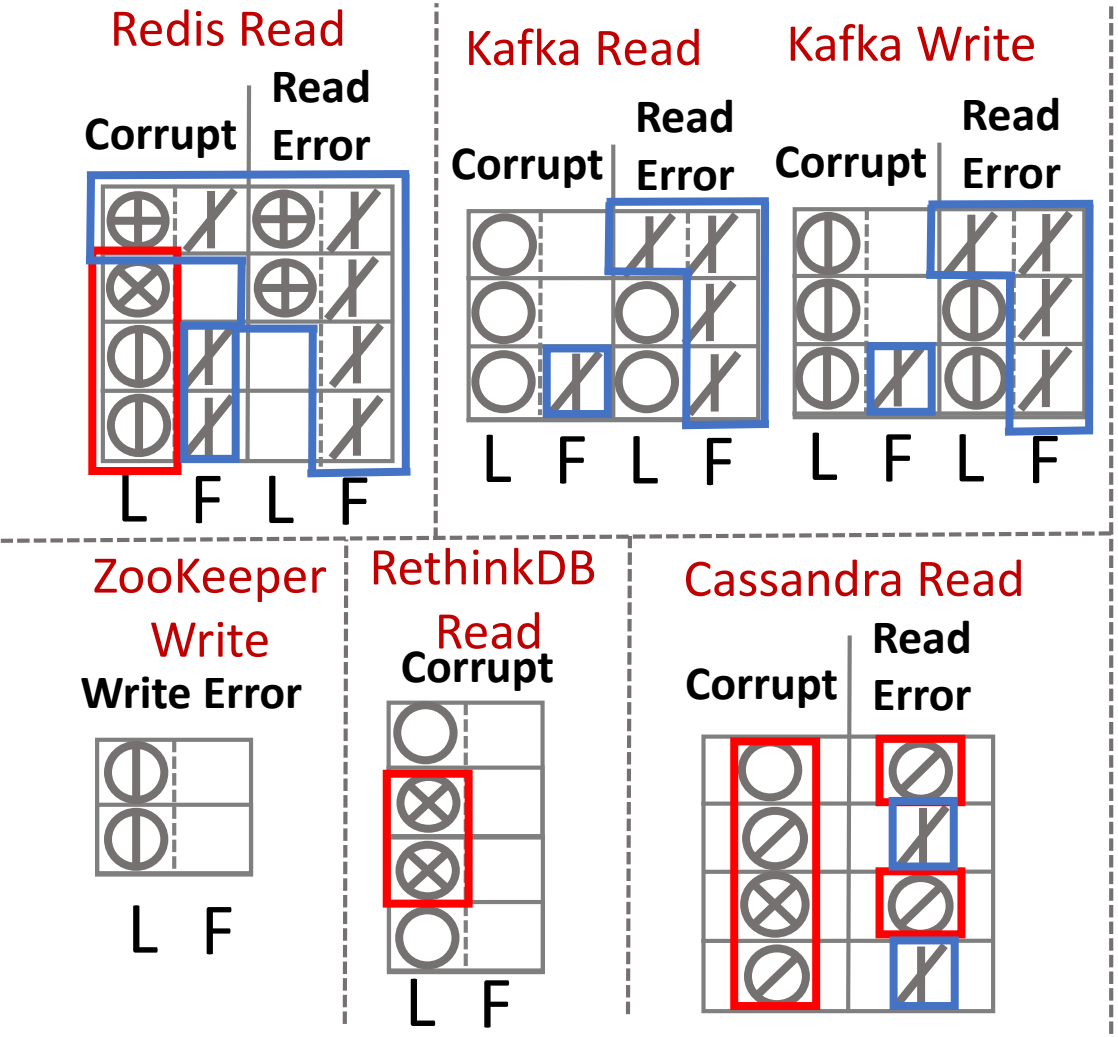


Why does Redundancy Not Imply Fault Tolerance?



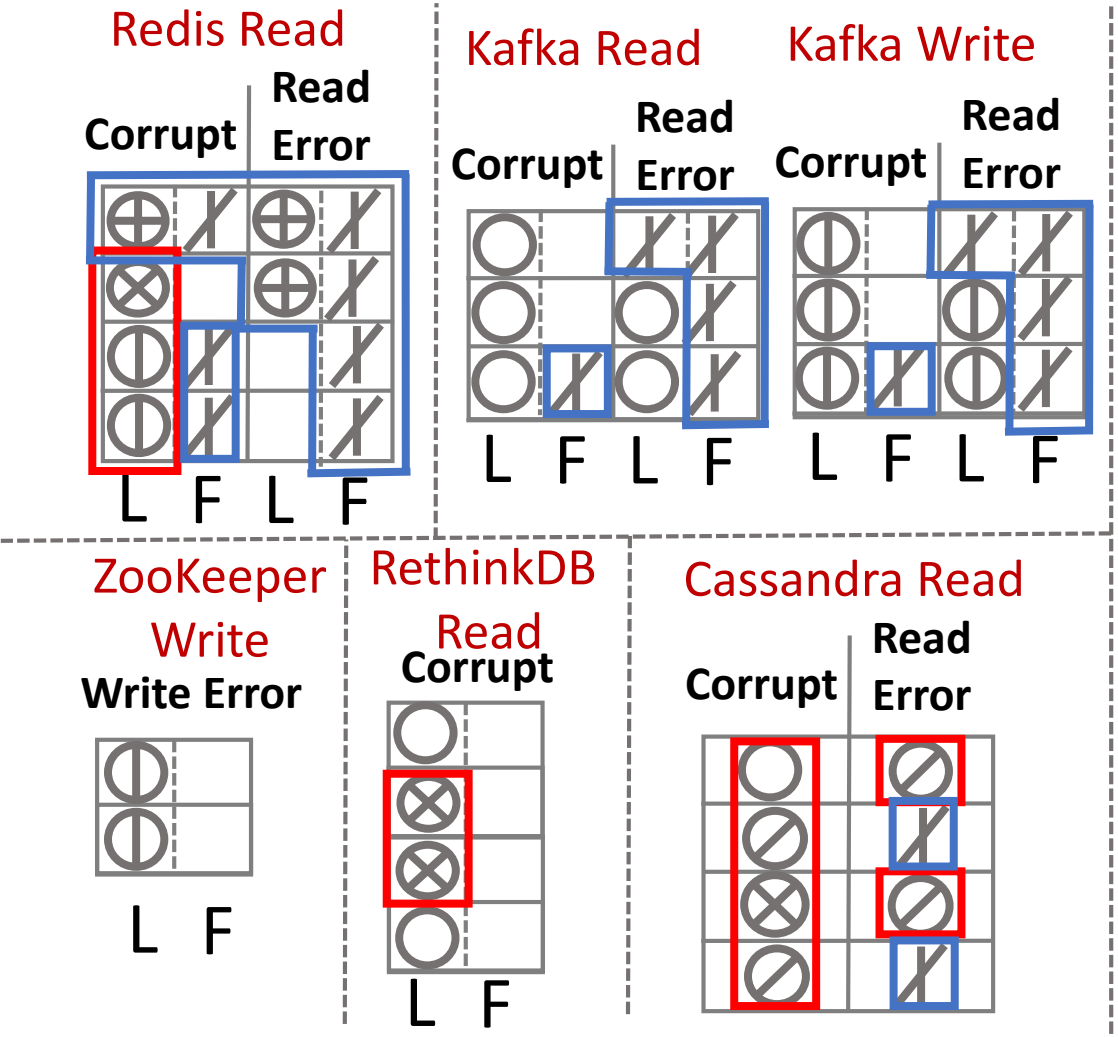
Faults are often locally undetected

Why does Redundancy Not Imply Fault Tolerance?



Faults are often locally undetected

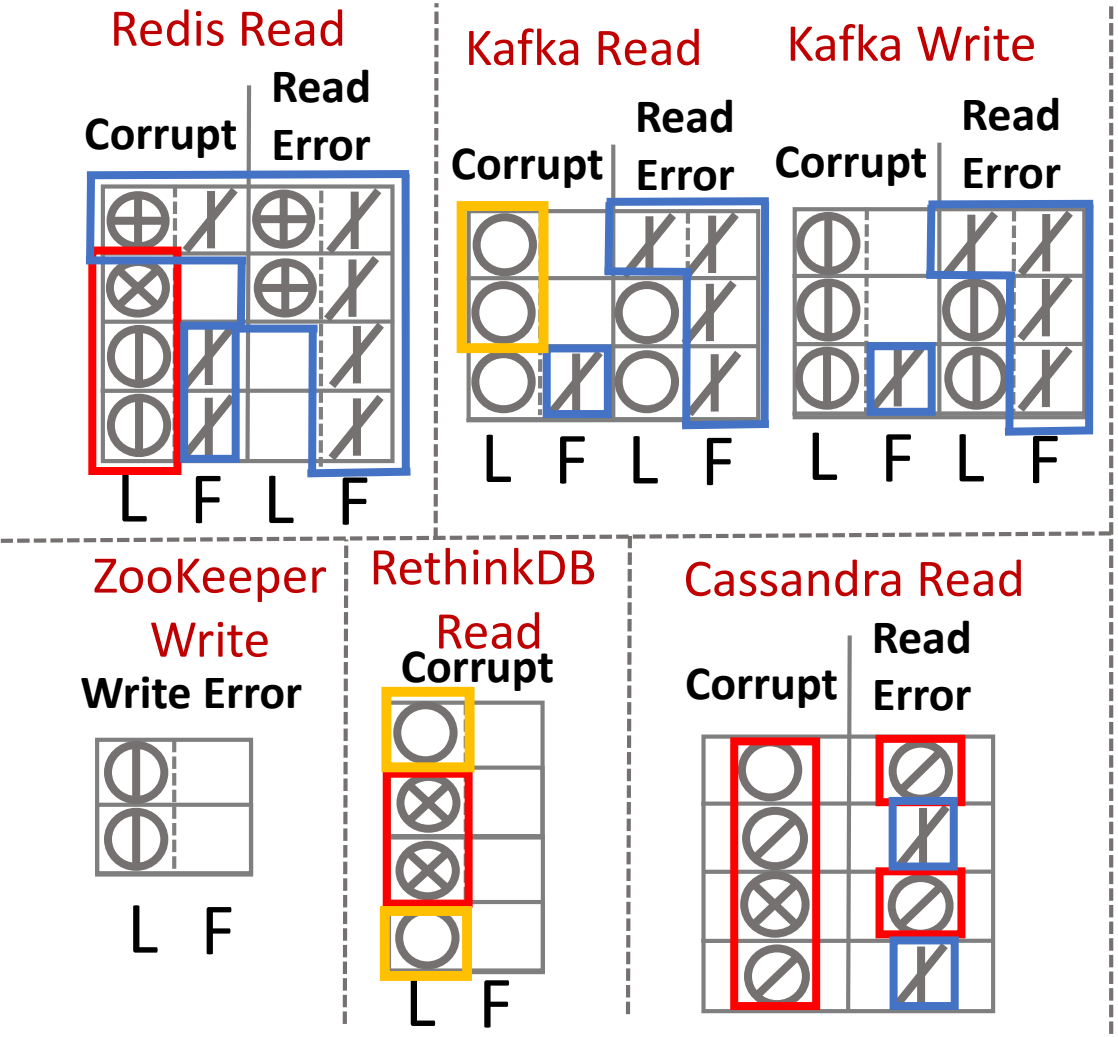
Why does Redundancy Not Imply Fault Tolerance?



Faults are often locally undetected

Crashing on detecting faults is the common reaction

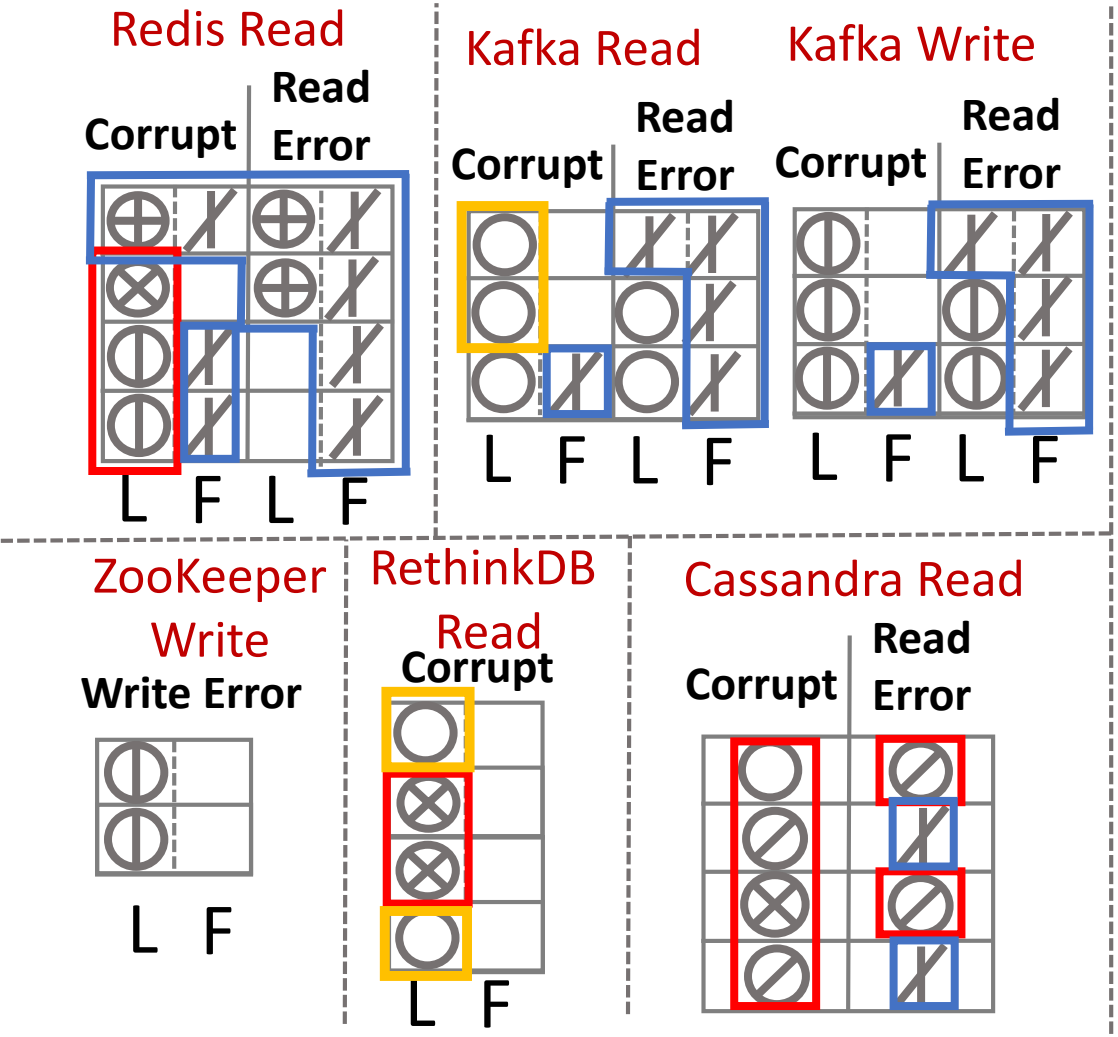
Why does Redundancy Not Imply Fault Tolerance?



Faults are often locally undetected

Crashing on detecting faults is the common reaction

Why does Redundancy Not Imply Fault Tolerance?

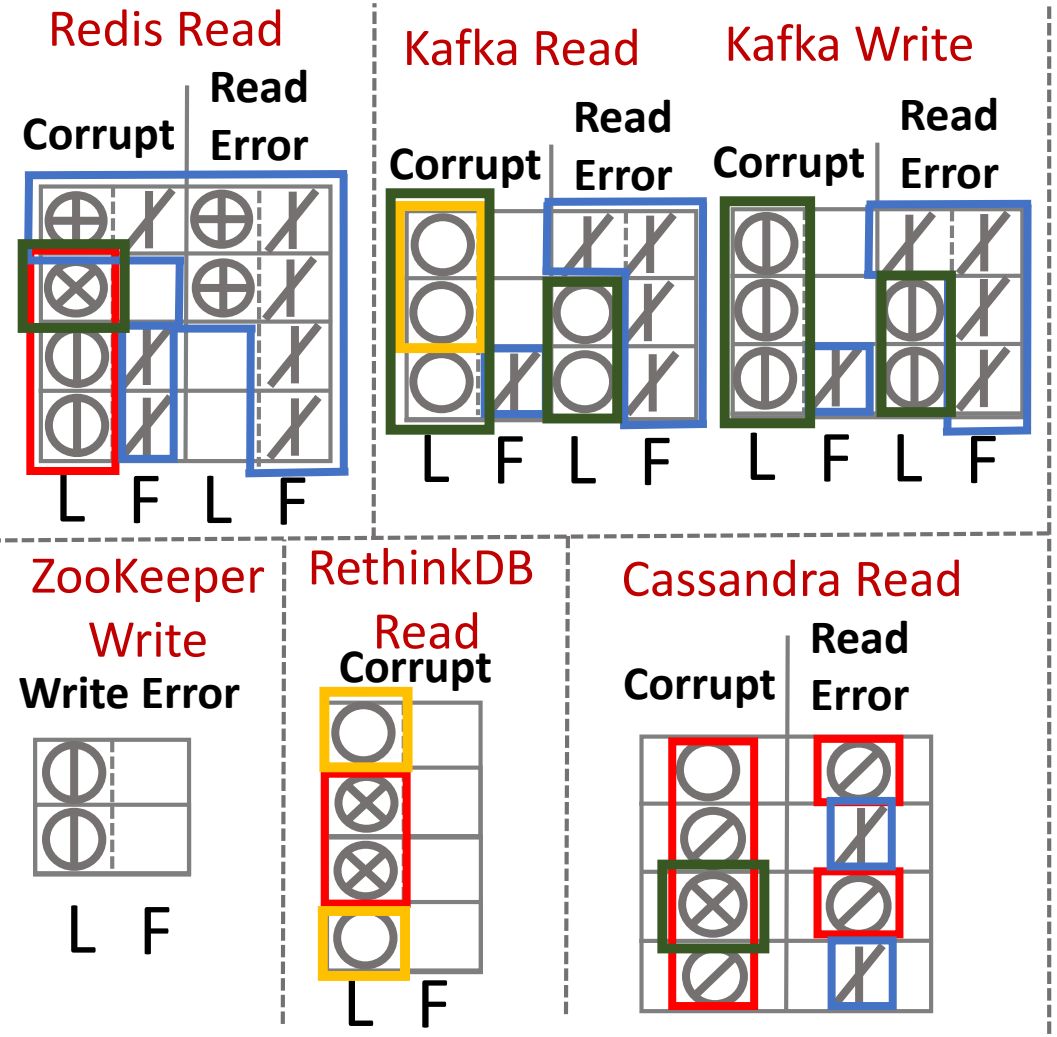


Faults are often locally undetected

Crashing on detecting faults is the common reaction

Crash and corruption handling are entangled

Why does Redundancy Not Imply Fault Tolerance?

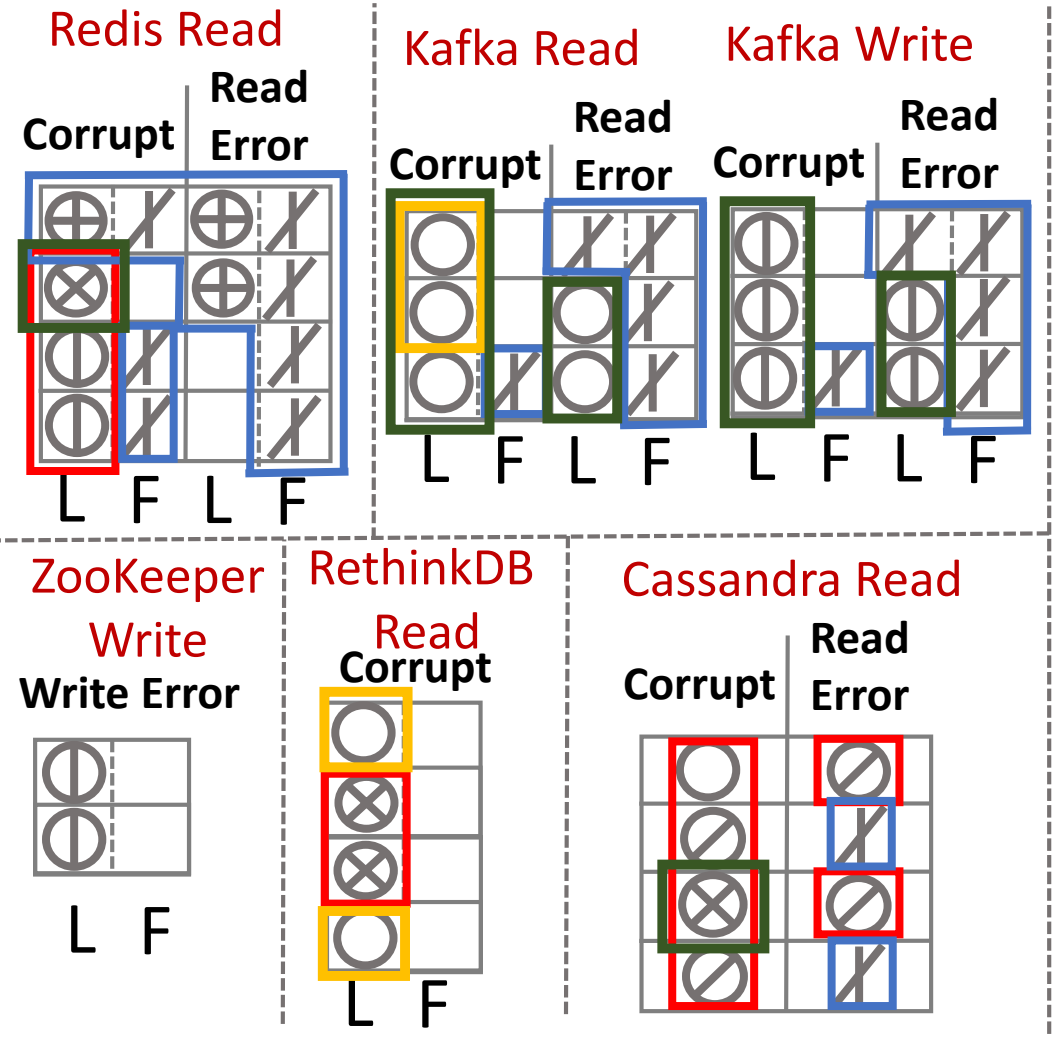


Faults are often locally undetected

Crashing on detecting faults is the common reaction

Crash and corruption handling are entangled

Why does Redundancy Not Imply Fault Tolerance?



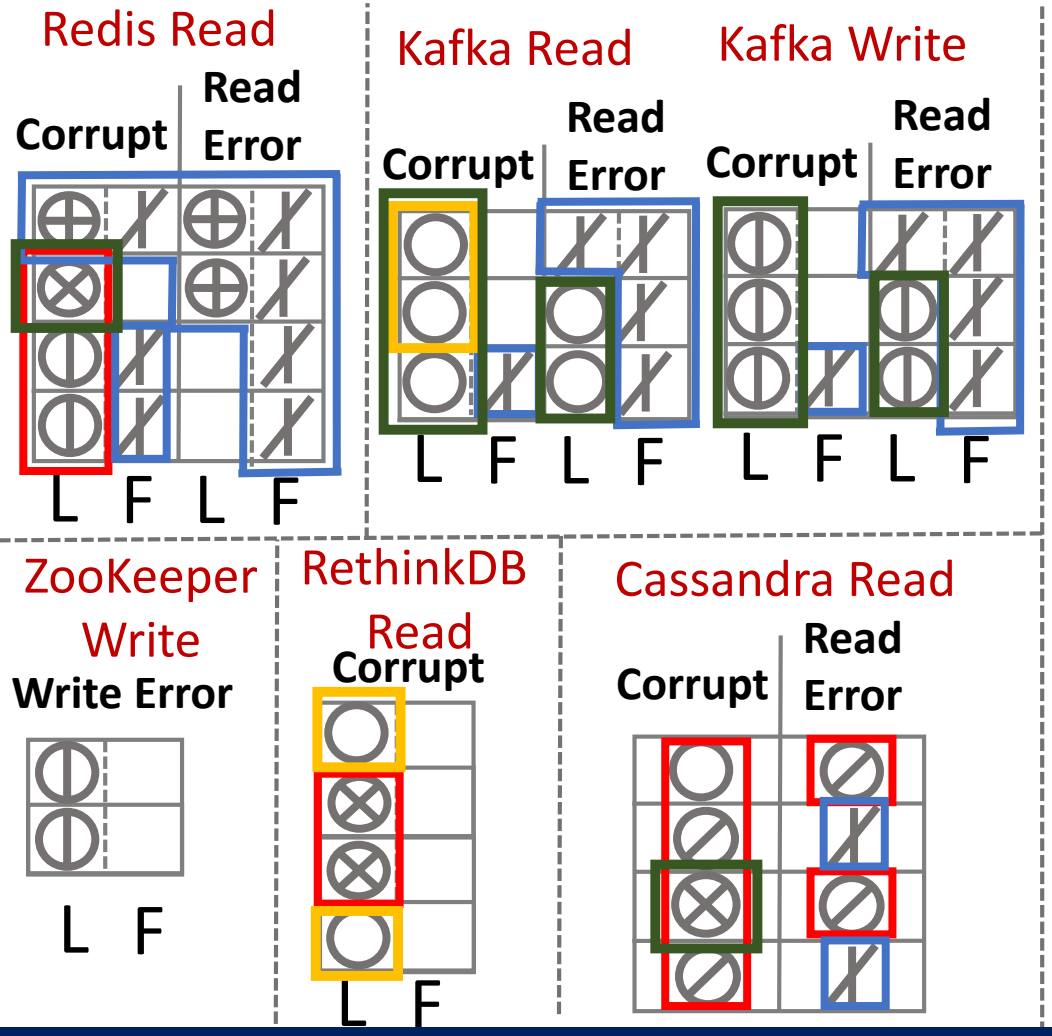
Faults are often locally undetected

Crashing on detecting faults is the common reaction

Crash and corruption handling are entangled

Unsafe interaction between local and global protocols

Why does Redundancy Not Imply Fault Tolerance?



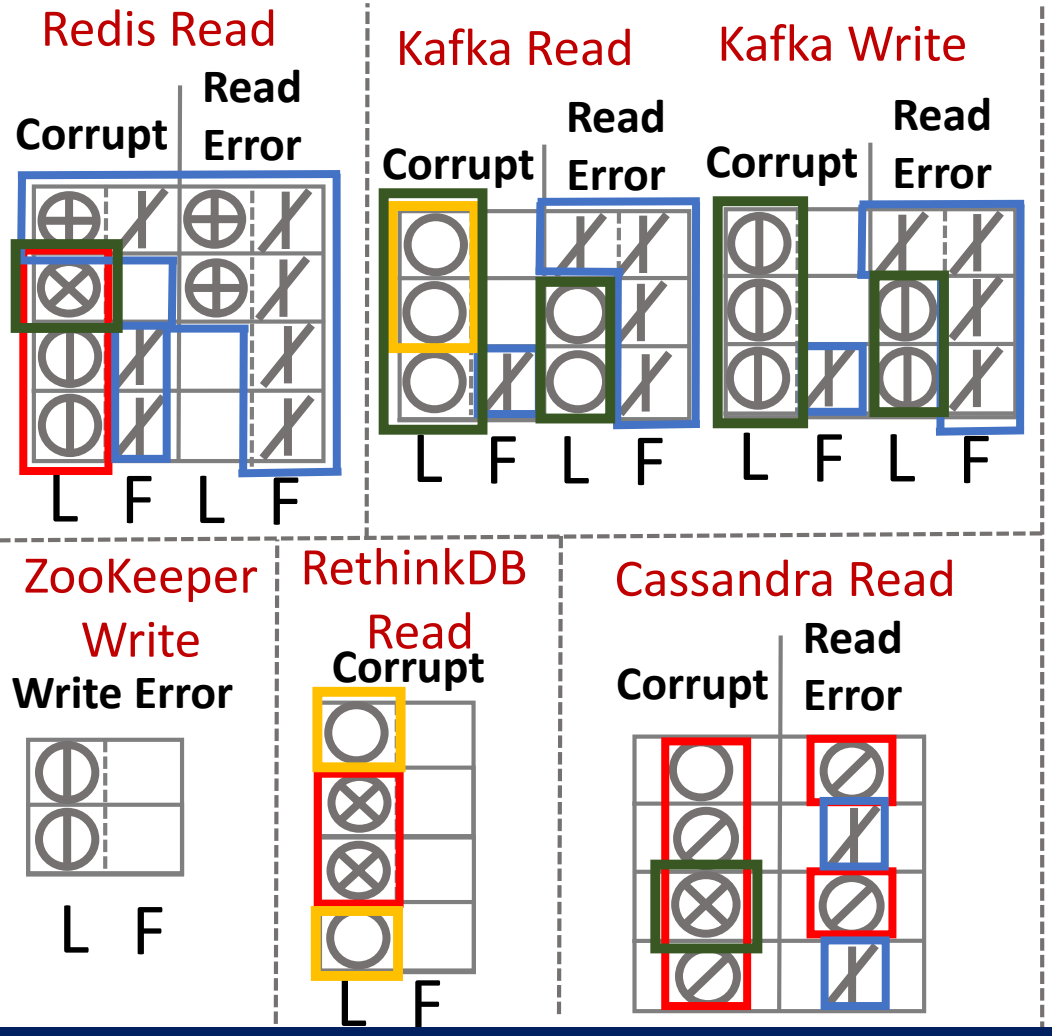
Faults are often locally undetected

Crashing on detecting faults is the common reaction

Crash and corruption handling are entangled

Unsafe interaction between local and global protocols

Why does Redundancy Not Imply Fault Tolerance?



Faults are often locally undetected

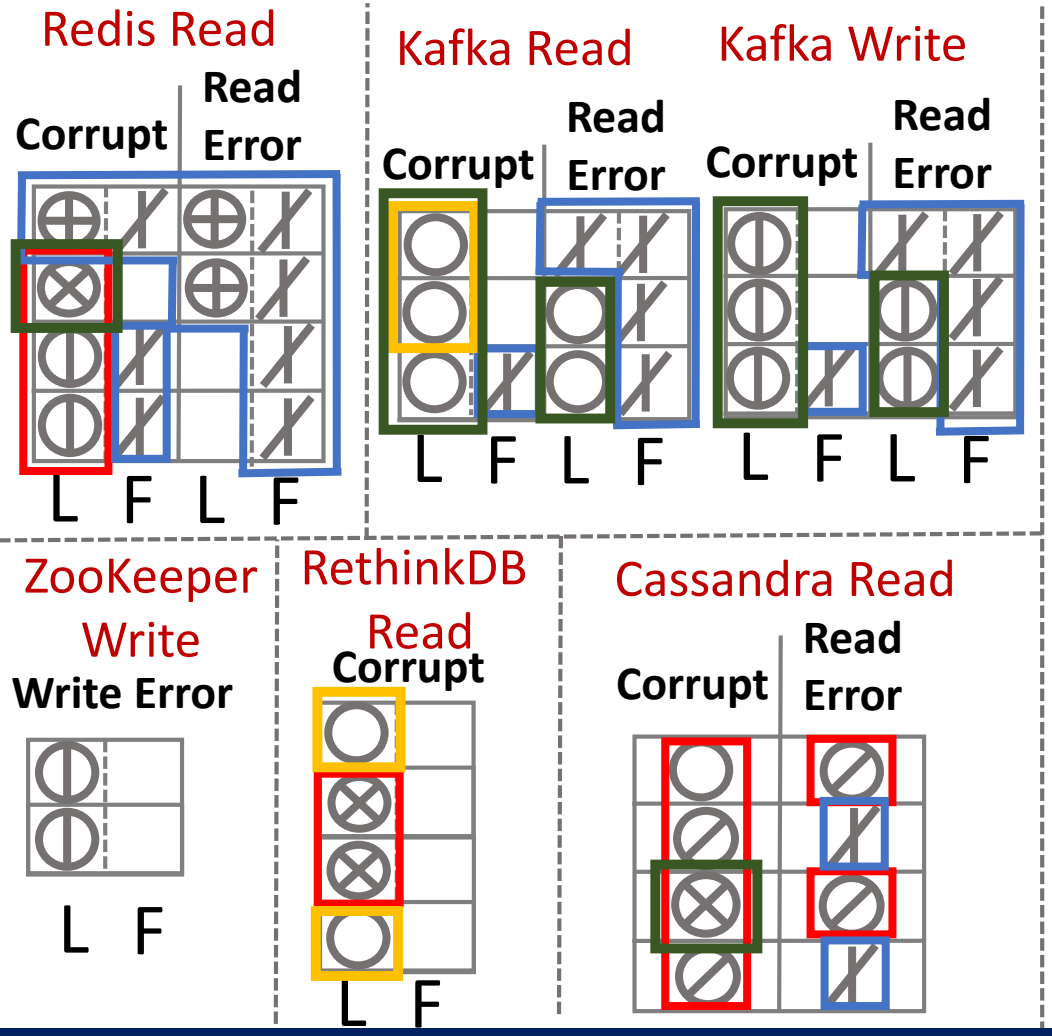
Crashing on detecting faults is the common reaction

Crash and corruption handling are entangled

Unsafe interaction between local and global protocols

Not simple implementation bugs - fundamental problems across multiple systems!

Why does Redundancy Not Imply Fault Tolerance?



Faults are often locally undetected

Crashing on detecting faults is the common reaction

Crash and corruption handling are entangled

Unsafe interaction between local and global protocols

Not simple implementation bugs - fundamental problems across multiple systems!

Redundancy underutilized as a source of recovery

Fundamental Problems: Summary

Fundamental Problems: Summary

	Locally undetected faults?	Crashing - common local action?	Crash & corruption handling entangled?	Unsafe interaction of local & global protocols?	Redundancy underutilized for recovery?
Redis					
ZooKeeper					
Cassandra					
Kafka					
RethinkDB					
MongoDB					
LogCabin					
CockroachDB					

Fundamental Problems: Summary

	Locally undetected faults?	Crashing - common local action?	Crash & corruption handling entangled?	Unsafe interaction of local & global protocols?	Redundancy underutilized for recovery?
Redis	■	■		■	
ZooKeeper	■	■	■		
Cassandra	■	■		■	
Kafka		■	■	■	
RethinkDB	■	■	■		
MongoDB		■	■		
LogCabin		■	■		
CockroachDB	■	■			

Fundamental Problems: Summary

	Locally undetected faults?	Crashing - common local action?	Crash & corruption handling entangled?	Unsafe interaction of local & global protocols?	Redundancy underutilized for recovery?
Redis	■	■		■	■
ZooKeeper	■	■	■		■
Cassandra	■	■		■	■
Kafka		■	■	■	■
RethinkDB	■	■	■		■
MongoDB		■	■		■
LogCabin		■	■		■
CockroachDB	■	■			■

Fundamental Problems: Summary

	Locally undetected faults?	Crashing - common local action?	Crash & corruption handling entangled?	Unsafe interaction of local & global protocols?	Redundancy underutilized for recovery?
Redis	■	■		■	■
ZooKeeper	■	■	■		■
Cassandra	■	■		■	■
Kafka		■	■	■	■
RethinkDB	■	■	■		■
MongoDB		■	■		■
LogCabin		■	■		■
CockroachDB	■	■			■

More observations, results, and discussions in the paper …

Outline

Introduction

Fault Injection

System Behavior Analysis

Major Results

Observations Across Systems

- Faults are Often Undetected Locally
- Crashing: Common Local Reaction
- Crash and Corruption Handling are Entangled
- Unsafe interaction between local and global protocols

Conclusion

Summary

Summary

We analyzed distributed storage reactions to single file-system faults

Summary

We analyzed distributed storage reactions to single file-system faults

Redis, ZooKeeper, Cassandra, Kafka, MongoDB, LogCabin, RethinkDB, and CockroachDB

Summary

We analyzed distributed storage reactions to single file-system faults

Redis, ZooKeeper, Cassandra, Kafka, MongoDB, LogCabin, RethinkDB, and CockroachDB

Redundancy does not provide fault tolerance

Summary

We analyzed distributed storage reactions to single file-system faults

Redis, ZooKeeper, Cassandra, Kafka, MongoDB, LogCabin, RethinkDB, and CockroachDB

Redundancy does not provide fault tolerance

A single fault in one node can cause catastrophic outcomes

Summary

We analyzed distributed storage reactions to single file-system faults

Redis, ZooKeeper, Cassandra, Kafka, MongoDB, LogCabin, RethinkDB, and CockroachDB

Redundancy does not provide fault tolerance

A single fault in one node can cause catastrophic outcomes

data loss, corruption, unavailability, and spread of corruption to other intact replicas

Summary

We analyzed distributed storage reactions to single file-system faults

Redis, ZooKeeper, Cassandra, Kafka, MongoDB, LogCabin, RethinkDB, and CockroachDB

Redundancy does not provide fault tolerance

A single fault in one node can cause catastrophic outcomes

data loss, corruption, unavailability, and spread of corruption to other intact replicas

Some fundamental problems across multiple systems:

Summary

We analyzed distributed storage reactions to single file-system faults

Redis, ZooKeeper, Cassandra, Kafka, MongoDB, LogCabin, RethinkDB, and CockroachDB

Redundancy does not provide fault tolerance

A single fault in one node can cause catastrophic outcomes

data loss, corruption, unavailability, and spread of corruption to other intact replicas

Some fundamental problems across multiple systems:

Faults are often undetected locally – leads to harmful global effects

Summary

We analyzed distributed storage reactions to single file-system faults

Redis, ZooKeeper, Cassandra, Kafka, MongoDB, LogCabin, RethinkDB, and CockroachDB

Redundancy does not provide fault tolerance

A single fault in one node can cause catastrophic outcomes

data loss, corruption, unavailability, and spread of corruption to other intact replicas

Some fundamental problems across multiple systems:

Faults are often undetected locally – leads to harmful global effects

On detection, crashing is the common action – redundancy underutilized

Summary

We analyzed distributed storage reactions to single file-system faults

- Redis, ZooKeeper, Cassandra, Kafka, MongoDB, LogCabin, RethinkDB, and CockroachDB

Redundancy does not provide fault tolerance

- A single fault in one node can cause catastrophic outcomes
- data loss, corruption, unavailability, and spread of corruption to other intact replicas

Some fundamental problems across multiple systems:

- Faults are often undetected locally – leads to harmful global effects
- On detection, crashing is the common action – redundancy underutilized
- Crash and corruption handling are entangled – loss of committed data

Summary

We analyzed distributed storage reactions to single file-system faults

- Redis, ZooKeeper, Cassandra, Kafka, MongoDB, LogCabin, RethinkDB, and CockroachDB

Redundancy does not provide fault tolerance

- A single fault in one node can cause catastrophic outcomes
- data loss, corruption, unavailability, and spread of corruption to other intact replicas

Some fundamental problems across multiple systems:

- Faults are often undetected locally – leads to harmful global effects
- On detection, crashing is the common action – redundancy underutilized
- Crash and corruption handling are entangled – loss of committed data
- Unsafe interaction between local behavior and global distributed protocols can spread corruption or data loss

Conclusion

Conclusion

Most distributed systems not yet resilient

Conclusion

Most distributed systems not yet resilient

Always detect faults – important in layered stacks on commodity hardware

Conclusion

Most distributed systems not yet resilient

Always detect faults – important in layered stacks on commodity hardware

Detecting faults and not using redundancy to recover is undesirable

Conclusion

Most distributed systems not yet resilient

Always detect faults – important in layered stacks on commodity hardware

Detecting faults and not using redundancy to recover is undesirable

Cannot always assume corruption to be caused by a crash

Conclusion

Most distributed systems not yet resilient

Always detect faults – important in layered stacks on commodity hardware

Detecting faults and not using redundancy to recover is undesirable

Cannot always assume corruption to be caused by a crash

Local behavior has implications for distributed systems

Conclusion

Most distributed systems not yet resilient

Always detect faults – important in layered stacks on commodity hardware

Detecting faults and not using redundancy to recover is undesirable

Cannot always assume corruption to be caused by a crash

Local behavior has implications for distributed systems

Our study provides directions for more robust distributed storage design

Conclusion

Most distributed systems not yet resilient

Always detect faults – important in layered stacks on commodity hardware

Detecting faults and not using redundancy to recover is undesirable

Cannot always assume corruption to be caused by a crash

Local behavior has implications for distributed systems

Our study provides directions for more robust distributed storage design

Our fault injection framework available online:
http://research.cs.wisc.edu/adsl/Software/cords/

Thank you!