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
- disk block corruption

File system may return I/O error on a read
- latent sector error

We call these file-system faults

Do distributed storage systems use redundancy to recover from local 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

<table>
<thead>
<tr>
<th></th>
<th>Silent corruption</th>
<th>Unavailability</th>
<th>Data loss</th>
<th>Reduced redundancy</th>
<th>Query failures</th>
</tr>
</thead>
<tbody>
<tr>
<td>Redis</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ZooKeeper</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cassandra</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Kafka</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RethinkDB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MongoDB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LogCabin</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CockroachDB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th></th>
<th>Silent corruption</th>
<th>Unavailability</th>
<th>Data loss</th>
<th>Reduced redundancy</th>
<th>Query failures</th>
</tr>
</thead>
<tbody>
<tr>
<td>Redis</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>ZooKeeper</td>
<td></td>
<td>✗</td>
<td>✗</td>
<td></td>
<td>✗</td>
</tr>
<tr>
<td>Cassandra</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td></td>
<td>✗</td>
</tr>
<tr>
<td>Kafka</td>
<td></td>
<td>✗</td>
<td>✗</td>
<td></td>
<td>✗</td>
</tr>
<tr>
<td>RethinkDB</td>
<td>✗</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MongoDB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LogCabin</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CockroachDB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Redis</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ZooKeeper</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cassandra</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Kafka</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RethinkDB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MongoDB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LogCabin</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CockroachDB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Fundamental Problems: Summary

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Redis</td>
<td><img src="#" alt="Red" /></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ZooKeeper</td>
<td><img src="#" alt="Red" /></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cassandra</td>
<td><img src="#" alt="Red" /></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Kafka</td>
<td><img src="#" alt="Red" /></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RethinkDB</td>
<td><img src="#" alt="Red" /></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MongoDB</td>
<td><img src="#" alt="Red" /></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LogCabin</td>
<td><img src="#" alt="Red" /></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CockroachDB</td>
<td><img src="#" alt="Red" /></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Fundamental Problems: Summary

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Redis</td>
<td>🟢</td>
<td>🟢</td>
<td>🟢</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ZooKeeper</td>
<td>🟢</td>
<td>🟢</td>
<td>🟢</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cassandra</td>
<td>🟢</td>
<td>🟢</td>
<td>🟢</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Kafka</td>
<td>🟢</td>
<td>🟢</td>
<td>🟢</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RethinkDB</td>
<td>🟢</td>
<td>🟢</td>
<td>🟢</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MongoDB</td>
<td>🟢</td>
<td>🟢</td>
<td>🟢</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LogCabin</td>
<td>🟢</td>
<td>🟢</td>
<td>🟢</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CockroachDB</td>
<td>🟢</td>
<td>🟢</td>
<td>🟢</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Fundamental Problems: Summary

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Redis</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ZooKeeper</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cassandra</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Kafka</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RethinkDB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MongoDB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LogCabin</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CockroachDB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# Fundamental Problems: Summary

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Redis</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>ZooKeeper</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Cassandra</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Kafka</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>RethinkDB</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>MongoDB</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>LogCabin</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>CockroachDB</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
</tbody>
</table>
## Fundamental Problems: Summary

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Redis</td>
<td>🟥</td>
<td>🟥</td>
<td>🟥</td>
<td>🟥</td>
<td>🟥</td>
</tr>
<tr>
<td>ZooKeeper</td>
<td>🟥</td>
<td>🟥</td>
<td>🟥</td>
<td>🟥</td>
<td>🟥</td>
</tr>
<tr>
<td>Cassandra</td>
<td>🟥</td>
<td>🟥</td>
<td>🟥</td>
<td>🟥</td>
<td>🟥</td>
</tr>
<tr>
<td>Kafka</td>
<td>🟥</td>
<td>🟥</td>
<td>🟥</td>
<td>🟥</td>
<td>🟥</td>
</tr>
<tr>
<td>RethinkDB</td>
<td>🟥</td>
<td>🟥</td>
<td>🟥</td>
<td>🟥</td>
<td>🟥</td>
</tr>
<tr>
<td>MongoDB</td>
<td>🟥</td>
<td>🟥</td>
<td>🟥</td>
<td>🟥</td>
<td>🟥</td>
</tr>
<tr>
<td>LogCabin</td>
<td>🟥</td>
<td>🟥</td>
<td>🟥</td>
<td>🟥</td>
<td>🟥</td>
</tr>
<tr>
<td>CockroachDB</td>
<td>🟥</td>
<td>🟥</td>
<td>🟥</td>
<td>🟥</td>
<td>🟥</td>
</tr>
</tbody>
</table>
Outline

Introduction
Fault Injection
System Behavior Analysis
Major Results
Observations Across Systems
Conclusion
Fault Model
Fault Model

Server 1  Server 2  Server 3
Fault Model
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
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

Client

Server 1

Request

Server 2

Server 3

ext4

ext4

ext4
Fault Model: ext4 and btrfs

Client → Request → Server 1 → Server 2 → Server 3

File System

ext4

Corrupt data
Fault Model: ext4 and btrfs
Fault Model: ext4 and btrfs

Ext4: disk corruption  $\rightarrow$ 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 $\rightarrow$ corrupted data to applications
Fault Model: ext4 and btrfs

Ext4: disk corruption $\rightarrow$ corrupted data to applications

Btrfs: disk corruption $\rightarrow$ 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 for current run: server 1, block B1 read corruption
Fault Injection Methodology

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

Fault for current run: server 1, block B1 read corruption
Fault Injection Methodology

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

Fault for current run: server 1, block B1 read corruption
Fault Injection Methodology

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

Fault for current run: server 1, block B1 read corruption
Fault Injection Methodology

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

Fault for current run: server 1, block B1 read corruption
Fault Injection Methodology

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

Fault for current run: server 1, block B1 read corruption

Local Behavior
Fault Injection Methodology

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

**Local Behavior**
Crash
Retry
Ignore faulty data
No detection/recovery
Fault Injection Methodology

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

**Local Behavior**
- Crash
- Retry
- Ignore faulty data
- No detection/recovery

**Global Effect**
Fault Injection Methodology

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

**Fault for current run:**
server 1, block B1
read corruption

**Local Behavior**
- Crash
- Retry
- Ignore faulty data
- No detection/recovery

**Global Effect**
- Corruption
- Data loss
- Unavailability

**Diagram:**
- Client
  - (Read) [Server 1]
  - [errfs (FUSE FS)]
    - read B1-B4
    - return B1'-B4

- [errfs (FUSE FS)]
  - read B1-B4
  - return B1-B4

- [errfs (FUSE FS)]
  - return

- [errfs (FUSE FS)]

- [Server 2]
  - (Read) [Server 3]
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

Client \(\xrightarrow{\text{Write}}\) Leader

Leader \(\xrightarrow{\text{appendonlyfile}}\) Follower

Follower \(\xrightarrow{\text{appendonlyfile}}\) Follower
An Example: Redis

Redis is a popular data structure store

Client → Write → Leader

Leader: appendonlyfile

Follower: appendonlyfile

Follower: appendonlyfile
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

<table>
<thead>
<tr>
<th>Corrupt</th>
<th>Read I/O Error</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Redis: Behavior Analysis

## Read Workload

### Local Behavior

<table>
<thead>
<tr>
<th>Corrupt</th>
<th>Read I/O Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>L</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>
Redis: Behavior Analysis

Read Workload

Local Behavior

<table>
<thead>
<tr>
<th>Corrupt</th>
<th>Read I/O Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>L</td>
</tr>
<tr>
<td>L</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>L</td>
</tr>
</tbody>
</table>

L Leader
Redis: Behavior Analysis

Read Workload

Local Behavior

Corrupt | Read I/O Error
------- | -----------------
L       | L
F       | F
L       | L
F       | F

L Leader
F Follower
Redis: Behavior Analysis

Read Workload

Local Behavior

<table>
<thead>
<tr>
<th>Corrupt</th>
<th>Read I/O Error</th>
<th>On-disk Structures</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td></td>
<td>appendonlyfile.metadata</td>
</tr>
<tr>
<td>F</td>
<td></td>
<td>appendonlyfile.data</td>
</tr>
<tr>
<td>L</td>
<td></td>
<td>redis_database.block_0</td>
</tr>
<tr>
<td>F</td>
<td></td>
<td>redis_database.metadata</td>
</tr>
<tr>
<td>L</td>
<td></td>
<td>redis_database.userdata</td>
</tr>
</tbody>
</table>

L Leader

F Follower
Redis: Behavior Analysis

Read Workload

Local Behavior

L Leader
F Follower

On-disk Structures

appendonlyfile.metadata
appendonlyfile.data
redis_database.block_0
redis_database.metadata
redis_database.userdata
Redis: Behavior Analysis

Read Workload

Local Behavior

Corrupt

Read I/O Error

On-disk Structures

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

L  Leader
F  Follower
Redis: Behavior Analysis

Read Workload

Local Behavior

Corrupt | Read I/O Error
---------|------------------
\(\times\) | 

On-disk Structures

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

\(L\) Leader
\(F\) Follower

Local Behavior

\(\times\) Crash
Redis: Behavior Analysis

Read Workload

<table>
<thead>
<tr>
<th>Local Behavior</th>
<th>Read I/O Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Corrupt</td>
<td></td>
</tr>
<tr>
<td>L</td>
<td>F</td>
</tr>
<tr>
<td>L</td>
<td>L</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

On-disk Structures

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

Global Effect

<table>
<thead>
<tr>
<th>Corrupt</th>
<th>Read I/O Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>F</td>
</tr>
<tr>
<td>L</td>
<td>L</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

Local Behavior

L Leader
F Follower

Crash
## Redis: Behavior Analysis

### Read Workload

**Local Behavior**

<table>
<thead>
<tr>
<th>Corrupt</th>
<th>Read I/O Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>![X]</td>
<td></td>
</tr>
</tbody>
</table>

**On-disk Structures**

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

**Global Effect**

<table>
<thead>
<tr>
<th>Corrupt</th>
<th>Read I/O Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>![X]</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>L</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>F</td>
</tr>
<tr>
<td>L</td>
<td>F</td>
</tr>
</tbody>
</table>
Redis: Behavior Analysis

Read Workload

Local Behavior

<table>
<thead>
<tr>
<th>Corrupt</th>
<th>Read I/O Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>L</td>
</tr>
</tbody>
</table>

On-disk Structures

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

Global Effect

<table>
<thead>
<tr>
<th>Corrupt</th>
<th>Read I/O Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>L</td>
</tr>
</tbody>
</table>

Local Behavior

Crash

L Leader
F Follower
Redis: Behavior Analysis

Read Workload

Local Behavior

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>F</td>
<td>L</td>
<td>F</td>
</tr>
</tbody>
</table>

Global Effect

Unavailability

On-disk Structures

appendonlyfile.metadata
appendonlyfile.data
redis_database.block_0
redis_database.metadata
redis_database.userdata
### Redis: Behavior Analysis

#### Read Workload

<table>
<thead>
<tr>
<th>Local Behavior</th>
<th>Global Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>L: Leader</td>
<td></td>
</tr>
<tr>
<td>F: Follower</td>
<td></td>
</tr>
</tbody>
</table>

#### Local Behavior

- **Corrupt**
- **Read I/O Error**

#### On-disk Structures

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

#### Global Effect

- **Corrupt**
- **Read I/O Error**

- **Unavailability**

No checksums to detect corruption
Leader crashes due to failed deserialization
No automatic failover - cluster **unavailable**
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

Global Effect:
- Unavailability

Local Behavior

Crash

L Leader
F Follower
Redis: Behavior Analysis

Read Workload

Local Behavior

<table>
<thead>
<tr>
<th>Corrupt</th>
<th>Read I/O Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>F</td>
</tr>
</tbody>
</table>

Global Effect

On-disk Structures

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

Global Effect

- Unavailability
Redis: Behavior Analysis

Read Workload

Local Behavior

<table>
<thead>
<tr>
<th>L</th>
<th>F</th>
<th>L</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
</tbody>
</table>

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

Global Effect

- Unavailability
Redis: Behavior Analysis

Local Behavior

On-disk Structures
- appendonlyfile.metadata
- appendonlyfile.data
- redis_database.block_0
- redis_database.metadata
- redis_database.userdata

Global Effect

Unavailability

Leadership Levels
- L: Leader
- F: Follower

Local Behavior
- Crash
Redis: Behavior Analysis

Read Workload

Local Behavior

Corrupt Read I/O Error

On-disk Structures

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

Global Effect

Corrupt Read I/O Error

Global Effect

Unavailability Reduced Redundancy

Local Behavior

Crash

L Leader
F Follower
## Redis: Behavior Analysis

### Read Workload

<table>
<thead>
<tr>
<th>Local Behavior</th>
<th>Global Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Corrupt</td>
<td>Leader</td>
</tr>
<tr>
<td>I/O Error</td>
<td>Follower</td>
</tr>
</tbody>
</table>

### On-disk Structures

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

### Global Effect

- Unavailability
- Reduced Redundancy

(Leader: L, Follower: F)
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

Unavailability Reduced

Redundancy

Local Behavior

Crash

Global Effect
Redis: Behavior Analysis

Read Workload

Local Behavior

Corrupt

Read I/O Error

On-disk Structures

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

Global Effect

Corrupt

Read I/O Error

Global Effect

Unavailability
Reduced
Redundancy

Leader (L)
Follower (F)

Local Behavior

Crash
## Redis: Behavior Analysis

### Read Workload

<table>
<thead>
<tr>
<th></th>
<th>Corrupt</th>
<th>Read I/O Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>❌</td>
<td>❌</td>
</tr>
<tr>
<td>F</td>
<td>❌</td>
<td>❌</td>
</tr>
</tbody>
</table>

### Local Behavior

- **On-disk Structures**
  - appendonlyfile.metadata
  - appendonlyfile.data
  - redis_database.block_0
  - redis_database.metadata
  - redis_database.userdata

### Global Effect

<table>
<thead>
<tr>
<th></th>
<th>Corrupt</th>
<th>Read I/O Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>☑️</td>
<td>☑️</td>
</tr>
<tr>
<td>F</td>
<td>☑️</td>
<td>☑️</td>
</tr>
</tbody>
</table>

### Global Effect

- **Unavailability**
- **Reduced Redundancy**
Redis: Behavior Analysis

Read Workload

Local Behavior

<table>
<thead>
<tr>
<th>Corrupt</th>
<th>Read I/O Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>☐</td>
<td>☐</td>
</tr>
</tbody>
</table>

On-disk Structures

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

Global Effect

<table>
<thead>
<tr>
<th>Corrupt</th>
<th>Read I/O Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>☐</td>
<td>☐</td>
</tr>
</tbody>
</table>

Global Effect

- Unavailability
- Reduced
- Redundancy

Local Behavior

- Crash
- No Detection/
- No Recovery

17
Redis: Behavior Analysis

Read Workload

Local Behavior

<table>
<thead>
<tr>
<th>Corrupt</th>
<th>Read I/O Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>φ</td>
<td></td>
</tr>
</tbody>
</table>

L | F | L | F

On-disk Structures
- appendonlyfile.metadata
- appendonlyfile.data
- redis_database.block_0
- redis_database.metadata
- redis_database.userdata

Global Effect

<table>
<thead>
<tr>
<th>Corrupt</th>
<th>Read I/O Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td></td>
</tr>
</tbody>
</table>

L | F | L | F

Global Effect
- Unavailability
- Reduced
- Redundancy
- Corruption

Local Behavior

- Crash
- No Detection/No Recovery
Redis: Behavior Analysis

Read Workload

Local Behavior

Corrupt I/O Error

On-disk Structures
appendonlyfile.metadata
appendonlyfile.data

No checksums to detect corruption
Leader returns corrupted data on queries
Corruption propagation to followers

Global Effect

Corrupt I/O Error

Global Effect

L Leader
F Follower

Local Behavior

Crash
No Detection/
No Recovery

Unavailability
Reduced
Redundancy
Corruption
Redis: Behavior Analysis

Read Workload

Local Behavior

<table>
<thead>
<tr>
<th>Corrupt</th>
<th>Read I/O Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>X X X X</td>
<td>X X X X</td>
</tr>
<tr>
<td>⊗</td>
<td>⊗</td>
</tr>
</tbody>
</table>

On-disk Structures

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

Global Effect

<table>
<thead>
<tr>
<th>Corrupt</th>
<th>Read I/O Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>L L L L</td>
<td>L L L L</td>
</tr>
</tbody>
</table>

L Leader
F Follower

Global Effect

Unavailability
Reduced Redundancy
Corruption

Local Behavior

Crash
No Detection/No Recovery
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

Local Behavior:
- Crash
- No Detection/
- No Recovery

Global Effect:
- Unavailability
- Reduced
- Redundancy
- Corruption
## Redis: Behavior Analysis

### Read Workload

<table>
<thead>
<tr>
<th>Local Behavior</th>
<th>Read I/O Error</th>
<th>On-disk Structures</th>
</tr>
</thead>
<tbody>
<tr>
<td>Corrupt</td>
<td>X</td>
<td>appendonlyfile.metadata</td>
</tr>
<tr>
<td></td>
<td>X</td>
<td>appendonlyfile.data</td>
</tr>
<tr>
<td></td>
<td></td>
<td>redis_database.block_0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>redis_database.metadata</td>
</tr>
<tr>
<td></td>
<td></td>
<td>redis_database.userdata</td>
</tr>
</tbody>
</table>

### Global Effect

<table>
<thead>
<tr>
<th>Global Effect</th>
<th>Read I/O Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Corrupt</td>
<td>L</td>
</tr>
<tr>
<td></td>
<td>F</td>
</tr>
</tbody>
</table>

### Local Behavior

- **Leader (L)**: Crash
- **Follower (F)**: No Detection/No Recovery

### Global Effect

- **Unavailability**
- **Reduced Redundancy**
- **Corruption**
Redis: Behavior Analysis

Read Workload

Local Behavior

Corrupt

Read I/O Error

On-disk Structures

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

Global Effect

Corrupt

Read I/O Error

Global Effect

Crash

No Detection/
No Recovery

Unavailability
Reduced
Redundancy
Corruption
Correct
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

Global Effect

Local Behavior

Crash

No Detection/
No Recovery

Unavailability
Reduced
Redundancy
Corruption
Correct
### Redis: Behavior Analysis

**Read Workload**

**Local Behavior**

<table>
<thead>
<tr>
<th>Corrupt</th>
<th>Read I/O Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>φ</td>
<td>φ</td>
</tr>
<tr>
<td></td>
<td>X</td>
</tr>
<tr>
<td>φ</td>
<td>φ</td>
</tr>
<tr>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

**On-disk Structures**

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

**Global Effect**

<table>
<thead>
<tr>
<th>Corrupt</th>
<th>Read I/O Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

**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

Redis Read

Kafka Read

Kafka Write

Cassandra Read

ZooKeeper Write

RethinkDB Read
# Redundancy Does not Provide Fault Tolerance

<table>
<thead>
<tr>
<th>Redis Read</th>
<th>Kafka Read</th>
<th>Kafka Write</th>
<th>Cassandra Read</th>
</tr>
</thead>
<tbody>
<tr>
<td>Corrupt</td>
<td>Corrupt</td>
<td>Corrupt</td>
<td>Corrupt</td>
</tr>
<tr>
<td>Read Error</td>
<td>Read Error</td>
<td>Read Error</td>
<td>Read Error</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ZooKeeper Write</th>
<th>RethinkDB Read</th>
</tr>
</thead>
<tbody>
<tr>
<td>Write Error</td>
<td>Corrupt</td>
</tr>
</tbody>
</table>
Redundancy Does not Provide Fault Tolerance

Redis Read
- Corrupt
- Read Error
- aof.metadata
- aof.data
- rdb.metadata
- rdb.userdata

Kafka Read
- Corrupt
- Read Error
- log.header
- log.other
- replication checkpoint

Kafka Write
- Corrupt
- Read Error

Cassandra Read
- Corrupt
- Read Error
- sstable.block0
- sstable.metadata
- sstable.userdata
- sstable.index

ZooKeeper Write
- Write Error
- txn_head
- log.tail

RethinkDB Read
- Corrupt
- db.txn_head
- db.txn_body
- db.txn_tail
- db.metablock
Redundancy Does not Provide Fault Tolerance

Redis Read
- Corrupt: aof.metadata, aof.data, rdb.metadata, rdb.userdata
- Read Error: L F L F

ZooKeeper Write
- Write Error: txn_head, log.tail
- Corrupt: L F

Kafka Read
- Corrupt: aof.metadata, aof.data, rdb.metadata, rdb.userdata
- Read Error: L F L F
- log.header, log.other, replication, checkpoint

Kafka Write
- Corrupt: log.header, log.other, replication, checkpoint
- Read Error: L F L F

RethinkDB Read
- Corrupt: db.txn_head, db.txn_body, db.txn_tail, db.metablock
- Read Error: L F

Cassandra Read
- Corrupt: sstable.block0, sstable.metadata, sstable.userdata, sstable.index
- Read Error: L F L F
Redundancy Does not Provide Fault Tolerance

Redis Read
- Corrupt: aof.metadata, aof.data, rdb.metadata, rdb.userdata
- Read Error

Kafka Read
- Corrupt: log.header, log.other, replication checkpoint

Kafka Write
- Corrupt: log.header, log.other, replication checkpoint

Cassandra Read
- Corrupt: sstable.block0, sstable.metadata, sstable.userdata, sstable.index

ZooKeeper Write
- Write Error: txn_head, log.tail

RethinkDB Read
- Corrupt: db.txn_head, db.txn_body, db.txn_tail, db.metablock

Corruption
Redundancy Does not Provide Fault Tolerance

Redis Read
- Corrupt: aof.metadata, aof.data, rdb.metadata, rdb.userdata
- Read Error: L, F, L, F

Kafka Read
- Corrupt: aof.metadata, aof.data
- Read Error: L, F, L, F

Kafka Write
- Corrupt: log.header, log.other, replication, checkpoint
- Read Error: L, F, L, F

Cassandra Read
- Corrupt: sstable.block0, sstable.metadata, sstable.userdata, sstable.index
- Read Error: L, F

ZooKeeper Write
- Write Error: txn_head, log.tail
- L, F

RethinkDB Read
- Corrupt: db.txn_head, db.txn_body, db.txn_tail, db.metablock
- Data Loss: L, F

Corruption

Data Loss
Redundancy Does not Provide Fault Tolerance

Redis Read
- Corrupt
- Read Error
- aof.metadata
- aof.data
- rdb.metadata
- rdb.userdata

Kafka Read
- Corrupt
- Read Error
- log.header
- log.other
- replication
- checkpoint

Kafka Write
- Corrupt
- Read Error

Cassandra Read
- Corrupt
- Read Error
- sstable.block0
- sstable.metadata
- sstable.userdata
- sstable.index

ZooKeeper Write
- Write Error
- txn_head
- log.tail

RethinkDB Read
- Corrupt
- db.txn_head
- db.txn_body
- db.txn_tail
- db.metablock

Corruption
Data Loss
Write Unavailability
Redundancy Does not Provide Fault Tolerance

Redis Read
- Corrupt
- Read Error
- aof.metadata
- aof.data
- rdb.metadata
- rdb.userdata

Kafka Read
- Corrupt
- Read Error
- log.header
- log.other
- replication checkpoint

Kafka Write
- Corrupt
- Read Error
- log.header
- log.other
- replication checkpoint

Cassandra Read
- Corrupt
- Read Error
- sstable.block0
- sstable.metadata
- sstable.userdata
- sstable.index

ZooKeeper Write
- Write Error
- txn_head
- log.tail

RethinkDB Read
- Corrupt
- db.txn_head
- db.txn_body
- db.txn_tail
- db.metablock

Unavailability
- Corruption
- Data Loss
- Write Unavailability
## Redundancy Does not Provide Fault Tolerance

### Redis Read
- **Corrupt**: aof.metadata, aof.data, rdb.metadata, rdb.userdata
- **Read Error**: L, F, L, F

### Kafka Read
- **Corrupt**: log.header, log.other, replication, checkpoint
- **Read Error**: L, F, L, F

### Kafka Write
- **Corrupt**: log.header, log.other, replication, checkpoint
- **Read Error**: L, F, L, F

### Cassandra Read
- **Corrupt**: sstable.block0, sstable.metadata, sstableuserdata, sstable.index
- **Read Error**: L, F, L, F

### ZooKeeper Write
- **Write Error**: txn_head, log.tail
- **L, F**

### RethinkDB Read
- **Corrupt**: db.txn_head, db.txn_body, db.txn_tail, db.metablock
- **Read Error**: L, F, L, F

### RethinkDB Write
- **Corruption**: L, F, L, F
- **Data Loss**: L, F, L, F
- **Write Unavailability**: L, F, L, F

### Unavailability
- **Query Failure**: L, F, L, F

---

20
Redundancy Does not Provide Fault Tolerance

Redis Read
- Corrupt
- Read Error

ZooKeeper Write
- Write Error

Kafka Read
- Corrupt
- Read Error

Kafka Write
- Corrupt
- Read Error

RethinkDB Read
- Corrupt

Cassandra Read
- Corrupt
- Read Error

Corruption
Data Loss
Write Unavailability
Unavailability
Query Failure
Reduced Redundancy
Redundancy Does not Provide Fault Tolerance

Redis Read
- Read Error
  - aof.metadata
  - aof.data
  - rdb.metadata
  - rdb.userdata

Kafka Read
- Read Error
  - log.header
  - log.other
  - replication
  - checkpoint

Kafka Write
- Read Error

Cassandra Read
- Read Error
  - sstable.block0
  - sstable.metadata
  - sstable.userdata
  - sstable.index

ZooKeeper Write
- Write Error
  - txn_head
  - log.tail

RethinkDB Read
- Corrupt
  - db.txn_head
  - db.txn_body
  - db.txn_tail
  - db.metablock

- Corruption
- Data Loss
- Write Unavailability

- Unavailability
- Query Failure
- Reduced Redundancy

Harmful global effects despite redundancy
Redundancy Does not Provide Fault Tolerance

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 $\rightarrow$ global silent corruption
Faults are Often Undetected Locally

Undetected faults may lead to harmful global effect
- Locally undetected corruption $\rightarrow$ 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

Cassandra: Locally Undetected Fault

Client | Replica 1 | Other Replicas
Faults are Often Undetected Locally

Undetected faults may lead to harmful global effect
- Locally undetected corruption $\rightarrow$ global silent corruption

**Cassandra: Locally Undetected Fault**

<table>
<thead>
<tr>
<th>Client</th>
<th>Replica 1</th>
<th>Other Replicas</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>sstable</td>
<td></td>
</tr>
<tr>
<td></td>
<td>key</td>
<td>value</td>
</tr>
<tr>
<td></td>
<td>sstable</td>
<td>key</td>
</tr>
</tbody>
</table>
Faults are Often Undetected Locally

Undetected faults may lead to harmful global effect
- Locally undetected corruption $\rightarrow$ global silent corruption

**Cassandra: Locally Undetected Fault**

<table>
<thead>
<tr>
<th>Client</th>
<th>Replica 1</th>
<th>Other Replicas</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>sstable.userdata corrupted</td>
<td></td>
</tr>
<tr>
<td></td>
<td>key</td>
<td>value</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>sstable</th>
<th>key</th>
<th>value</th>
<th>key</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Faults are Often Undetected Locally

Undetected faults may lead to harmful global effect
- Locally undetected corruption $\rightarrow$ global silent corruption

**Cassandra: Locally Undetected Fault**

<table>
<thead>
<tr>
<th>Client</th>
<th>Replica 1</th>
<th>Other Replicas</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>sstable.userdata corrupted</td>
<td>sstable</td>
</tr>
<tr>
<td>key</td>
<td>value</td>
<td>key</td>
</tr>
<tr>
<td>sstable compression = off</td>
<td>No checksums to detect corruption</td>
<td></td>
</tr>
</tbody>
</table>
Faults are Often Undetected Locally

Undetected faults may lead to harmful global effect
- Locally undetected corruption $\rightarrow$ global silent corruption

**Cassandra: Locally Undetected Fault**

<table>
<thead>
<tr>
<th>Client</th>
<th>Replica 1</th>
<th>Other Replicas</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ (R=1)</td>
<td>sstable.userdata corrupted</td>
<td>sstable</td>
</tr>
<tr>
<td></td>
<td>sstable compression = off</td>
<td>key</td>
</tr>
<tr>
<td></td>
<td>No checksums to detect corruption</td>
<td>key</td>
</tr>
</tbody>
</table>
Faults are Often Undetected Locally

Undetected faults may lead to harmful global effect
- Locally undetected corruption $\rightarrow$ global silent corruption

**Cassandra: Locally Undetected Fault**

Client

Replica 1

- sstable.userdata corrupted
- sstable.compression = off
- No checksums to detect corruption

Other Replicas

- sstable

<table>
<thead>
<tr>
<th>key</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Faults are Often Undetected Locally

Undetected faults may lead to harmful global effect
- Locally undetected corruption $\rightarrow$ global silent corruption

Cassandra: Locally Undetected Fault

Client

READ (R=1)

CORRUPT

Replica 1

sstable.userdata corrupted
sstable compression = off

No checksums to detect corruption

Other Replicas

sstable

key | value

Read Repair
Faults are Often Undetected Locally

Undetected faults may lead to harmful global effect
- Locally undetected corruption → global silent corruption

**Cassandra: Locally Undetected Fault**

- **Client**
  - READ (R=1)
- **Replica 1**
  - sstable.userdata corrupted
  - sstable compression = off
  - No checksums to detect corruption
- **Other Replicas**
  - sstable
  - Read Repair
Faults are Often Undetected Locally

Undetected faults may lead to harmful global effect
- Locally undetected corruption $\rightarrow$ global silent corruption

Cassandra: Locally Undetected Fault

Client

READ (R=1)

CORRUPT

Replica 1

sstable.userdata corrupted

sstable compression = off

No checksums to detect corruption

Read Repair

Other Replicas

sstable

CORRUPT

READ

key | value

key | value

key | value
Faults are Often Undetected Locally

Undetected faults may lead to harmful global effect
- Locally undetected corruption → global silent corruption

Cassandra: Locally Undetected Fault

Client | Replica 1 | Other Replicas
---|---|---
READ (R=1) | sstable.userdata corrupted | sstable
CORRUPT | sstable.compression = off |
| | No checksums to detect 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

collections.header
collections.metadata
collections.data
index
journal.header
journal.other
storage_bson
wiredtiger_wt

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

collections.header
collections.metadata
collections.data
index
journal.header
journal.other
storage_bson
wiredtiger_wt

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**
- collections.header
- collections.metadata
- collections.data
- index
- journal.header
- journal.other
- storage_bson
- wiredtiger_wt

**ZooKeeper**
- epoch
- epoch_tmp
- myid
- log.transaction_head
- log.transaction_body
- log.transaction_tail
- log.remaining
- log.tail

**Crash**
- L Leader
- F Follower
Many systems that reliably detect fault simply crash on encountering faults

Block Corruption during Read Workloads

MongoDB
- collections.header
- collections.metadata
- collections.data
- index
- journal.header
- journal.other
- storage_bson
- wiredtiger_wt

ZooKeeper
- epoch
- epoch_tmp
- myid
- log.transaction_head
- log.transaction_body
- log.transaction_tail
- log.remaining
- log.tail
Many systems that reliably detect fault **simply crash** on encountering faults.

Block Corruption during Read Workloads

**MongoDB**

- collections.header
- collections.metadata
- collections.data
- index
- journal.header
- journal.other
- storage_bson
- wiredtiger_wt

**ZooKeeper**

- epoch
- epoch_tmp
- myid
- log.transaction_head
- log.transaction_body
- log.transaction_tail
- log.remaining
- log.tail

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

MongoDB
- collections.header
- collections.metadata
- collections.data
- index
- journal.header
- journal.other
- storage_bson
- wiredtiger_wt

ZooKeeper
- epoch
- epoch_tmp
- myid
- log.transaction_head
- log.transaction_body
- log.transaction_tail
- log.remaining
- log.tail

Crash
- L  Leader
- F  Follower

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**

<table>
<thead>
<tr>
<th></th>
<th>MongoDB</th>
<th>ZooKeeper</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>collections.header</td>
<td>epoch</td>
</tr>
<tr>
<td></td>
<td>collections.metadata</td>
<td>epoch_tmp</td>
</tr>
<tr>
<td></td>
<td>collections.data</td>
<td>myid</td>
</tr>
<tr>
<td></td>
<td>index</td>
<td>log.transaction_head</td>
</tr>
<tr>
<td></td>
<td>journal.header</td>
<td>log.transaction_body</td>
</tr>
<tr>
<td></td>
<td>journal.other</td>
<td>log.transaction_tail</td>
</tr>
<tr>
<td></td>
<td>storage_bson</td>
<td>log.remaining</td>
</tr>
<tr>
<td></td>
<td>wiredtiger_wt</td>
<td>log.tail</td>
</tr>
<tr>
<td>L</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**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

0
Crash and Corruption Handling are Entangled

Kafka Message Log

0

data
Crash and Corruption Handling are Entangled

Kafka Message Log

checksum  data
Crash and Corruption Handling are Entangled

Kafka Message Log

checksum data
Crash and Corruption Handling are Entangled

Kafka Message Log

Append(log, entry 2)
Crash and Corruption Handling are Entangled

Kafka Message Log

Append("Entry 2")

checksum
data

0 1
Crash and Corruption Handling are Entangled
Crash and Corruption Handling are Entangled

Kafka Message Log

Checksum mismatch

Append("Entry 2")

checksum 0 1 2 data
Crash and Corruption Handling are Entangled

Kafka Message Log

- Append(log, entry 2)
- Checksum mismatch
- Action: Truncate log at 1
Crash and Corruption Handling are Entangled

Kafka Message Log

Append(“” entry 2)

Checksum mismatch

Action: Truncate log at 1
Crash and Corruption Handling are Entangled

Kafka Message Log

Checksum mismatch
Action: Truncate log at 1
Lose uncommitted data
Crash and Corruption Handling are Entangled

Kafka Message Log

Checksum mismatch
Action: Truncate log at 1
Lose uncommitted data
Crash and Corruption Handling are Entangled

Kafka Message Log

Checksum mismatch
Action: Truncate log at 1
Lose uncommitted data
Crash and Corruption Handling are Entangled

Kafka Message Log

Checksum mismatch
Action: Truncate log at 1
Lose uncommitted data
Crash and Corruption Handling are Entangled

Kafka Message Log

Checksum mismatch
Action: Truncate log at 1
Lose uncommitted data

Disk corruption
Crash and Corruption Handling are Entangled

Kafka Message Log

checksum | data
---|---
0 | 1 | 2

Checksum mismatch
Action: Truncate log at 1
Lose uncommitted data

Disk corruption
Checksum mismatch
Crash and Corruption Handling are Entangled

Kafka Message Log

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

Kafka Message Log

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

Kafka Message Log

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

Kafka Message Log

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

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

0 1 2

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

Node 1

Other Nodes

0 1 2
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

Node 1

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

0 1 2

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

Set of in-sync replicas
Node 1 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
Node 1 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

0 1 2

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

Set of in-sync replicas
Node 1 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

0 1 2

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

Client
READ
message:0
[Silent data loss]

Leader

0 1 2

Followers

0 1 2

Set of in-sync replicas

Node 1 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

<table>
<thead>
<tr>
<th></th>
<th>Local Behavior</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>1</td>
<td></td>
<td>2</td>
</tr>
</tbody>
</table>

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

Client
READ
message:0
[Silent data loss]

Leader

Followers

Truncate upto message 0
Assertion failure
Unsafe Interaction between Local & Global Protocols

Kafka: Message log at Node 1

Set of in-sync replicas
Node1 with truncated log not removed from in-sync replicas
Node 1 elected as leader

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

Local Behavior

0 1 2

Client

READ

message:0
[Silent data loss]

Leaders

0 1 2

WRITE (W=2)

Followers

0 1 2

Truncate upto message 0
Assertion failure
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

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

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
- Read Error
- Corrupt
- aof.metadata
- aof.data
- rdb.metadata
- rdb.userdata

Kafka Read
- Read Error
- Corrupt
- log.header
- log.other
- replication
- checkpoint

Kafka Write
- Read Error
- Corrupt

Cassandra Read
- Read Error
- Corrupt
- sstable.block0
- sstable.metadata
- sstable.userdata
- sstable.index

ZooKeeper Write
- Write Error
- Corrupt
- txn_head
- log.tail

RethinkDB Read
- Corrupt
- db.txn.head
- db.txn.body
- db.txn.tail
- db.metablock

- Corruption
- Data Loss
- Write Unavailability

- Unavailability
- Query Failure
- Reduced Redundancy
Why does Redundancy Not Imply Fault Tolerance?

Redis Read
- Corrupt
- Read Error

Kafka Read
- Corrupt
- Read Error

Kafka Write
- Corrupt
- Read Error

ZooKeeper
- Write
- Write Error

RethinkDB
- Read
- Corrupt

Cassandra Read
- Corrupt
- Read Error
## Why does Redundancy Not Imply Fault Tolerance?

<table>
<thead>
<tr>
<th>Redundancy</th>
<th>Fault Tolerance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

### Example 1: Redis Read
- **Corrupt**: If a node is corrupt, it cannot be read.
- **Read Error**: If a node returns an error, it cannot be read.

### Example 2: Kafka Read
- **Corrupt**: If a node is corrupt, it cannot be read.
- **Read Error**: If a node returns an error, it cannot be read.

### Example 3: Kafka Write
- **Corrupt**: If a node is corrupt, the write cannot be performed.
- **Read Error**: If a node returns an error, the write cannot be performed.

### Example 4: ZooKeeper Write
- **Corrupt**: If a node is corrupt, the write cannot be performed.
- **Read Error**: If a node returns an error, the write cannot be performed.

### Example 5: RethinkDB Read
- **Corrupt**: If a node is corrupt, it cannot be read.
- **Read Error**: If a node returns an error, it cannot be read.

### Example 6: Cassandra Read
- **Corrupt**: If a node is corrupt, it cannot be read.
- **Read Error**: If a node returns an error, it cannot be read.
Why does Redundancy Not Imply Fault Tolerance?

Faults are often locally undetected

The diagram illustrates the failure modes of various systems, including Redis, Kafka, ZooKeeper, RethinkDB, and Cassandra. The table shows the read and write operations with possible corruptions and read errors, highlighting how redundancy does not guarantee fault tolerance due to undetected local faults.
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?

- **Redis Read**
- **Kafka Read**
- **Kafka Write**
- **ZooKeeper Write**
- **RethinkDB Read**
- **Cassandra Read**

**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

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ZooKeeper</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cassandra</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Kafka</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RethinkDB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MongoDB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LogCabin</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CockroachDB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Fundamental Problems: Summary

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Redis</td>
<td>🟥</td>
<td>🟥</td>
<td>🟥</td>
<td>🟥</td>
<td>🟥</td>
</tr>
<tr>
<td>ZooKeeper</td>
<td>🟥</td>
<td>🟥</td>
<td>🟥</td>
<td>🟥</td>
<td>🟥</td>
</tr>
<tr>
<td>Cassandra</td>
<td>🟥</td>
<td>🟥</td>
<td>🟥</td>
<td>🟥</td>
<td>🟥</td>
</tr>
<tr>
<td>Kafka</td>
<td>🟥</td>
<td>🟥</td>
<td>🟥</td>
<td>🟥</td>
<td>🟥</td>
</tr>
<tr>
<td>RethinkDB</td>
<td>🟥</td>
<td>🟥</td>
<td>🟥</td>
<td>🟥</td>
<td>🟥</td>
</tr>
<tr>
<td>MongoDB</td>
<td>🟥</td>
<td>🟥</td>
<td>🟥</td>
<td>🟥</td>
<td>🟥</td>
</tr>
<tr>
<td>LogCabin</td>
<td>🟥</td>
<td>🟥</td>
<td>🟥</td>
<td>🟥</td>
<td>🟥</td>
</tr>
<tr>
<td>CockroachDB</td>
<td>🟥</td>
<td>🟥</td>
<td>🟥</td>
<td>🟥</td>
<td>🟥</td>
</tr>
</tbody>
</table>
## Fundamental Problems: Summary

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Redis</td>
<td><img src="#" alt="Red" /></td>
<td><img src="#" alt="Red" /></td>
<td><img src="#" alt="Red" /></td>
<td><img src="#" alt="Red" /></td>
<td><img src="#" alt="Red" /></td>
</tr>
<tr>
<td>ZooKeeper</td>
<td><img src="#" alt="Red" /></td>
<td><img src="#" alt="Red" /></td>
<td><img src="#" alt="Red" /></td>
<td><img src="#" alt="Red" /></td>
<td><img src="#" alt="Red" /></td>
</tr>
<tr>
<td>Cassandra</td>
<td><img src="#" alt="Red" /></td>
<td><img src="#" alt="Red" /></td>
<td><img src="#" alt="Red" /></td>
<td><img src="#" alt="Red" /></td>
<td><img src="#" alt="Red" /></td>
</tr>
<tr>
<td>Kafka</td>
<td><img src="#" alt="Red" /></td>
<td><img src="#" alt="Red" /></td>
<td><img src="#" alt="Red" /></td>
<td><img src="#" alt="Red" /></td>
<td><img src="#" alt="Red" /></td>
</tr>
<tr>
<td>RethinkDB</td>
<td><img src="#" alt="Red" /></td>
<td><img src="#" alt="Red" /></td>
<td><img src="#" alt="Red" /></td>
<td><img src="#" alt="Red" /></td>
<td><img src="#" alt="Red" /></td>
</tr>
<tr>
<td>MongoDB</td>
<td><img src="#" alt="Red" /></td>
<td><img src="#" alt="Red" /></td>
<td><img src="#" alt="Red" /></td>
<td><img src="#" alt="Red" /></td>
<td><img src="#" alt="Red" /></td>
</tr>
<tr>
<td>LogCabin</td>
<td><img src="#" alt="Red" /></td>
<td><img src="#" alt="Red" /></td>
<td><img src="#" alt="Red" /></td>
<td><img src="#" alt="Red" /></td>
<td><img src="#" alt="Red" /></td>
</tr>
<tr>
<td>CockroachDB</td>
<td><img src="#" alt="Red" /></td>
<td><img src="#" alt="Red" /></td>
<td><img src="#" alt="Red" /></td>
<td><img src="#" alt="Red" /></td>
<td><img src="#" alt="Red" /></td>
</tr>
</tbody>
</table>
## Fundamental Problems: Summary

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Redis</td>
<td>◼</td>
<td>◼</td>
<td>◼</td>
<td>◼</td>
<td>◼</td>
</tr>
<tr>
<td>ZooKeeper</td>
<td>◼</td>
<td>◼</td>
<td>◼</td>
<td>◼</td>
<td>◼</td>
</tr>
<tr>
<td>Cassandra</td>
<td>◼</td>
<td>◼</td>
<td>◼</td>
<td>◼</td>
<td>◼</td>
</tr>
<tr>
<td>Kafka</td>
<td>◼</td>
<td>◼</td>
<td>◼</td>
<td>◼</td>
<td>◼</td>
</tr>
<tr>
<td>RethinkDB</td>
<td>◼</td>
<td>◼</td>
<td>◼</td>
<td>◼</td>
<td>◼</td>
</tr>
<tr>
<td>MongoDB</td>
<td>◼</td>
<td>◼</td>
<td>◼</td>
<td>◼</td>
<td>◼</td>
</tr>
<tr>
<td>LogCabin</td>
<td>◼</td>
<td>◼</td>
<td>◼</td>
<td>◼</td>
<td>◼</td>
</tr>
<tr>
<td>CockroachDB</td>
<td>◼</td>
<td>◼</td>
<td>◼</td>
<td>◼</td>
<td>◼</td>
</tr>
</tbody>
</table>

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!