Understanding Real-World Timeout Problems in Cloud Server Systems

Ting Dai, Jingzhu He, Xiaohui (Helen) Gu, Shan Lu*

*NC State University   *University of Chicago*
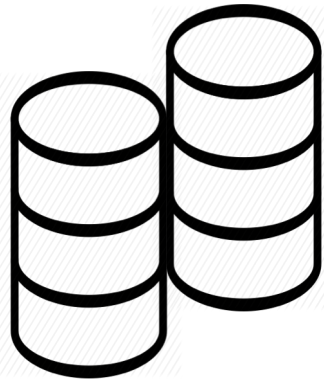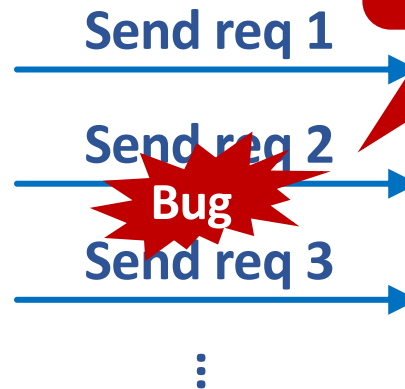
Real-world timeout problems

Amazon DynamoDB service was down for 5 hours.

https://aws.amazon.com/cn/message/5467D2/

No proper limit of retry.

Send req 1

Send req 2

Bug

Send req 3

Overloaded
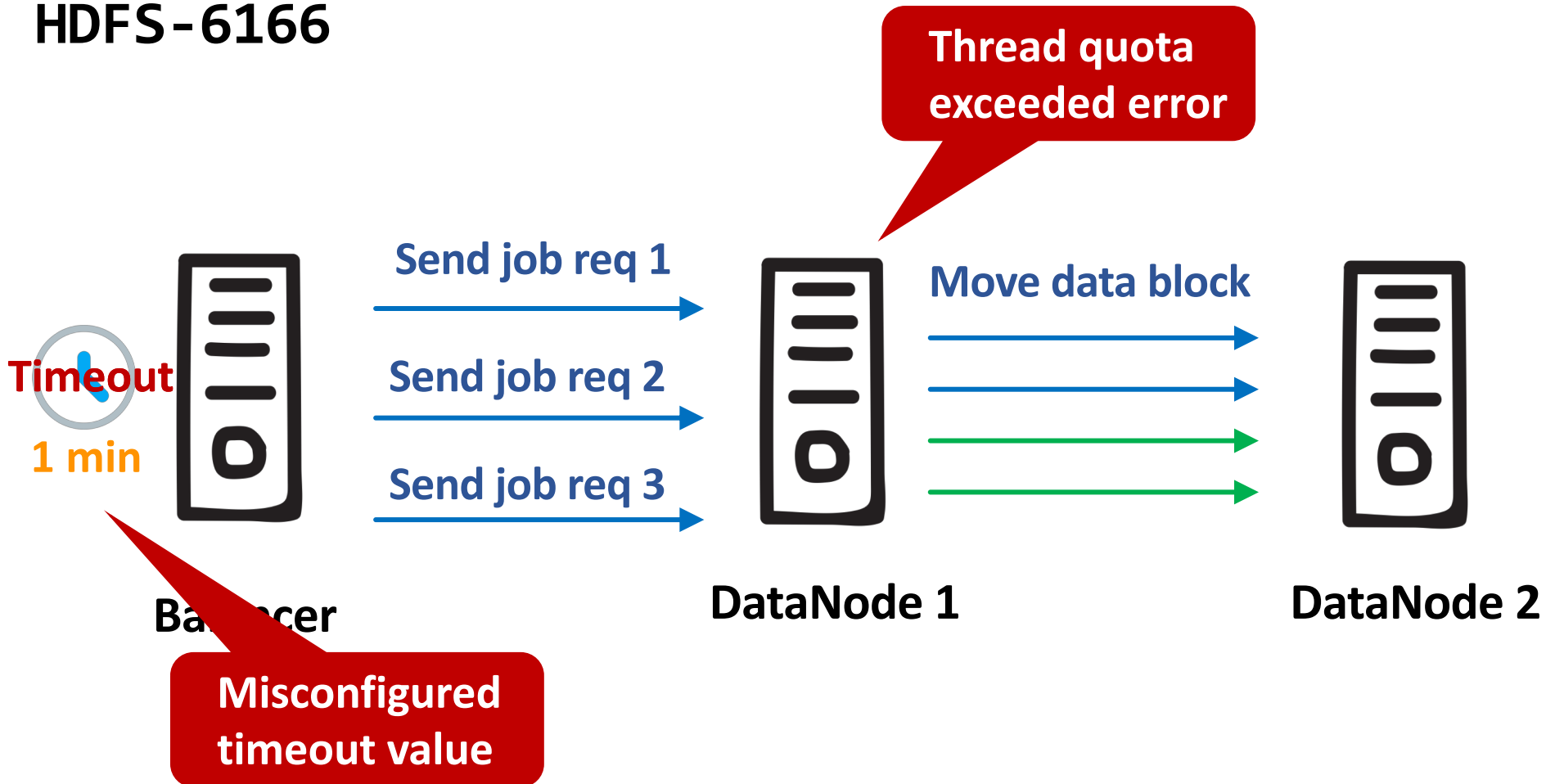
Timeout
Timeout
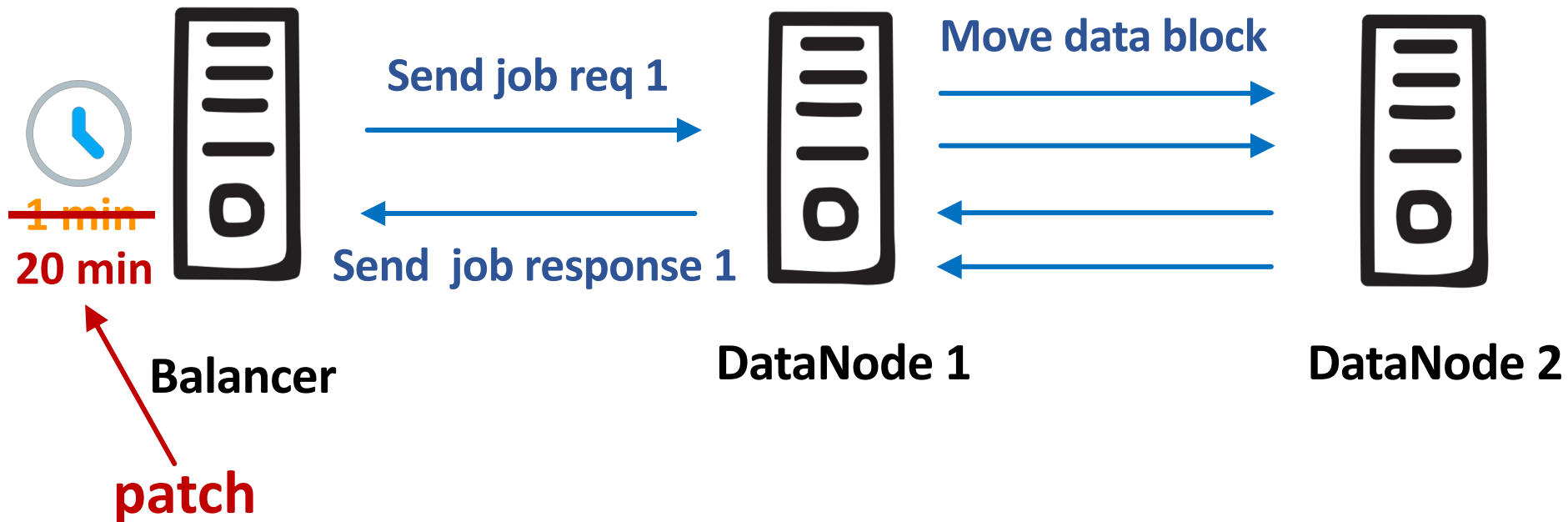**Timeout**

Storage servers

Metadata server

A Motivating Example

A Motivating Example

HDFS-6166

What are timeout bugs?

Timeout bugs happen when the server applications lack proper **configuration** and **handling** of the timeout events.

Why are timeout bugs prevalent?

- Cloud server systems have become increasingly complex.

- Timeout is one of the commonly used mechanisms to handle unexpected failures in distributed computing environments.

Methodology

- We searched timeout bugs in **11** popular cloud server applications from Apache JIRA.

- We extensively studied **156** bugs.

System	# of bugs
Cassandra	17
Flume	13
Hadoop Common	15
Hadoop Mapreduce	15
Hadoop Yarn	4
HDFS	26

System	# of bugs
HBase	28
Phoenix	6
Qpid	20
Spark	4
Zookeeper	8
Total	**156**

Methodology

We classified the **156** timeout bugs in regard to **three** characteristics:

- ❖ root causes
- ❖ impact to systems or applications
- ❖ diagnosability

Root Cause

Legend:
- Misused timeout value
- Missing timeout checking
- Improper handling
- Unnecessary timeout
- Clock drifting

Pie chart values: 47%, 31%, 12%, 5%, 5%
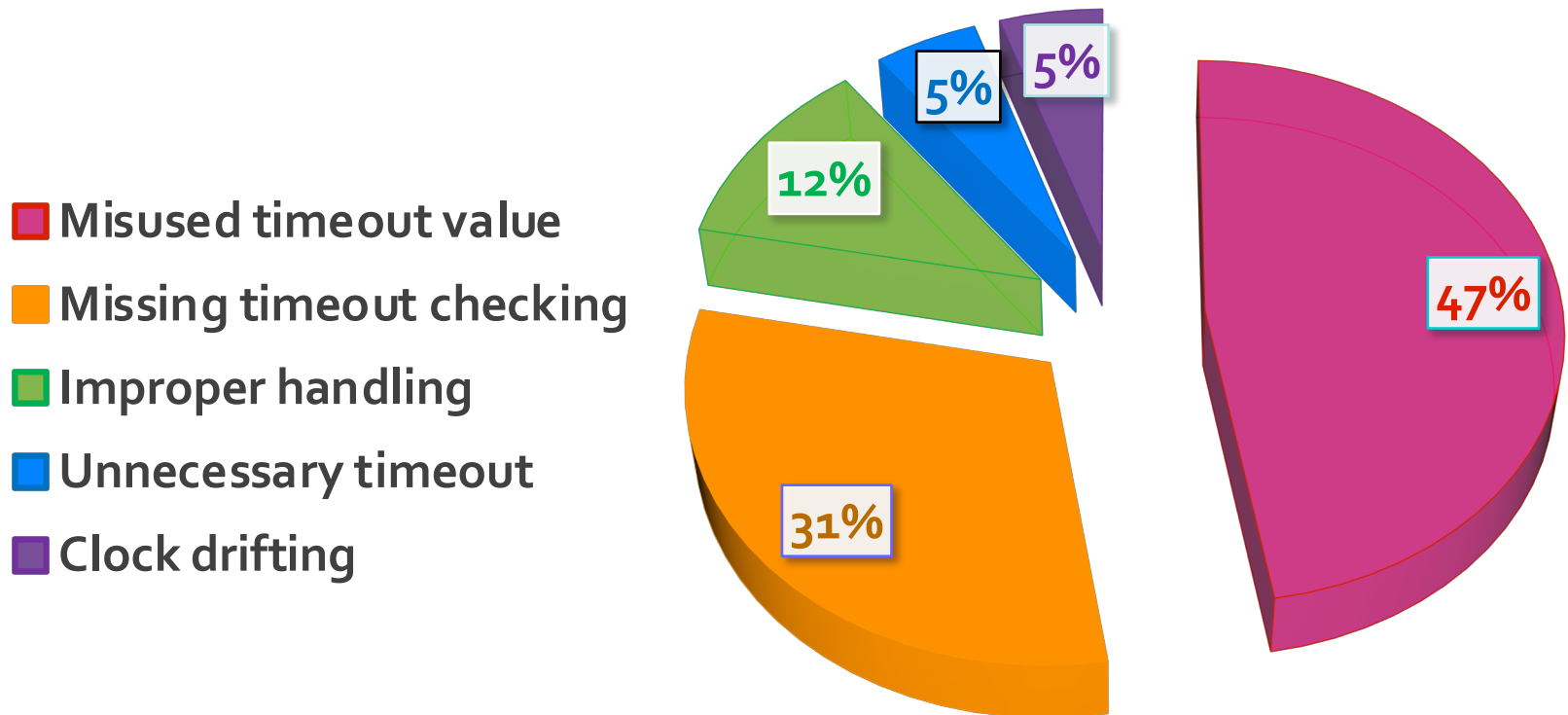
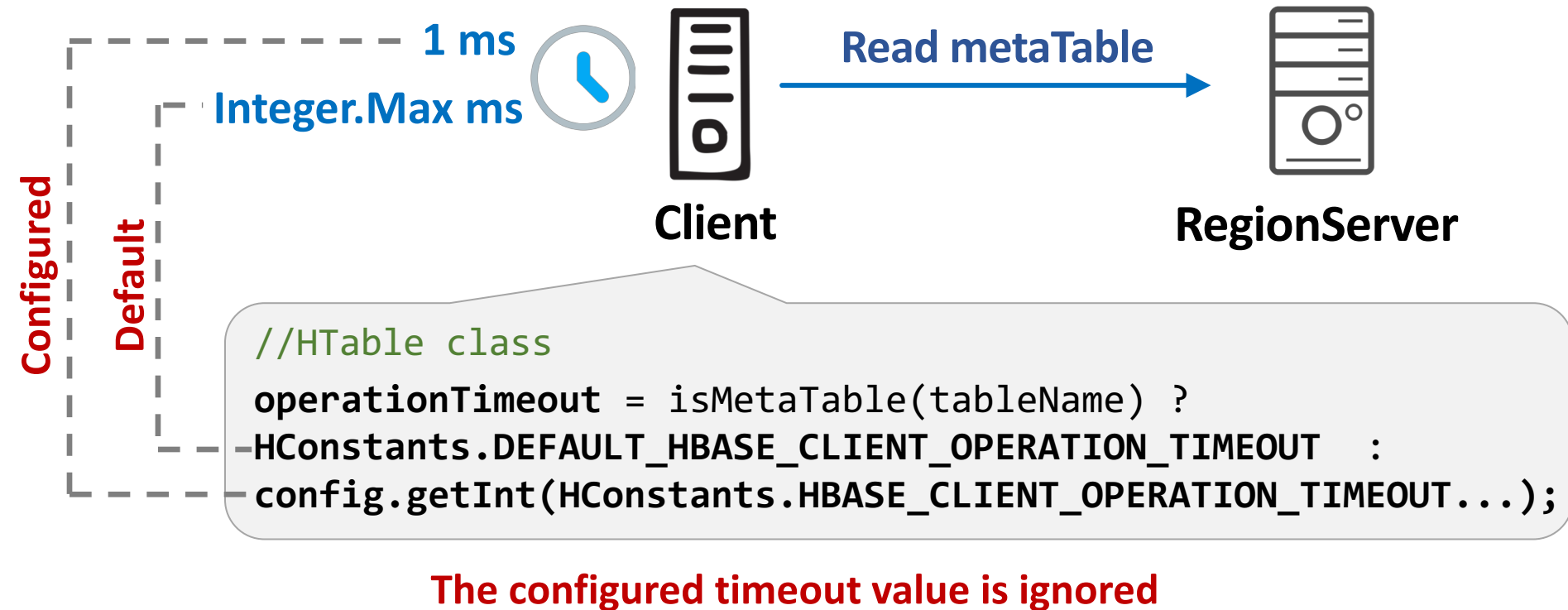**Misused timeout value & Missing timeout checking dominate.**

Root Cause

Misused timeout value (**65** bugs)

❖ Misconfigured timeout value (**38** bugs)

❖ Ignored timeout value (**10** bugs)

❖ Incorrectly reused timeout value (**8** bugs)

❖ Inconsistent timeout value (**4** bugs)

❖ Stale timeout value (**3** bugs)

❖ Improper timeout scope (**2** bugs)

An Ignored Timeout Value Example

HBase-8581

1 ms

Integer.Max ms

Configured

Default

Client

Read metaTable

RegionServer

```
//HTable class
operationTimeout = isMetaTable(tableName) ?
HConstants.DEFAULT_HBASE_CLIENT_OPERATION_TIMEOUT  :
config.getInt(HConstants.HBASE_CLIENT_OPERATION_TIMEOUT...);
```

The configured timeout value is ignored

Observation

Misused timeout value bugs often occur when:

- ❖ lack **extensive testing** on timeout configurations;
- ❖ do not understand the system's **timeout mechanisms**.

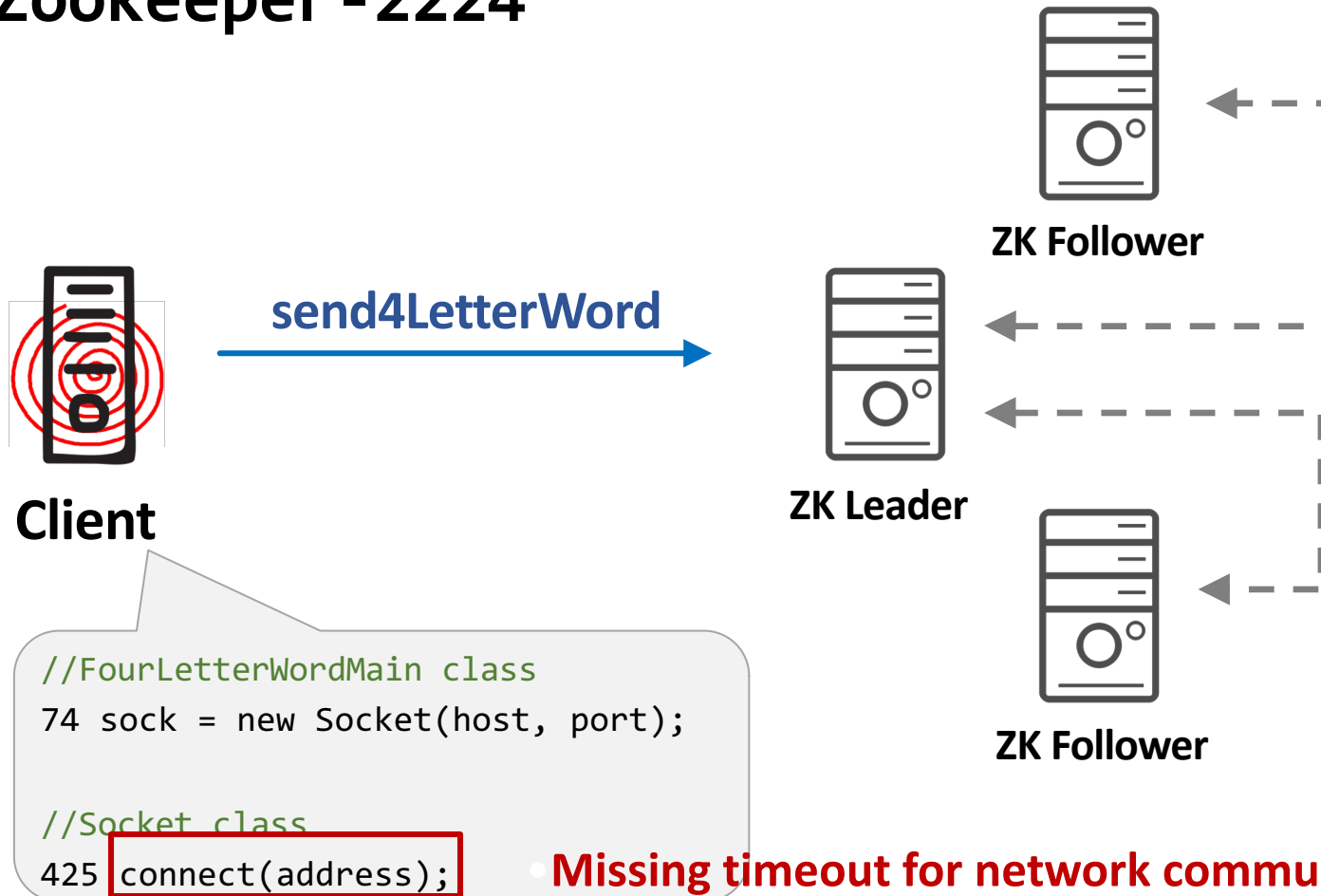**Setting proper timeout value is challenging.**

Root Cause

Missing timeout checking (**42** bugs)

❖ Missing timeout for network communication (**26** bugs)
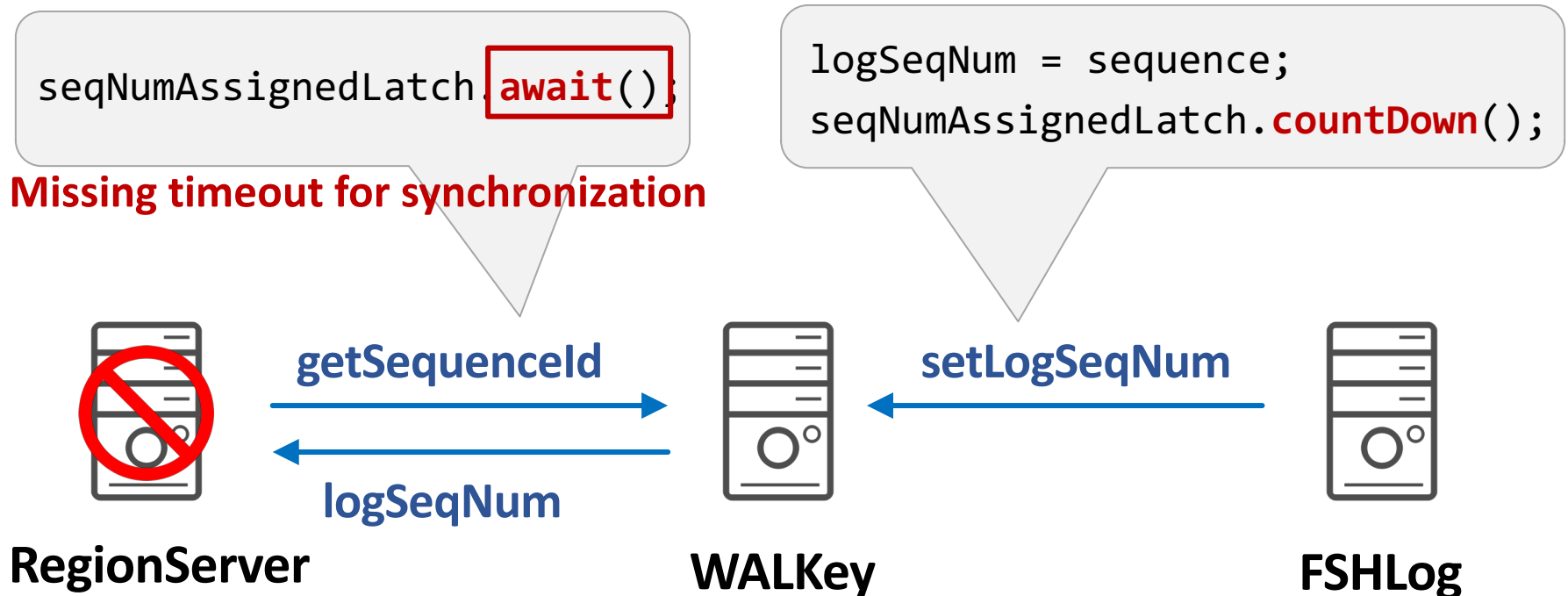
❖ Missing timeout for synchronization (**16** bugs)

A Missing Timeout Example

Zookeeper-2224

send4LetterWord

Client

ZK Follower

ZK Leader

ZK Follower

```
//FourLetterWordMain class
74 sock = new Socket(host, port);

//Socket class
425 connect(address);
```

- **Missing timeout for network communication**

Another Missing Timeout Example

HBase-13971

```
seqNumAssignedLatch.await();
```

Missing timeout for synchronization

```
logSeqNum = sequence;
seqNumAssignedLatch.countDown();
```

getSequenceId

logSeqNum

setLogSeqNum

RegionServer　　　　**WALKey**　　　　**FSHLog**

Observation

Missing timeout bugs often occur when developers do not consider the system's **failover mechanisms**.
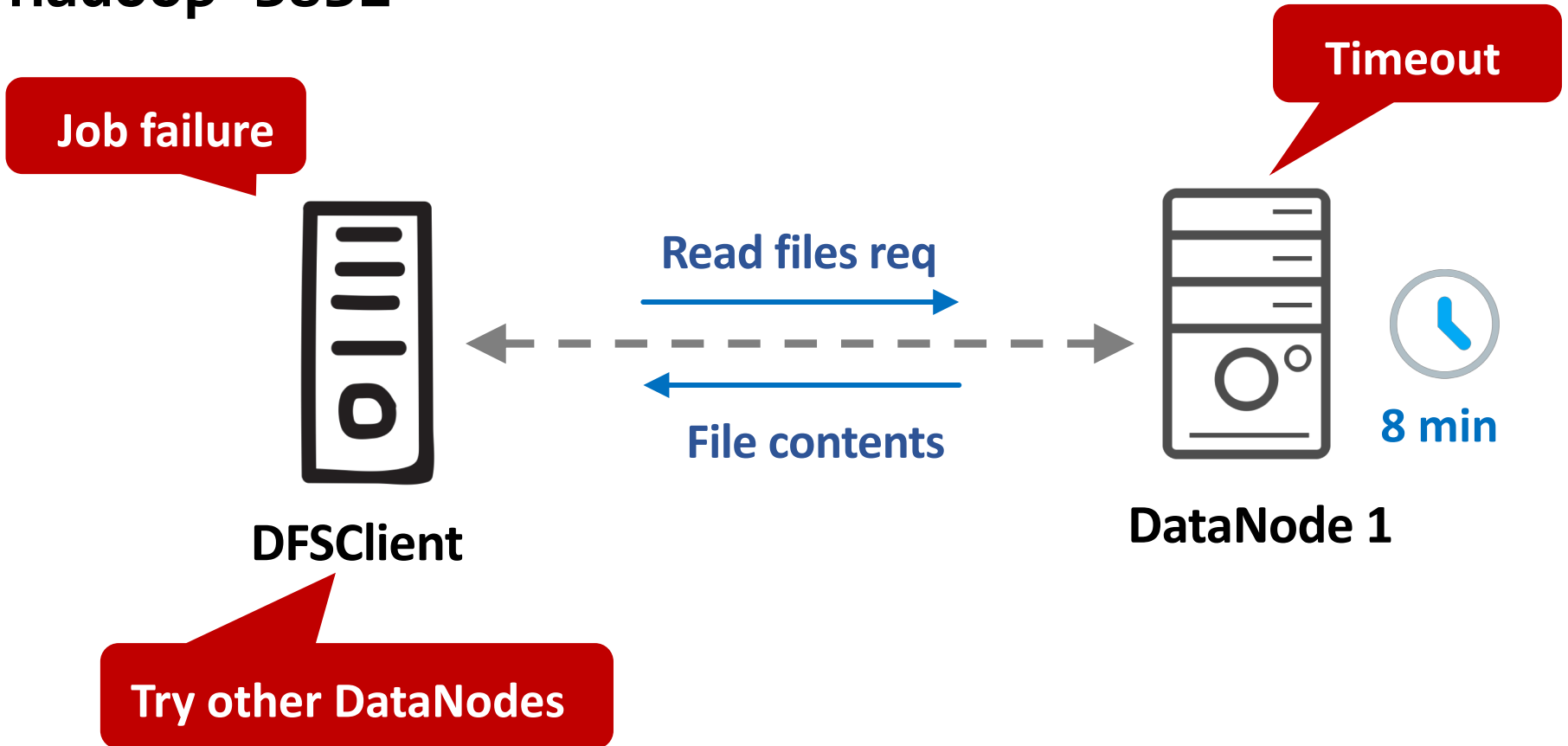
Root Cause

Improper timeout handling (**16** bugs)

- ❖ Insufficient/missing retries (**8** bugs)
- ❖ Excessive retries (**3** bugs)
- ❖ Incorrect retry (**2** bugs)
- ❖ Incomplete abort (**2** bugs)
- ❖ Incorrect abort (**1** bug)

Insufficient/missing retries cause job failure

Hadoop-3831

Observation

It is **challenging** to implement proper timeout handling mechanisms, which requires developers to understand:

❖ the **tradeoffs** between handling schemes (e.g., aborting v.s. retry);

❖ each handling scheme's **impact** to the systems and applications.

Root Cause

Unnecessary timeout protection (**7** bugs)

Those bugs occur when developers mistakenly use timeout retry mechanisms over operations which requires continuous or at-most-once-execution semantics.

Root Cause

Clock drifting (**7** bugs)

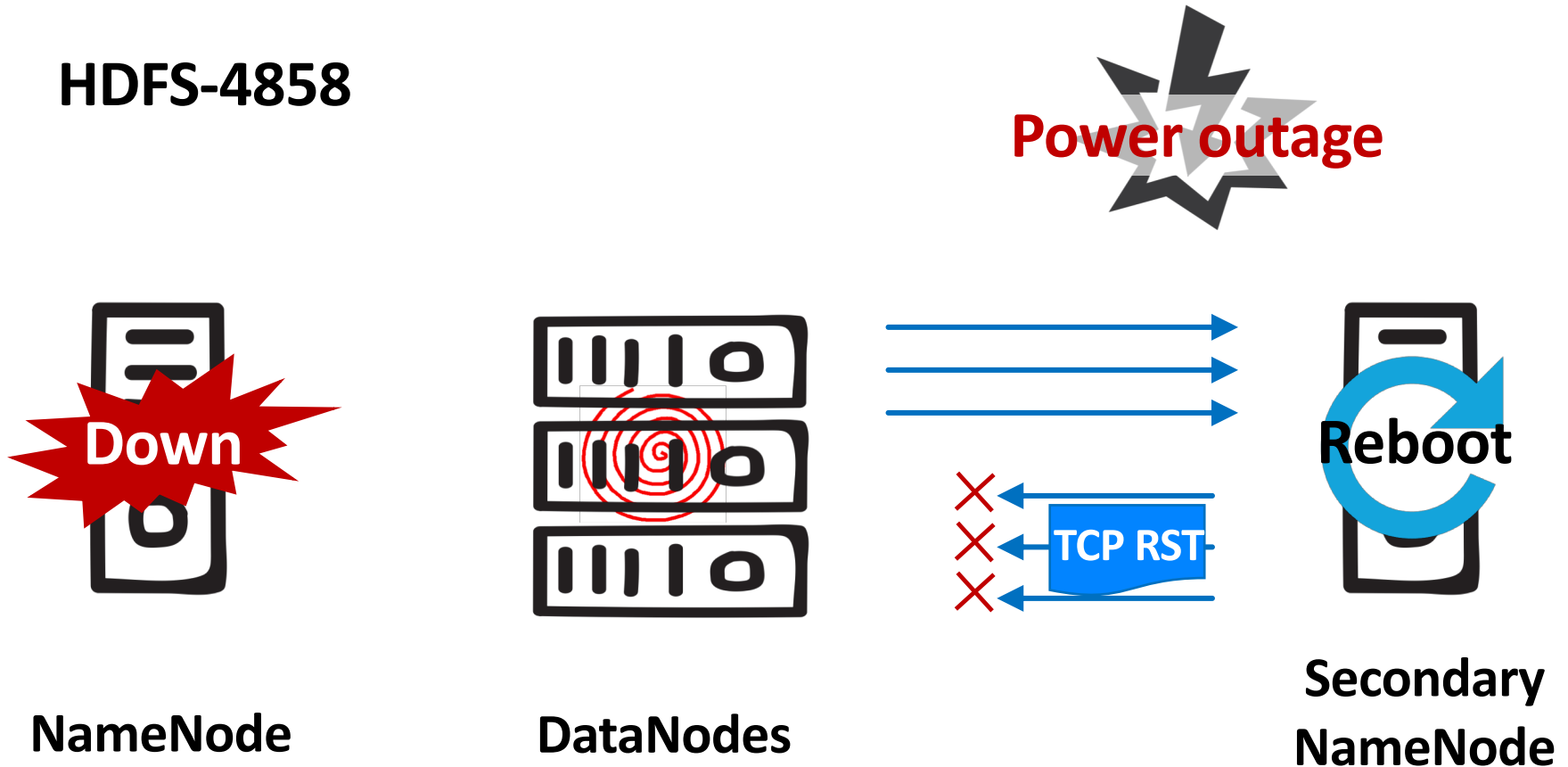**Those bugs occur when the clocks are out-of-synchronization, the elapsed time is miscalculated, which generates a wrong timer value.**
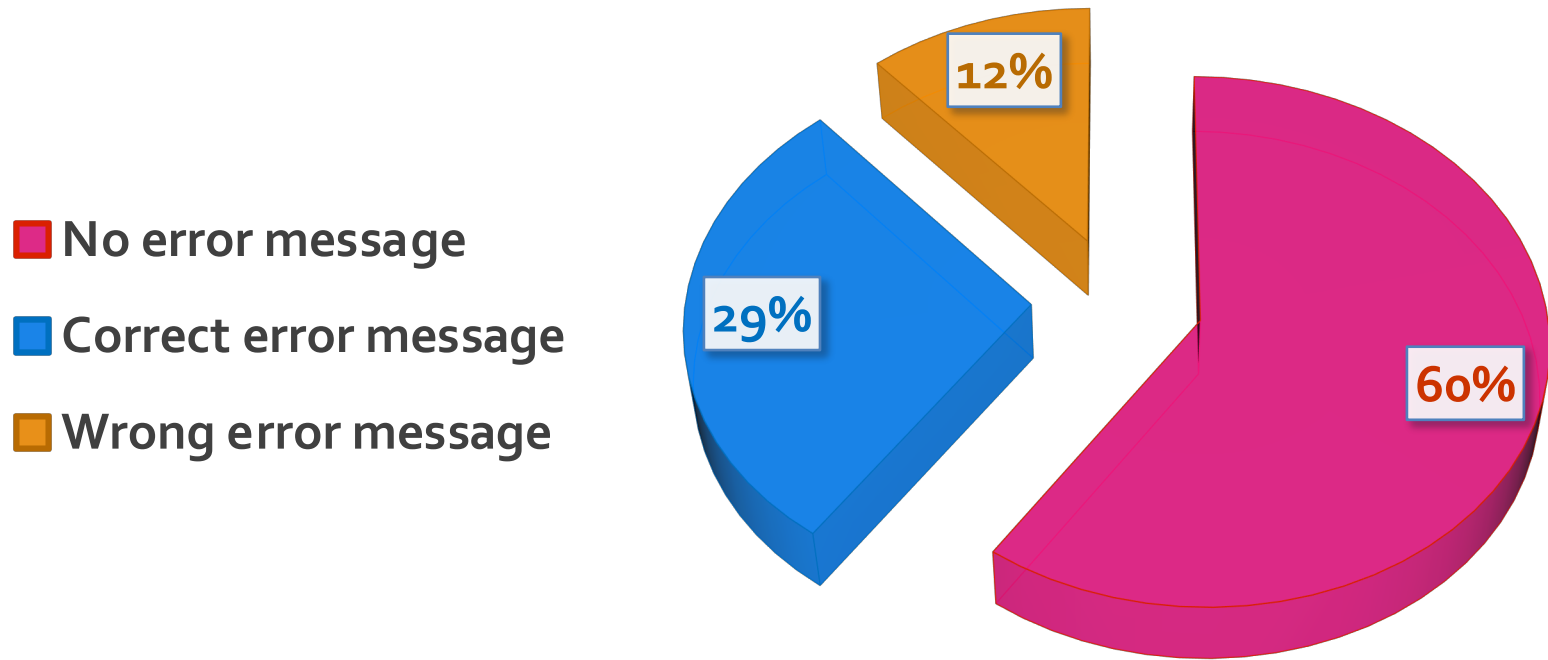
Impact

- 🟥 System unavailability
- 🟧 Job failure

Pie chart: 40%, 33%, 26%, 2%
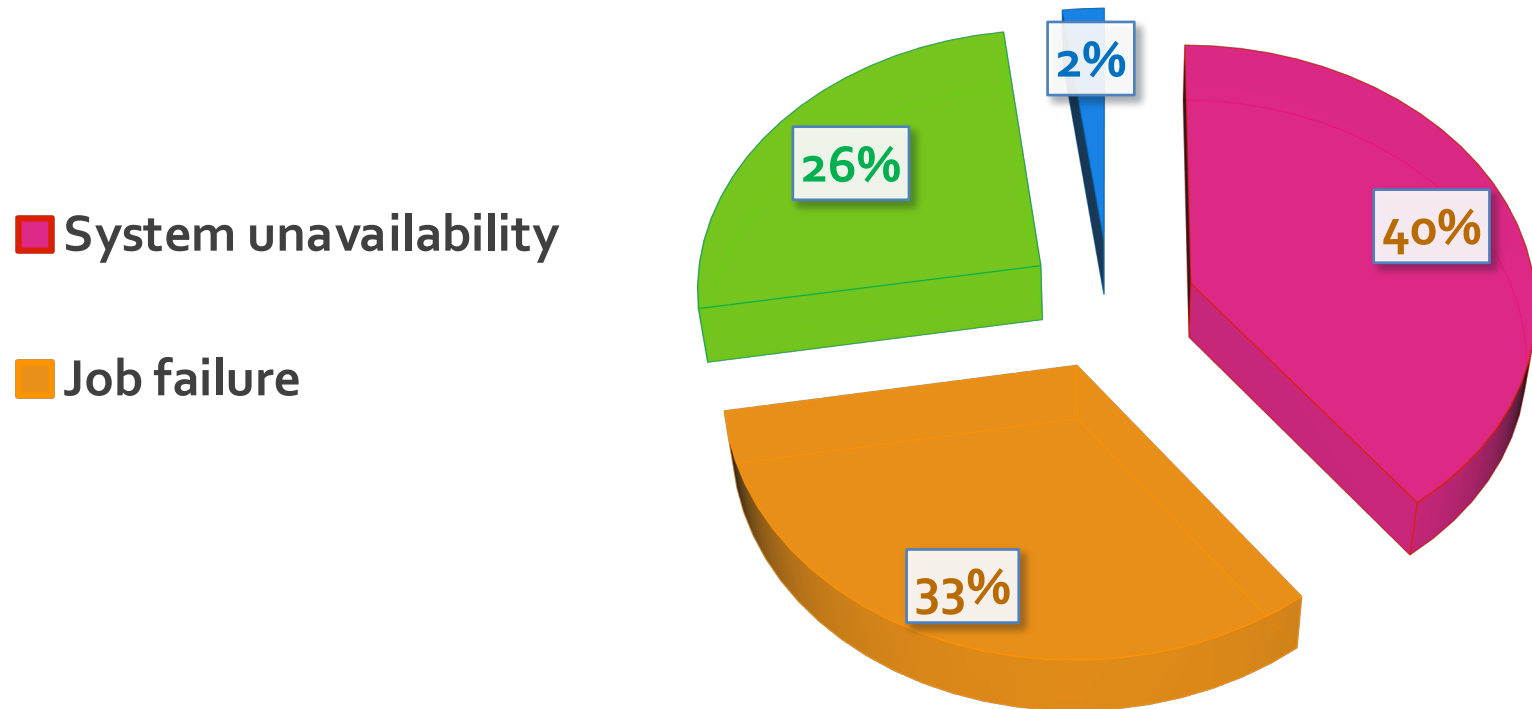
Unavailability caused by missing timeout

HDFS-4858

Power outage

Down

TCP RST

Reboot

NameNode

DataNodes

Secondary NameNode

DataNodes miss timeout. HDFS becomes unavailable.

Diagnosability

- No error message
- Correct error message
- Wrong error message

12%

29%

60%

Only 29 % timeout bugs report the correct error messages.

A Wrong Error Message Example

Cassandra-3651


```
try {
  ...
} catch (TimeoutException e) {
  throw new UnavailableException();
}
```
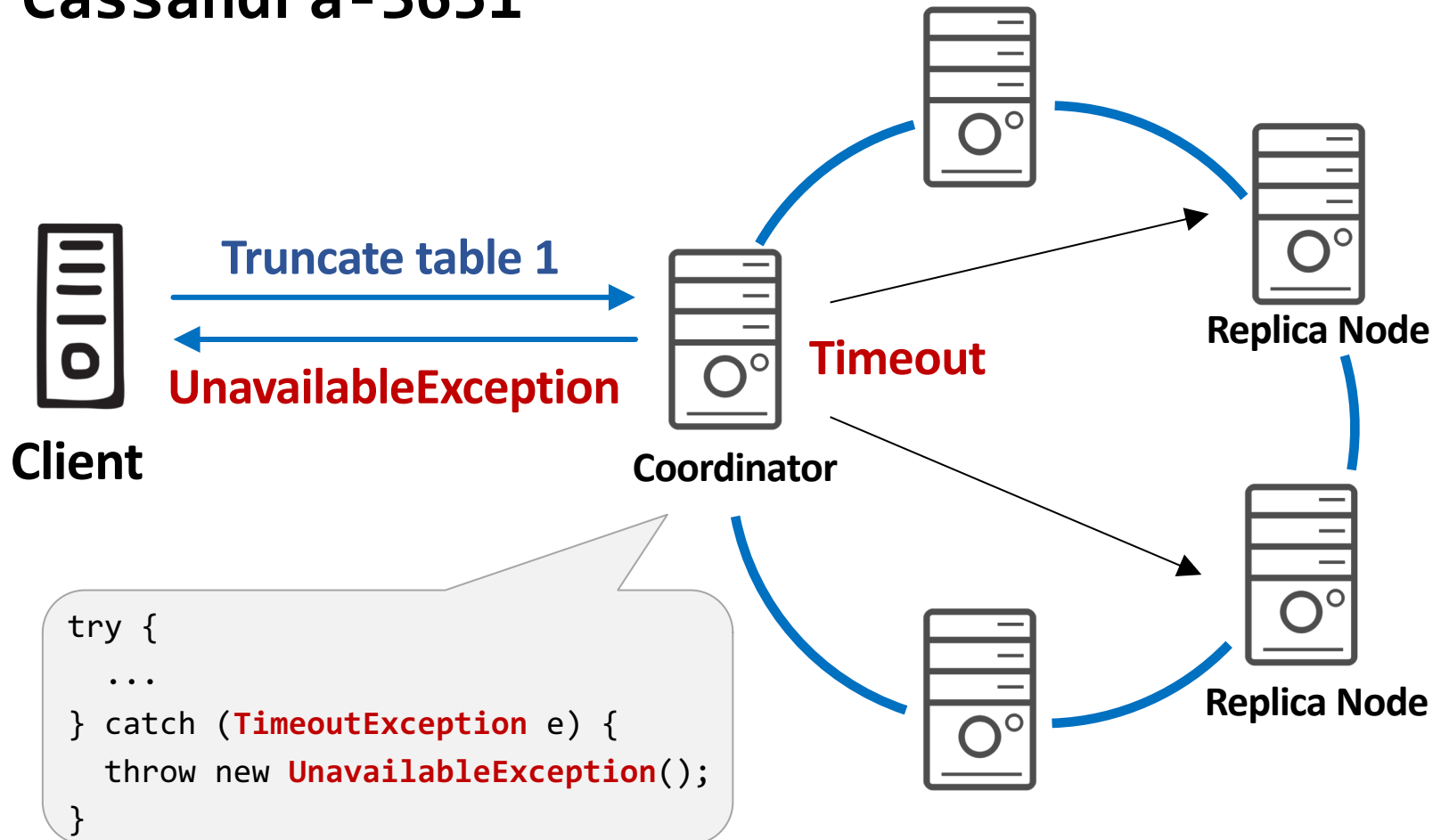
Future Work

Enhanced timeout detection tool

- ❖ Feature extraction
- ❖ Semi-supervised machine learning scheme

State of the Art

General bug studies [Gunawi et al. SoCC'14, Huang et al. SoCC'15, etc]

- ❖ They found timeout bugs widely exist in distributed systems.

Specific bug studies [Yin et al. SOSP'11, Wang et al. IC2E'15, etc]

- ❖ Misconfigurations; Data Corruption; Performance; Concurrency.

Performance bug diagnosis [Dean et al. SoCC'14, etc]

- ❖ Existing tools cannot detect/diagnose performance anomalies caused by timeout bugs [ICAC'15].

Concurrency bug detection/fix [Jin et al. OSDI'12, PLDI'12, etc]

- ❖ Our study reveals under-studied types of root causes for concurrency bugs: missing, misused, and unnecessary timeout.

Conclusion

- We perform a characteristic study of **156** real-world timeout bugs in **11** popular open source cloud server systems.

- **81%** timeout bugs are caused by either misused timeout values or missing timeout checking.

- Timeout problems have serious impact to both cloud server systems and applications.

- Existing timeout issues are difficult to diagnose with **71%** bugs producing no error message or misleading error messages.

Thank you!