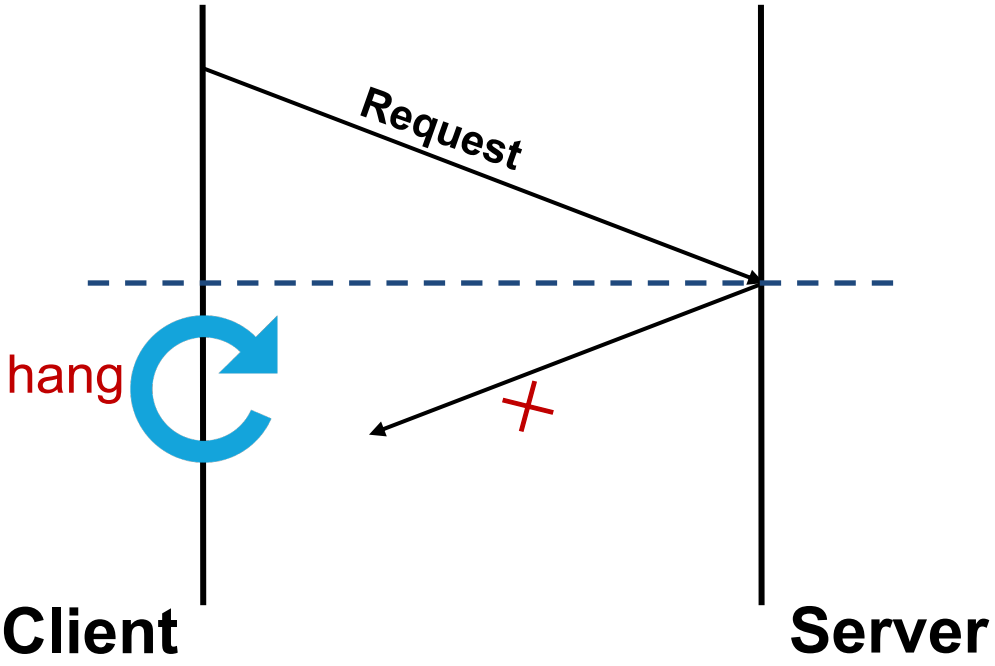
# TFix: Automatic Timeout Bug Fixing in Production Systems

Jingzhu He, Ting Dai, Xiaohui (Helen) Gu
NC State University

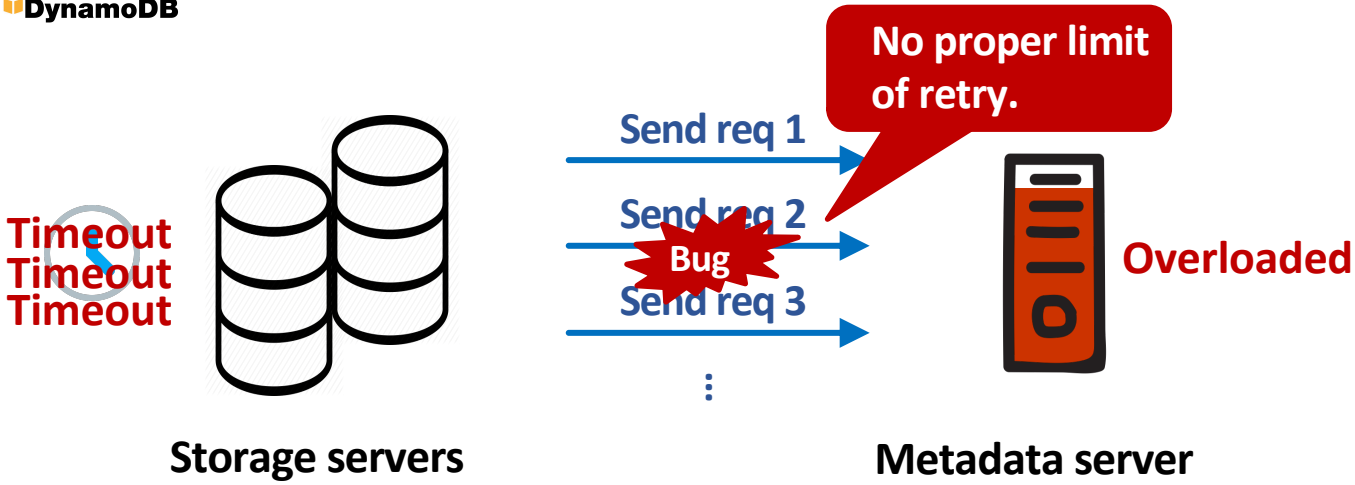
# Why Do We Need Timeout?



# Real-world Timeout Problems

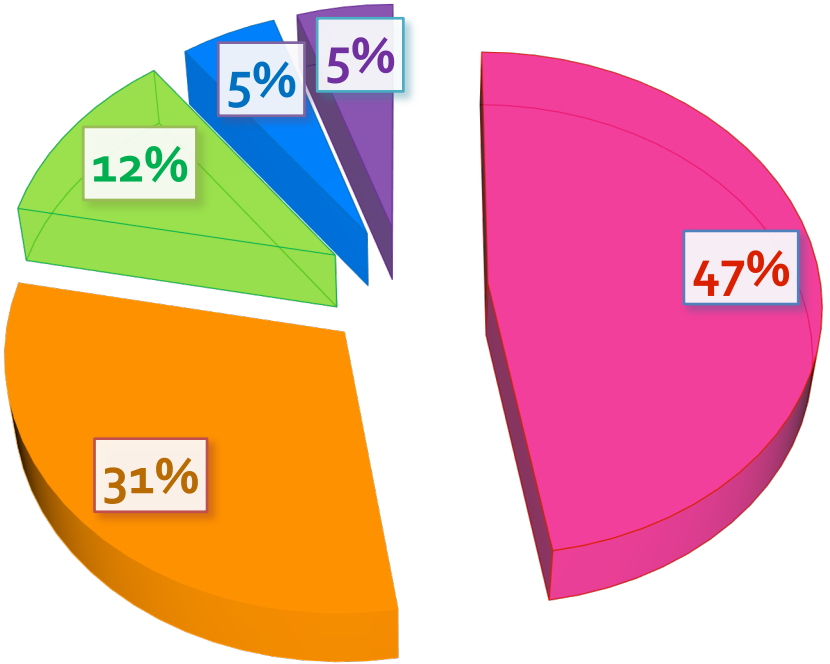

**Amazon DynamoDB service was down for 5 hours.**

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



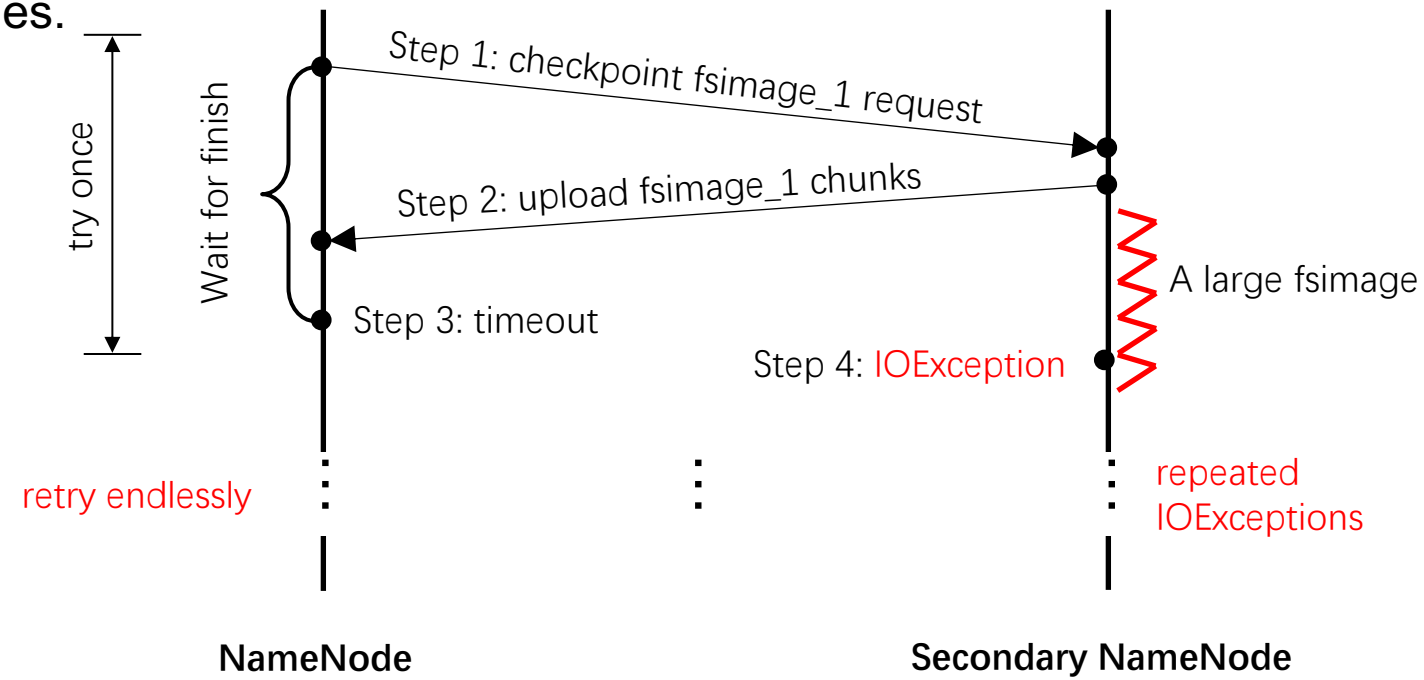
# Root Causes of Timeout Bugs

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

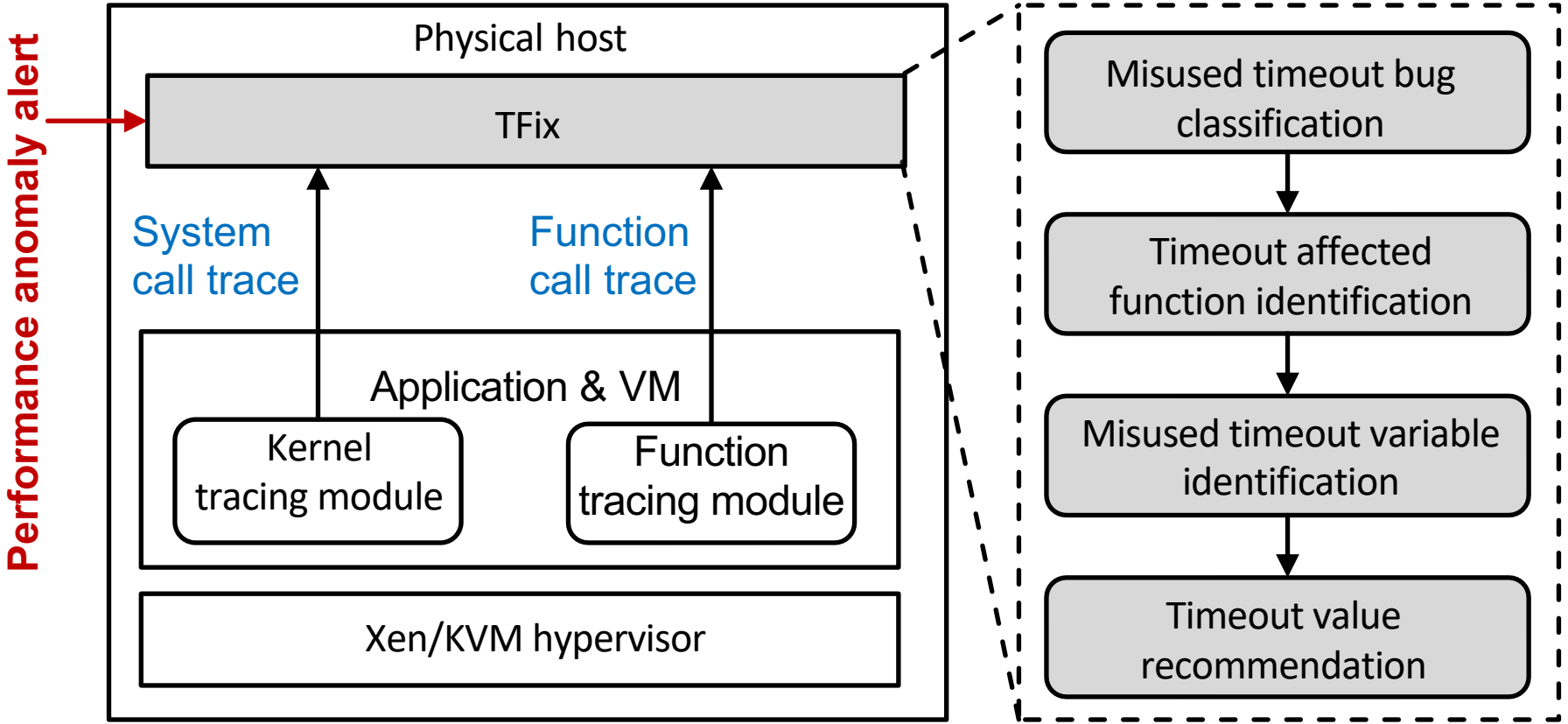


# Motivating Example (HDFS-4301)

Root cause: too small timeout value configured for transferring large fsimages.



# TFix's Overall Architecture



# Misused Timeout Bug Classification

- **Two major categories of timeout bugs:** misused timeout variable and missing timeout mechanism.
- **Timeout related function:** Java functions invoked by timeout mechanisms of particular systems.
- **Matching timeout related function:** frequent episode mining for system call sequences.

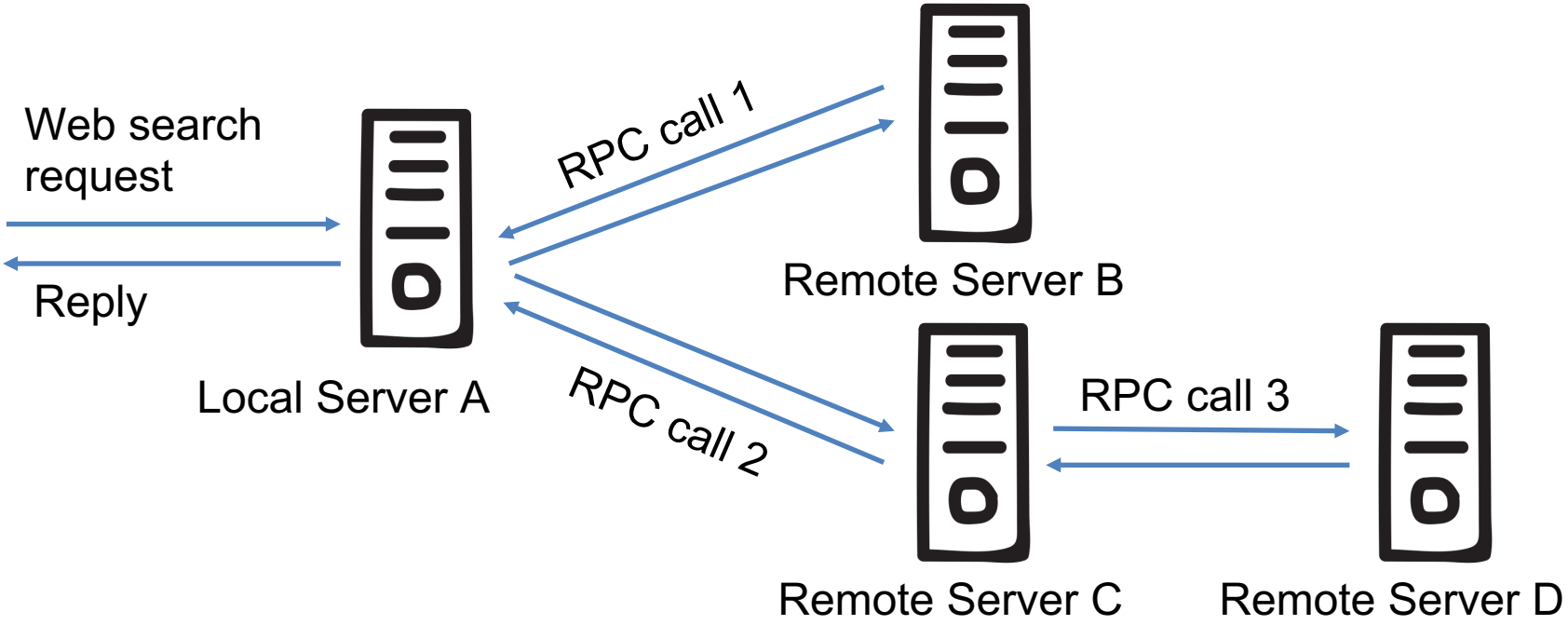
# Misused Timeout Bug Classification (Cont.)

**How to extract timeout related function:** dual testing

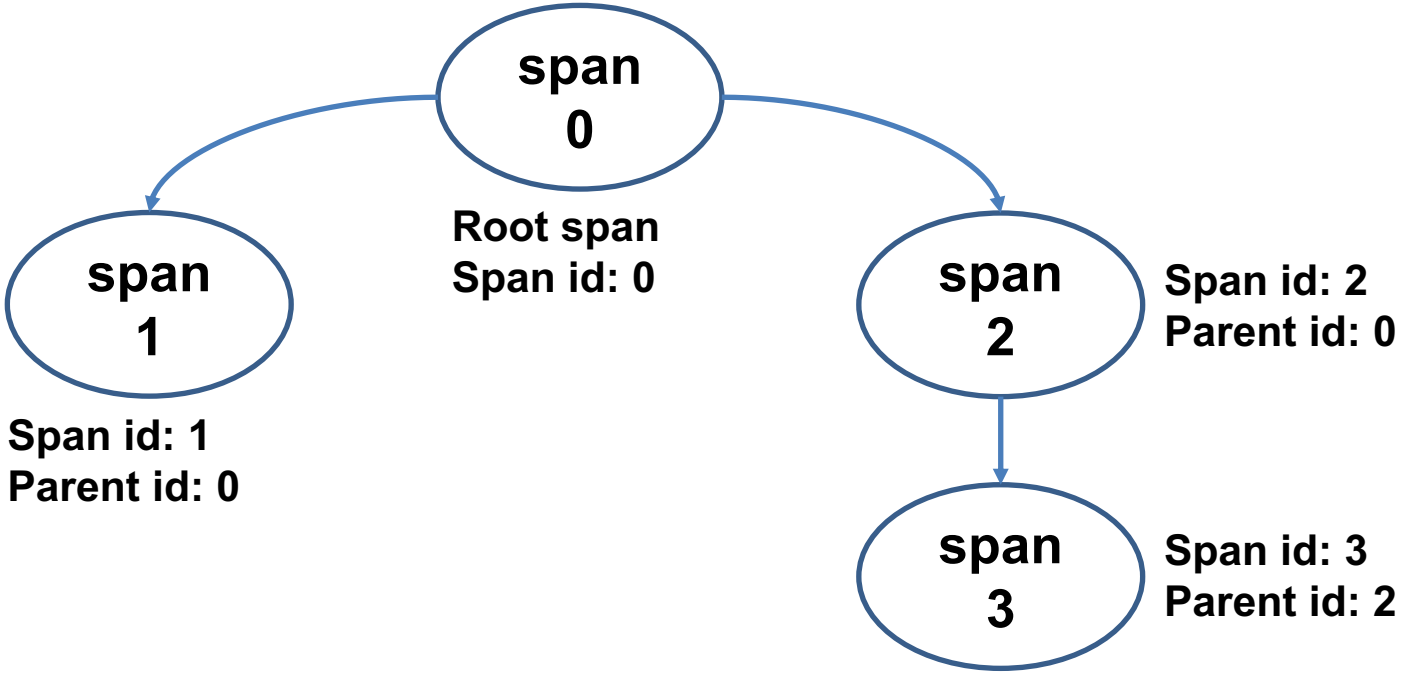
```
//Hadoop RPC mechanism with timeout
...
final Configuration conf = new Configuration();
conf.setInt(CommonConfigurationKeys.IPC_CLIENT_RPC_TIMEOUT_KEY, 1000);
TestProtocol proxy =  RPC.getProxy(TestProtocol.class,
TestProtocol.versionID, Inetaddr, conf);
...

//Hadoop RPC mechanism without timeout
...
final Configuration conf = new Configuration();
TestProtocol proxy =  RPC.getProxy(TestProtocol.class,
TestProtocol.versionID, Inetaddr, conf)
...
```

# Timeout Affected Function Identification



# Timeout Affected Function Identification (Cont.)



# Timeout Affected Function Identification (Cont.)

- Google's Dapper tracing: applicable for distributed system and incurs low runtime overhead.
- TFix augments the existing Dapper framework by instrumenting synchronization operations and IPC libraries.

{ "i":"1b1bdfddac521ce8", "s":"df4646ae00070999",
"b":1543260568612, "e":1543260568654,
"d":"org.apache.hadoop.hdfs.protocol.ClientProtocol.getDatanodeReport",
"r":"RunJar", "p":["84d19776da97fe78"] }

# Timeout Affected Function Identification (Cont.)

## Timeout value is too large:

- The execution time of the timeout affected function is much longer.
- Example:
  HBase-13647: Integer.MAX_VALUE

## Timeout value is too small:

- The frequency of the timeout affected function is much higher.
- The execution time is similar to the maximum execution time during the normal run.
- Example:
  HDFS-4301: repeated failures

# Misused Timeout Variable Identification

```
//hdfs-site.xml
1327 <property>
1328 <name>dfs.image.transfer.timeout
     </name>
1329 <value>60000</value>
...
1336 </property>

//DFSConfigKeys class
862 public static final String
863 DFS_IMAGE_TRANSFER_TIMEOUT_KEY
864 = "dfs.image.transfer.timeout";
865 public static final int
866 DFS_IMAGE_TRANSFER_TIMEOUT_DEFAULT
                          = 60 * 1000;
```

```
//TransferFsImage class
258 public static doGetUrl(...) throws
                              IOException {
...
271 timeout = conf.getInt(
272 DFSConfigKeys.DFS_IMAGE_TRANSFER_TIMEOUT_KEY,
273 DFSConfigKeys.DFS_IMAGE_TRANSFER_TIMEOUT_DEFAULT);
...
277 connection.setReadTimeout(timeout);
...
319 InputStream stream = connection.getInputStream();
...
358 num = stream.read(buf);
...
401 }
```

# Timeout Value Recommendation

## Timeout value is too large:

- Set the timeout value to the maximum execution time of the timeout affected function during normal run.
- Example:
  HBase-13647: RPC connection time

## Timeout value is too small:

- Suggests a larger timeout value by continuously multiplying the current timeout value by a ratio $\alpha > 1$
- Example:
  HDFS-4301: double the maximum image transferring time to tolerate failure

# Experiment Setting

- **5 Server systems**: built by Java, 3 systems are set up in distributed modes.
- **13 timeout bugs**: 8 misused timeout bugs and 5 missing timeout bugs.
- **Workloads**: run simple workloads that trigger timeout affected functions on each system.

# Timeout Bug Benchmark

| Bug ID | System version | Root cause | Impact |
|---|---|---|---|
| **Hadoop-9106** | v2.0.3-alpha | Misused (too large) | Slowdown |
| **Hadoop-11252(v2.6.4)** | v2.6.4 | Misused (too large) | Hang |
| **HDFS-4301** | v2.0.3-alpha | Misused (too small) | Job failure |
| **HDFS-10223** | v2.8.0 | Misused (too large) | Slowdown |
| **MapReduce-6263** | v2.7.0 | Misused (too small) | Job failure |
| **MapReduce-4089** | v2.7.0 | Misused (too large) | Slowdown |
| **HBase-15645** | v1.3.0 | Misused (too large) | Hang |
| **HBase-17341** | v1.3.0 | Misused (too large) | Hang |
| **Hadoop-11252(v2.5.0)** | v2.5.0 | Missing | Hang |
| **HDFS-1490** | v2.0.2-alpha | Missing | Hang |
| **MapReduce-5066** | v2.0.3-alpha | Missing | Hang |
| **Flume-1316** | v1.1.0 | Missing | Hang |
| **Flume-1819** | v1.3.0 | Missing | Slowdown |

# Classification Results of Timeout Bugs

| Bug ID | Identified bug type | Correct classification |
|---|---|---|
| **Hadoop-9106** | Misused | ✓ |
| **Hadoop-11252(v2.6.4)** | Misused | ✓ |
| **HDFS-4301** | Misused | ✓ |
| **HDFS-10223** | Misused | ✓ |
| **MapReduce-6263** | Misused | ✓ |
| **MapReduce-4089** | Misused | ✓ |
| **HBase-15645** | Misused | ✓ |
| **HBase-17341** | Misused | ✓ |
| **Hadoop-11252(v2.5.0)** | Missing | ✓ |
| **HDFS-1490** | Missing | ✓ |
| **MapReduce-5066** | Missing | ✓ |
| **Flume-1316** | Missing | ✓ |
| **Flume-1819** | Missing | ✓ |

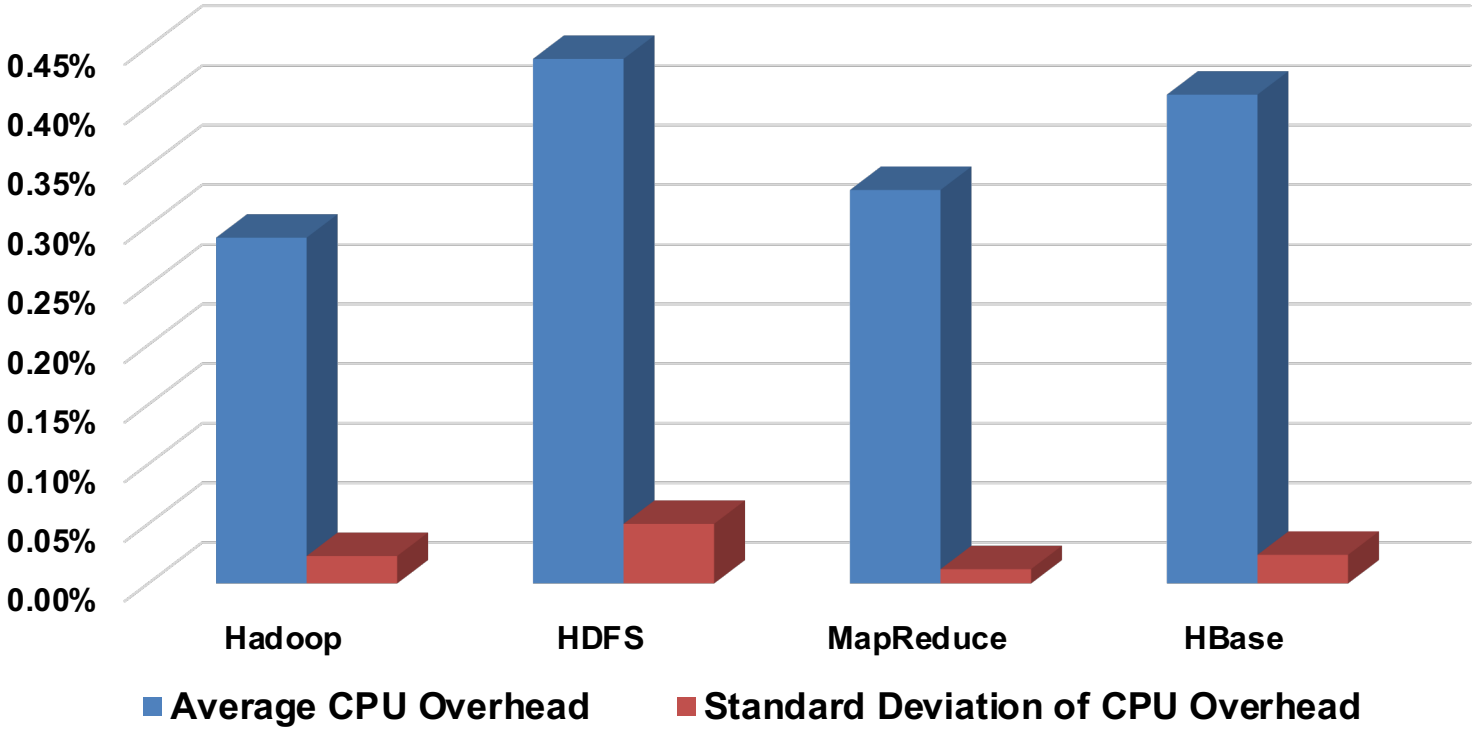
# Timeout Affected Function and Misused Timeout Variable

| Bug ID | Timeout affected function | Misused timeout variable |
|---|---|---|
| **Hadoop-9106** | Client.setupConnection() | ipc.client.connect.timeout |
| **Hadoop-11252(v2.6.4)** | RPC.getProtocolProxy() | ipc.client.rpc-timeout.ms |
| **HDFS-4301** | TransferImage.doGetUrl() | dfs.image.transfer.timeout |
| **HDFS-10223** | DFSUtilClient.peerFromSocketAndKey() | dfs.client.socket-timeout |
| **MapReduce-6263** | YARNRunner.killJob() | yarn.app.mapreduce.am.hard-kill-timeout-ms |
| **MapReduce-4089** | TaskHeartbeatHandler.PingChecker.run() | mapreduce.task.timeout |
| **HBase-15645** | RpcRetryingCaller.callWithRetries() | hbase.client.operation.timeout |
| **HBase-17341** | ReplicationSource.terminate() | replication.source.maxretriesmultiplier |

# TFix's Fixing Result

| Bug ID | Recommended timeout value | Timeout value in the patch | Is bug fixed after applying TFix recommendation |
|---|---|---|---|
| **Hadoop-9106** | 2s | 20s | ✓ |
| **Hadoop-11252(v2.6.4)** | 80ms | 0ms | ✓ |
| **HDFS-4301** | 120s | 60s | ✓ |
| **HDFS-10223** | 10ms | 1min | ✓ |
| **MapReduce-6263** | 20s | 10s | ✓ |
| **MapReduce-4089** | 100ms | 10min | ✓ |
| **HBase-15645** | 4.05s | 20min | ✓ |
| **HBase-17341** | 27ms | 1s | ✓ |

# TFix's CPU Runtime Overhead



# Related Work

- **Tracing-based bug detection and diagnosis:** X-ray(OSDI'12), Chopstix(OSDI'08), Fournier et al.(SIGOPS'10), REPT(OSDI'18), TScope(ICAC'18)
  TFix fixes timeout bugs through function call and system call tracing.
- **Configuration bug detection and diagnosis:** SPEX(SOSP'13), Yin et al.(SOSP'11), ConfValley(EuroSys'15), Xu et al.(CSUR'15), PCheck.(OSDI'16), CODE(ATC'11), ConfAid(OSDI'10), ConfDiagnoser(ICSE'13), EnCore(CAN'14)
  TFix can fix timeout bugs that are triggered during system runtime.
- **Automaic bug fix:** AFix(PLDI'11), CFix(OSDI'12), ClearView(SOSP '09), Tian et al.(ICSE'12), Tufano et al. (ASE'18)
  TFix focuses on fixing timeout bugs with a new drill-down approach that can both identify root cause variable and suggest bug fixing.

# Conclusion

- TFix employs a new drill-down analysis framework for narrowing down the root cause and recommending bug fix.
- We present a unique system call analysis scheme to classify timeout bugs.
- TFix combines dynamic function tracing and static taint analysis to localize the misused timeout variable.
- We implement a prototype of TFix and evaluate it using 13 real world timeout bugs.
- TFix is light-weight, imposing less than 1% runtime overhead.

# Acknowledgements

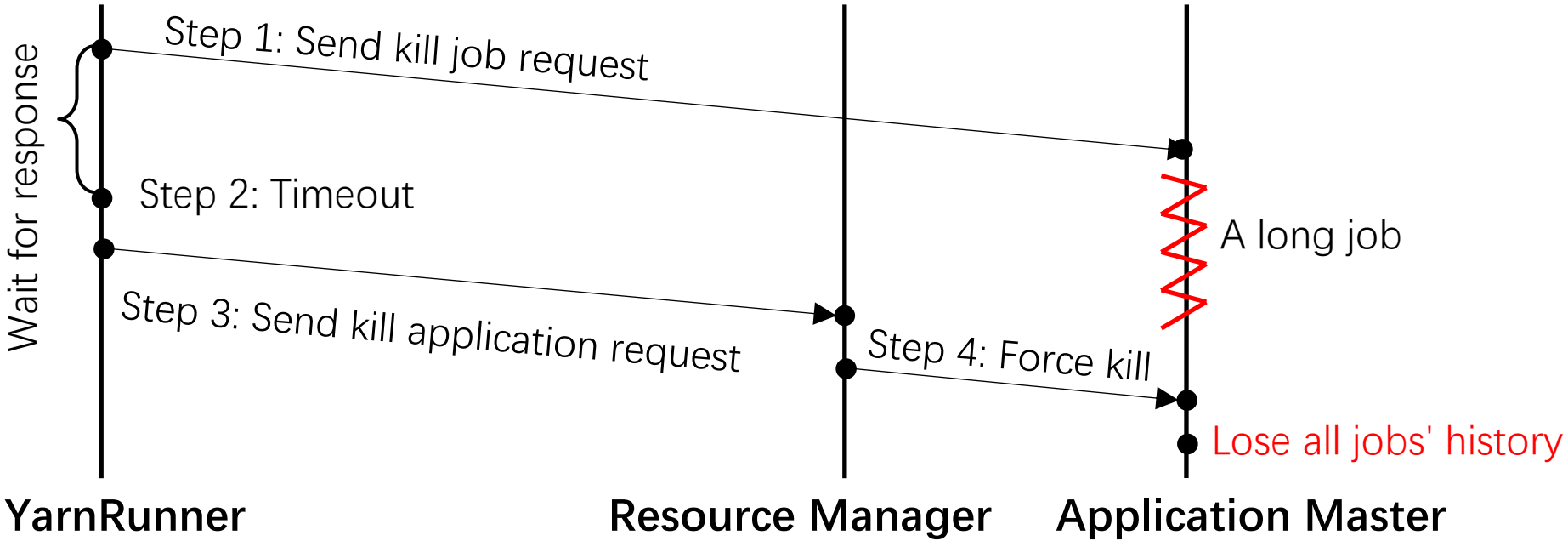

- Thanks for the comments from anonymous reviewers.

- TFix is supported in part of NSF CNS1513942 grant and NSF CNS1149445 grant.

- Thank you.

# Case Studies

- **Mapreduce-6263:** too small timeout value



# Case Studies (Cont.)

- **Mapreduce-6263:**

1) **Timeout related functions:** DecimalFormatSymbols.initialize, ReentrantLock.unlock, AbstractQueuedSynchronizer, ConcurrentHashMap.PutIfAbsent, ByteBuffer.allocate

2) **Timeout affected function identification:** YARNRunner.killJob()

3) **Misused timeout variable identification:** yarn.app.mapreduce.am.hard-kill-timeout-ms

4) **Timeout value recommendation:** double the current timeout value and replace 10s with 20s.

# Case Studies (Cont.)

- **Hadoop-9106:** too large timeout value for Hadoop's IPC connection

1) **Timeout related functions:** System.nanoTime, URL.<init>, DecimalFormatSymbols.getInstance, ManagementFactory.getThreadMXBean

2) **Timeout affected function identification:** Client.setupConnection()

3) **Misused timeout variable identification:** ipc.client.connect.timeout

4) **Timeout value recommendation:** the maximum execution time of Client.setupConnection() during normal run, recommend 2s for IPC timeout

# Limitation

- TFix cannot detect hard-coded timeout value, although TFix can pinpoint the timeout affected function.
- TFix cannot provide a proper timeout value if the timeout affected function is not invoked under current workload type.
- TFix only supports Java server systems currently.