# HangFix: Automatically Fixing Software Hang Bugs for Production Cloud Systems

Jingzhu He[1], Ting Dai[2], Xiaohui (Helen) Gu[1], Guoliang Jin[1]

[1]*NC State University*

[2]*IBM Research*

# Motivation

- 2017, British Airways experienced a serious service outage due to a software hang bug triggered by corrupted data.
- 2015, Amazon DynamoDB experienced a five-hour service outage due to endless retries during improper error handling.

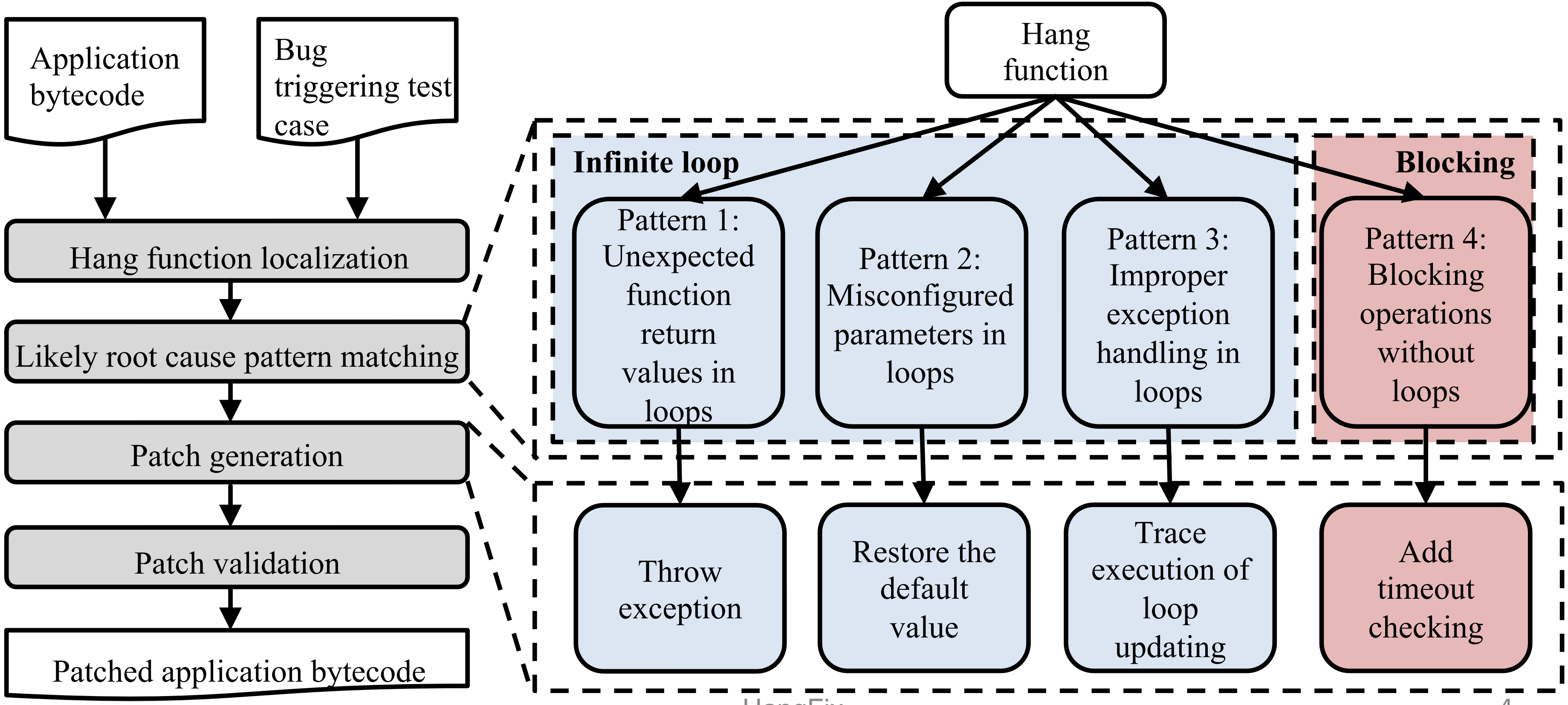
# A Hang Bug Example (HBase-8389 Bug)

- Hang bugs: software bugs cause unresponsive or frozen systems instead of system crashing.

```
public void recoverFileLease(…) … {
  boolean recovered = false;
  while (!recovered) {
      recovered = dfs.recoverLease(p);
} }
```

Corrupted file causes continuous recovery failures.

# Domain-agnostic, Byte-code-based Hang Bug Fixing


Application bytecode
Bug triggering test case
Hang function localization
Likely root cause pattern matching
Patch generation
Patch validation
Patched application bytecode
Hang function
Infinite loop
Blocking
Pattern 1: Unexpected function return values in loops
Pattern 2: Misconfigured parameters in loops
Pattern 3: Improper exception handling in loops
Pattern 4: Blocking operations without loops
Throw exception
Restore the default value
Trace execution of loop updating
Add timeout checking

# Hang Function Localization

- **Infinite loop hang function**

**Compress-451**

```
//Stack trace dump 1 at time 12:11:30
…
at compress.utils.IOUtils.copy(IOUtils.java:47)
at testcode.testCopy(testcode.java:32)
at testcode.main(testcode.java:12)

//Stack trace dump 2 at time 12:11:31
…
at compress.utils.IOUtils.copy(IOUtils.java:49)
at testcode.testCopy(testcode.java:32)
at testcode.main(testcode.java:12)
```

- **Blocking hang function**

**Mapreduce-5066**

```
//Stack trace dump 1 at time 10:11:30
…
at JobEndNotifier.httpNotification(JobEndNotifier.java:138)
at JobEndNotifier.localRunnerNotification(JobEndNotifier.java:148)
at TestJobEndNotifier.main(TestJobEndNotifier.java:139)

//Stack trace dump 2 at time 10:11:31
…
at JobEndNotifier.httpNotification(JobEndNotifier.java:138)
at JobEndNotifier.localRunnerNotification(JobEndNotifier.java:148)
at TestJobEndNotifier.main(TestJobEndNotifier.java:139)
```

- Hang functions repeatedly appear in the stack trace.
- The root cause function is on the top of the call stack.

# Likely Root Cause Pattern 1 and the Patching Strategy

- **Root cause pattern:**
  - The hang function contains a **loop**.
  - Loop stride depends on the **function's return value**.
- **Patching strategy:**
  - Insert **checkers** for the function's return value.
  - **Throw exceptions** on error values.

# Likely Root Cause Pattern 1: Unexpected Function Return Values in Loops

Cassandra-7330(v2.0.8)

```
protected void drain(InputStream dis, long bytesRead) … {
  long toSkip = totalSize() - bytesRead;
   toSkip = toSkip - dis.skip(toSkip);
   while (toSkip > 0) {
       toSkip = toSkip - dis.skip(toSkip);
} }
```

Corrupted InputStream

The loop stride (ret) is always 0/-1 when dis is corrupted.

# Patch Generation for Likely Root Cause Pattern 1

## Cassandra-7330(v2.0.8)

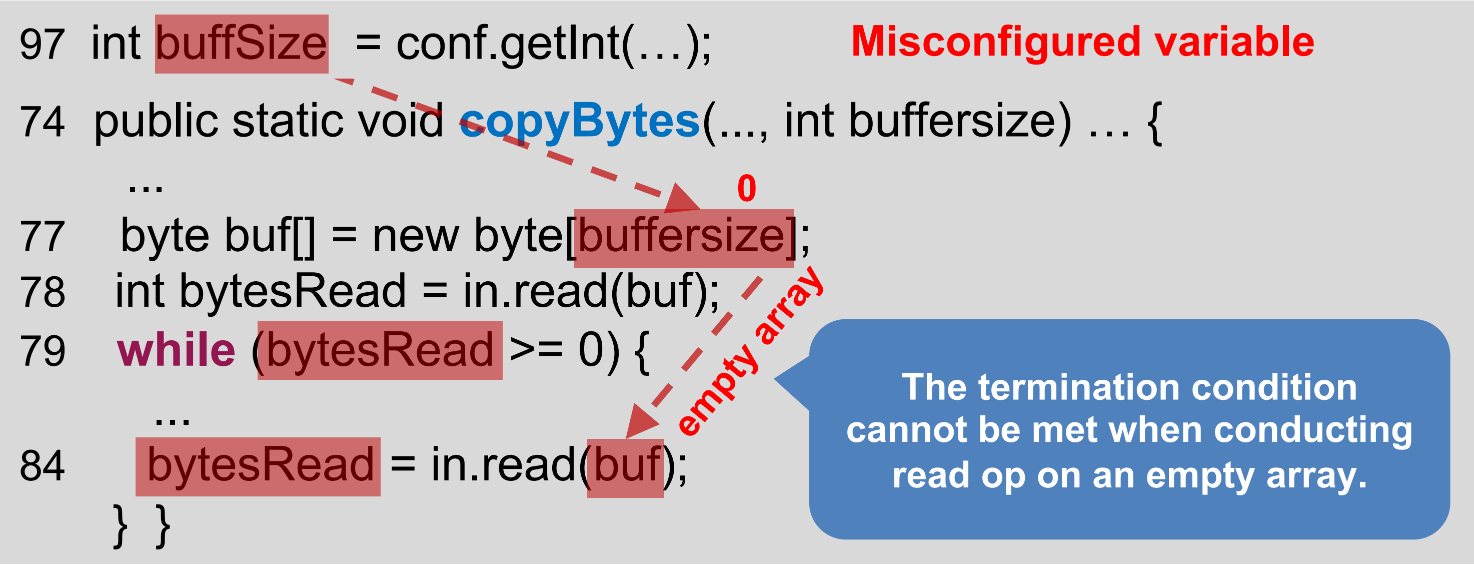
```
    protected void drain(InputStream dis, long bytesRead) … {
       …
      while (toSkip > 0) {
  -     toSkip = toSkip - dis.skip(toSkip);
  +     long skipped = dis.skip(toSkip);
  +     toSkip = toSkip - skipped;
  +     if (skipped <= 0) {
  +        throw new IOException(“Unexpected return value causes the
  +                                  loop stride to be incorrectly updated.”);
  +      }
    } }
```

# Likely Root Cause Pattern 2 and the Patching Strategy

- **Root cause pattern:**
  - The hang function contains a **loop**.
  - Loop stride depends on a **configurable parameter**.
- **Patching strategy:**
  - Insert **checkers** for the configurable parameter.
  - **Throw exceptions** on error values.

# Likely Root Cause Pattern 2: Misconfigured Parameters in Loops

## Hadoop-15415(v2.5.0)



```
int buffSize  = conf.getInt(…);
public static void copyBytes(..., int buffersize) … {
    ...
    byte buf[] = new byte[buffersize];
   int bytesRead = in.read(buf);
   while (bytesRead >= 0) {
      ...
     bytesRead = in.read(buf);
   } }
```

Misconfigured variable

0

empty array

The termination condition cannot be met when conducting read op on an empty array.

# Patch Generation for Likely Root Cause Pattern 2

## Hadoop-15415(v2.5.0)

```
int buffSize  = conf.getInt(…);
+       if (buffSize == 0) {
+          throw IOException(“Misconfigured buffSize with 0”);

public static void copyBytes(..., int buffersize) … {
        ...
+       if (buffSize == 0) {
+          throw new IOException(“buffSize cannot be 0”);
   byte buf[] = new byte[buffersize];
   int bytesRead = in.read(buf);
    while (bytesRead >= 0) {
        ...
      bytesRead = in.read(buf);
   } }
```

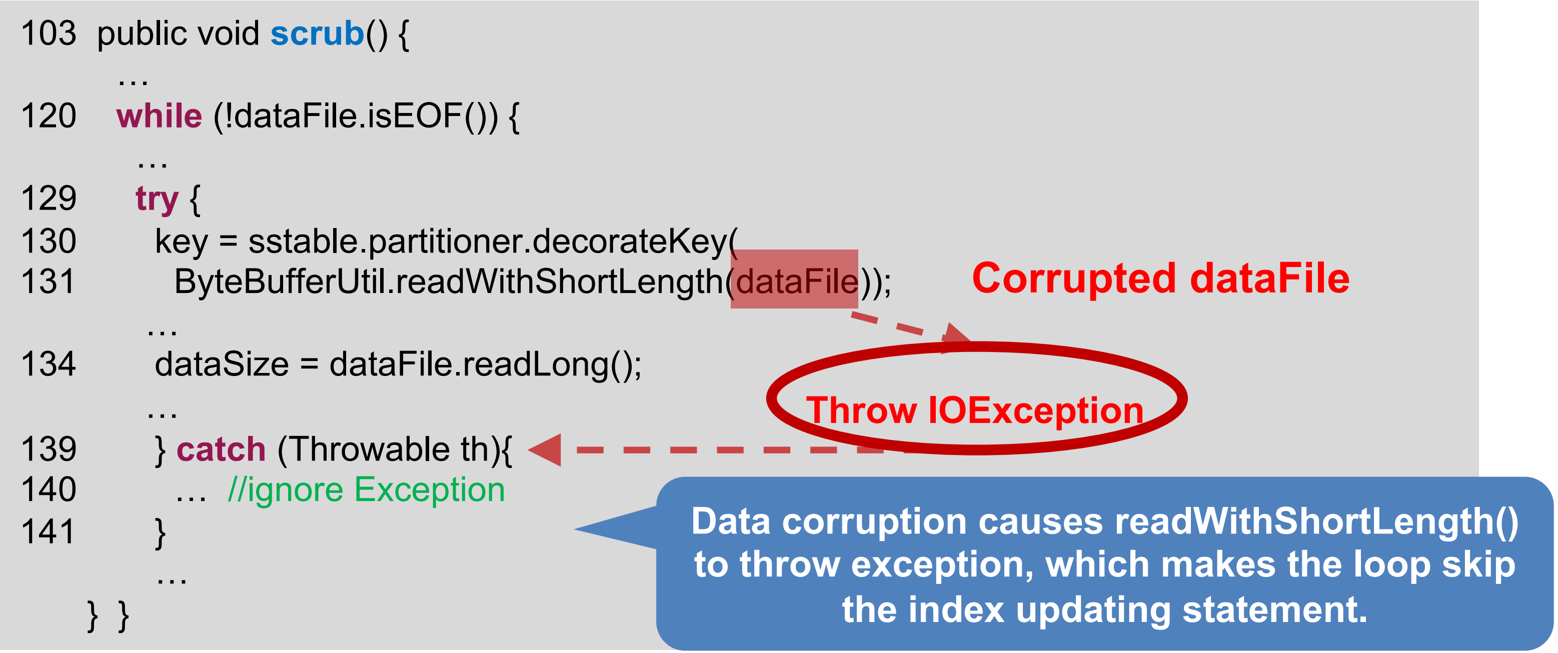
# Likely Root Cause Pattern 3 and the Patching Strategy

- **Root cause pattern:**
  - The hang function contains a **loop**.
  - Loop stride update is **skipped** due to some **exceptions**.

- **Patching strategy:**
  - **Index tracing**.
  - Insert **checkers** of the loop index.
  - **Throw exceptions** when index is not updated.

# Likely Root Cause Pattern 3: Improper Exception Handling in Loops

## Cassandra-9881(v2.0.8)



```
public void scrub() {
  …
  while (!dataFile.isEOF()) {
    …
    try {
      key = sstable.partitioner.decorateKey(
        ByteBufferUtil.readWithShortLength(dataFile));
     …
      dataSize = dataFile.readLong();
     …
      } catch (Throwable th){
        … //ignore Exception
      }
      …
   } }
```

Corrupted dataFile

Throw IOException

Data corruption causes readWithShortLength() to throw exception, which makes the loop skip the index updating statement.

# Patch Generation
# for Likely Root Cause Pattern 3

## Cassandra-9881(v2.0.8)

```
103-  public void scrub() {
   +  public void scrub() throws IOException {
120    while (!dataFile.isEOF()) {
   +     int index = 0;
129      try {
130        key = sstable.partitioner.decorateKey(
131          ByteBufferUtil.readWithShortLength(dataFile));
   +       int index += 3;
134        dataSize = dataFile.readLong();
   +       int index += 8;
139        } catch (Throwable th){
140          …  //ignore Exception
   +         if (index == 0)
   +           throw th;
141        }
} }
```

# Likely Root Cause Pattern 4 and the Patching Strategy

- **Root cause pattern:**
  - The hang function contains **blocking** operations without a loop.
- **Patching strategy:**
  - Put the blocking function into a **callable thread**.
  - Add a **timeout mechanism** to the callable thread.

# Likely Root Cause Pattern 4: Blocking Operations Without Loops

**Hive-5235(v1.0.0)**

```
public void decompress(...) … {
    …
    int cnt = inflater.inflate(out.array(), …);
    …
}
```

**Blocking operations**

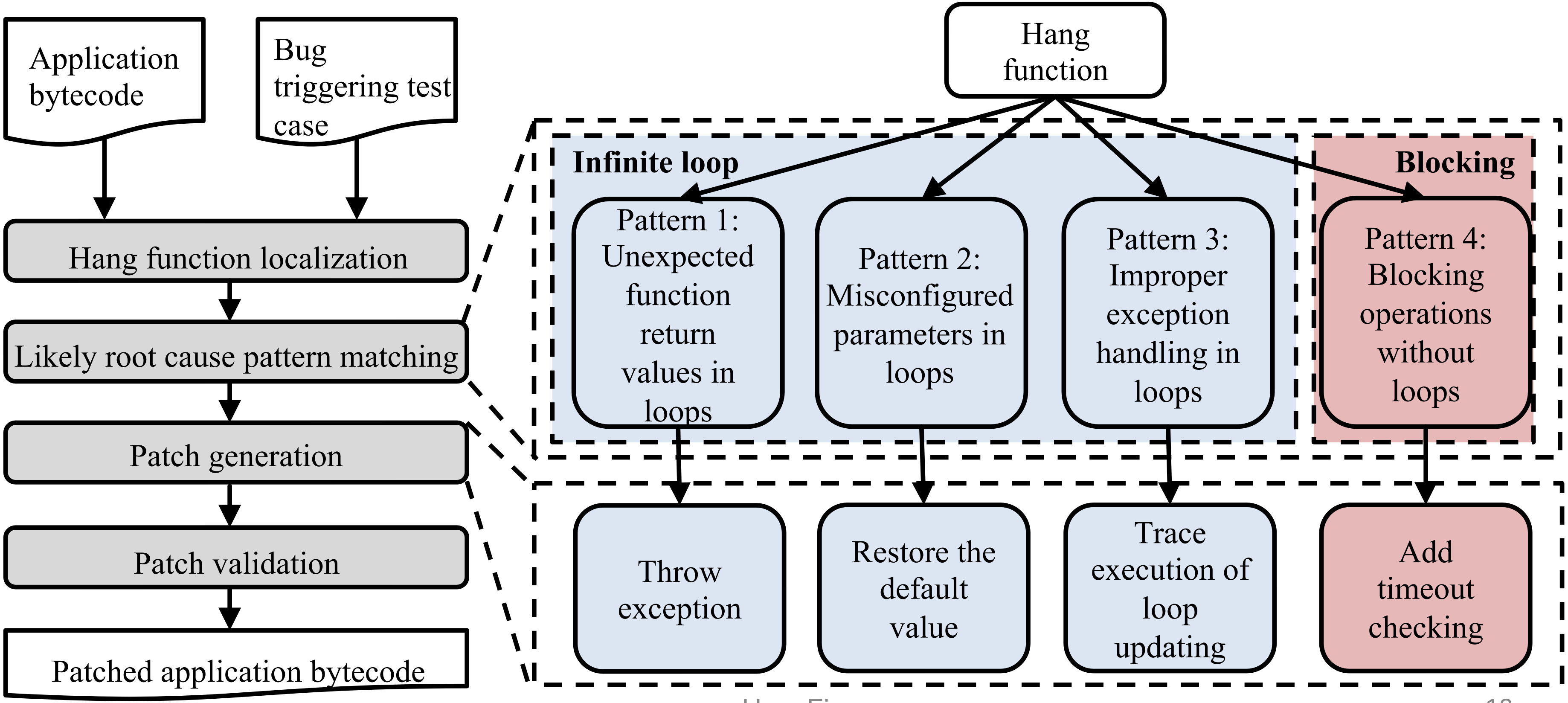
**Inflater.inflate() blocks in the underlying JNI code.**

# Patch Generation for Likely Root Cause Pattern 4

## Hive-5235(v1.0.0)

```
94-  int cnt = inflater.inflate(out.array(), …);
  +  int cnt = inflateWithTO(inflater, out.array(), ...);
      …
  +  private long timeout = conf.getLong(INFLATE_TIMEOUT_KEY, DEFAULT_INFLATE_TIMEOUT);
      …
  + public int inflateWithTO(final Inflater inflater, …) throws DataFormatException{
      …
  +  Callable<Integer> callable=new Callable<Integer>(){
       @Override
  +    public Integer call() throws DataFormatException {
         return inflater.inflate(…); }};
      …
  +  try {
        cnt = future.get(timeout, TimeUnit.MILLISECONDS);
  +  } catch (Exception e) { …
  +    throw new DataFormatException("Endless blocking");
  +  } …
```

# Domain-agnostic, Byte-code-based Hang Bug Fixing


Application bytecode
Bug triggering test case
Hang function localization
Likely root cause pattern matching
Patch generation
Patch validation
Patched application bytecode
Hang function
Infinite loop
Blocking
Pattern 1: Unexpected function return values in loops
Pattern 2: Misconfigured parameters in loops
Pattern 3: Improper exception handling in loops
Pattern 4: Blocking operations without loops
Throw exception
Restore the default value
Trace execution of loop updating
Add timeout checking

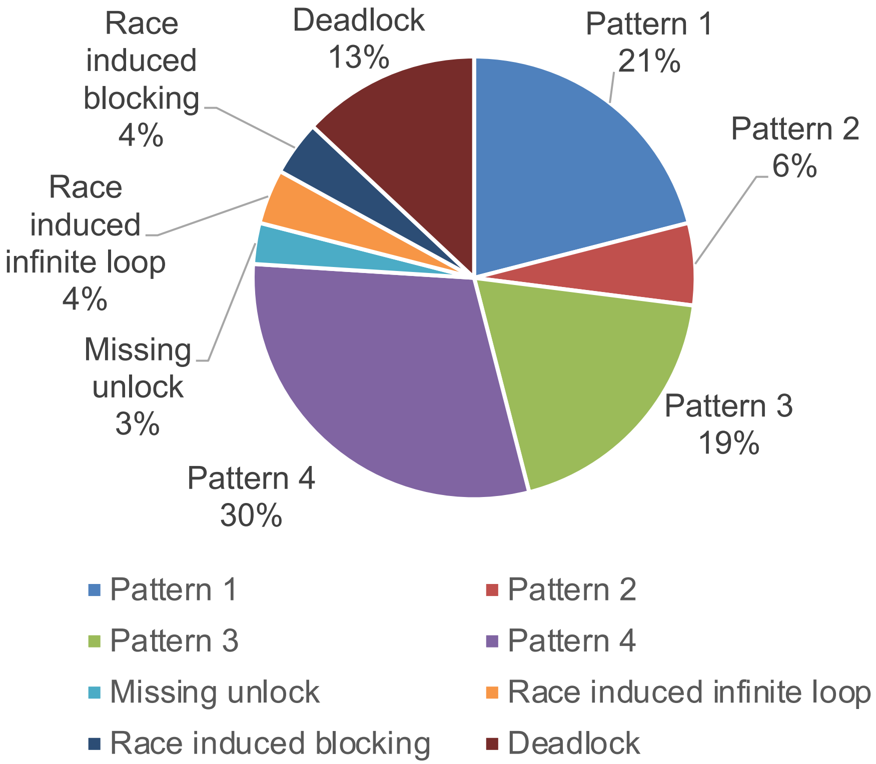
# Patch Validation

- Re-run the existing hang bug detection tools [TScope(ICAC’18), DScope(SOCC’18)].
- Re-run hang function localization tool.
- Run the applications’ regression test suites.

# Evaluation Methodology

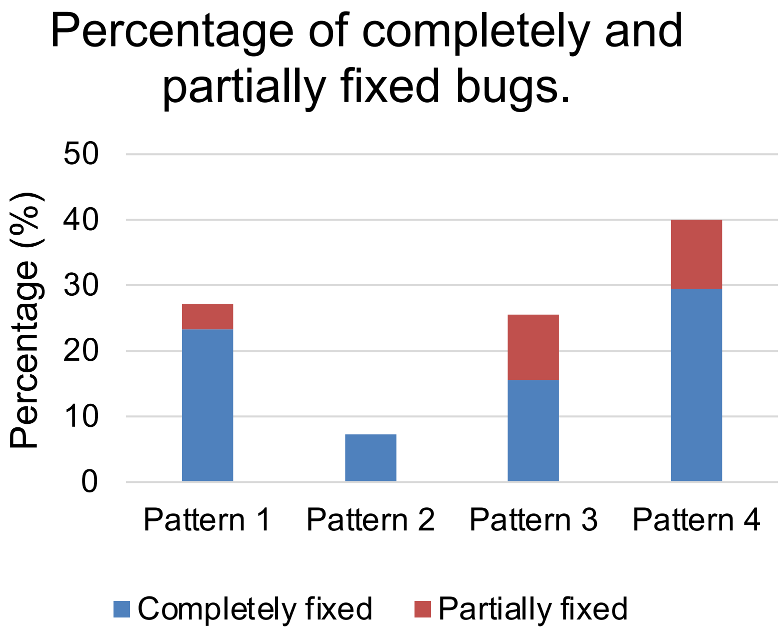
- **Empirical Study:**
  - Collected **237** bugs.
  - Quantified the generality of four root cause patterns.
  - Evaluated whether bugs of the four patterns can be fixed.
- **Experimental Evaluation:**
  - Reproduced **42** bugs.
  - Evaluated HangFix from fixing results, fixing time and patches' overhead.

# Empirical Study Results



- **4 likely root cause patterns:**
  - Cover **76%** (180/237) bugs.
- **Synchronization-related bug patterns:**
  - Missing unlock.
  - Race induced infinite loop.
  - Race induced blocking.
  - Deadlock.

# Empirical Study Results (Cont.)



- **Fixing results for the bugs of the 4 likely root cause patterns:**
  - **136** bugs can be fixed completely.
  - 44 bugs partially fixed. Application-specific operations contained or system’s state restoration is required.

# Experimental Evaluation

| System | Description | # of closed bugs | # of open bugs |
|---|---|---|---|
| **Cassandra** | Distributed database management system. | 1 | 1 |
| **Compress** | Libraries for I/O ops on compressed file. | 2 | 0 |
| **Hadoop Common** | Hadoop utilities and libraries. | 1 | 6 |
| **Mapreduce** | Hadoop big data processing framework. | 2 | 4 |
| **HDFS** | Hadoop distributed file system. | 3 | 5 |
| **HBase** | HBase database. | 1 | 0 |
| **Yarn** | Hadoop resource management platform. | 2 | 1 |
| **Hive** | Data warehouse. | 1 | 9 |
| **Kafka** | Distributed streaming platform. | 0 | 1 |
| **Lucene** | Indexing and search server. | 1 | 1 |
| **Total** | | **14** | **28** |

# Experimental Results

| Bug Patterns | Total # of bugs | # of bugs fixed by manual patches | # of bugs fixed by HangFix |
|---|---|---|---|
| Pattern 1 | 15 | 7 | 15 |
| Pattern 2 | 13 | 2 | 13 |
| Pattern 3 | 6 | 2 | 5 |
| Pattern 4 | 8 | 3 | 7 |
| Total | 42 | 14 | 40 |

Fix both closed and open bugs!

# Experimental Results (Cont.)

- **Fixing time:**
  - 0.7 to 22 seconds.
  - Depend on the intra- and inter-procedural analysis.
  - Developers take several weeks or even longer to provide manual patches.
- **CPU overhead after applying HangFix's patch:**
  - Less than 1%.

# Conclusion

- HangFix: a **domain-agnostic**, **byte-code-based** hang bug fixing framework.
  - Describe a hang bug root cause **pattern matching** scheme.
  - Present an automatic hang fix **patch generation** system.
  - Conduct an empirical study over **237** real production hang bugs and evaluation over **42** hang bugs on 10 cloud server systems.

# Acknowledgments

- Thank all anonymous reviewers for their valuable comments.

- This work is supported in part by NSF CNS1513942 grant and NSF CNS1149445 grant.

**Thank you!**

# Backup slides

# Related Work

- **Automatic bug fixing:**

  AFix[PLDI'11], CFix[OSDI'12], ClearView[SOSP'09], TFix[ICDCS'19], DFix[PLDI'19]

  - Little work focuses on hang bug fixing.

- **Hang bug detection:**

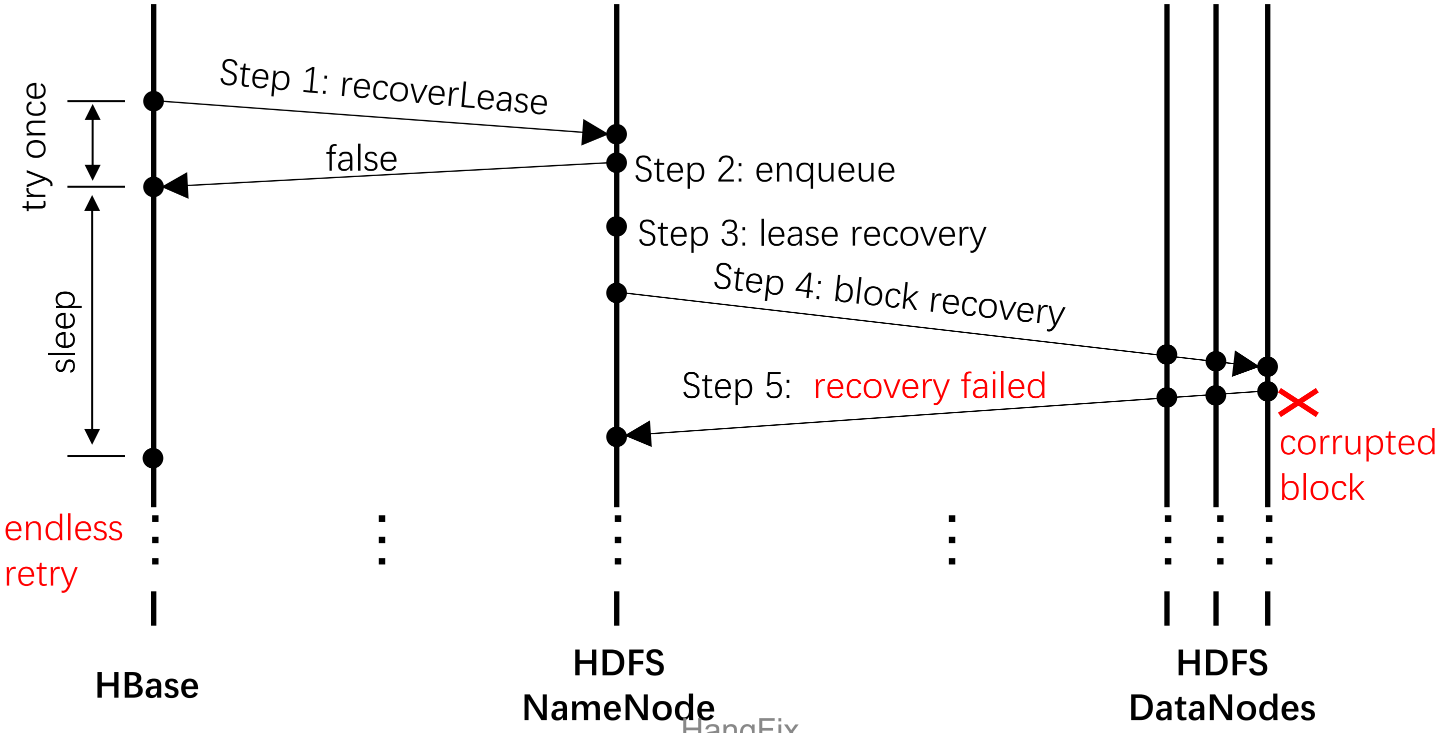
  Hang Doctor[EuroSys'18], PerfChcker[ICSE'14], HangWiz[EuroSys'08], TScope[ICAC'18], DScope[SoCC'18], Jolt[ECOOP'11], Carburier[SOSP'09]

  - Existing detection tools can be used as HangFix's front-end hang bug detection.

# Related Work

- **Automatic bug fixing:**
  - **Fixing tools for functional and performance bugs.** (AFix[PLDI’11], CFix[OSDI’12], ClearView[SOSP’09], TFix[ICDCS’19], DFix[PLDI’19])
  - **Hybrid methods to fix the bugs.** (Genprog[TSE’12], Assure[ASPLOS’09], Ares[ASE’16], SemFix[ICSE’13], Remix[SIGPLAN Notices’16], Huron[PLDI’19])
- **Hang bug detection:**
  - **Generic hang bug detection tools.** (Hang Doctor[EuroSys’18], PerfChcker[ICSE’14], HangWiz[EuroSys’08])
  - **Specific hang bug detection.** (TScope[ICAC’18], DScope[SoCC’18], Jolt[ECOOP’11], Carburier[SOSP’09])
  - **Detecting hang issues at middleware and hardware layers.** (BLeak[SIGPLAN Notices’18], CLARITY[SIGPLAN Notices’15], DeadWait[SIGPLAN Notices’17])

# Motivating Example (HBase-8389 Bug) change to code



# Hang Function Localization

## Compress-451

### //Dump 1

```
"main" #1 prio=5 os_prio=0 tid=0x00007f899c00b000 nid=0 x76b9 runnable [0x00007f89a27fa000]
java.lang.Thread.State: RUNNABLE
at java.io.FileInputStream.readBytes(Native Method)
at java.io.FileInputStream.read(FileInputStream.java:233)
at org.apache.commons.compress.utils.IOUtils.copy(IOUtils.java:47)
at testcode.testCopy(testcode.java:32)
at testcode.main(testcode.java:12)
```

### //Dump 2

```
"main" #1 prio=5 os_prio=0 tid=0x00007f899c00b000 nid=0 x76b9 runnable [0x00007f89a27fa000]
java.lang.Thread.State: RUNNABLE
at java.io.FileOutputStream.writeBytes(Native Method)
at java.io.FileOutputStream.write(FileOutputStream.java:326)
at org.apache.commons.compress.utils.IOUtils.copy(IOUtils.java:49)
at testcode.testCopy(testcode.java:32)
at testcode.main(testcode.java:12)
```

**...**

# Patch Generation for Likely Root Cause Pattern 4

## Hive-5235(v1.0.0)

```
94-  int cnt = inflater.inflate(out.array(), …);
  +  int cnt = inflateWithTO(inflater, out.array(), ...);

  + private Configuration conf = new Configuration();
  + private String INFLATE_TIMEOUT_KEY = "orc.zlibcodec.inflate.timeout";
  + private long DEFAULT_INFLATE_TIMEOUT = 5000;
  + private long timeout = conf.getLong(INFLATE_TIMEOUT_KEY, DEFAULT_INFLATE_TIMEOUT);

  + public int inflateWithTO(final Inflater inflater, …) throws DataFormatException{
  +   ExecutorService executor = Executors.newSingleThreadExecutor();
  +   Callable<Integer> callable=new Callable<Integer>(){ @Override
  +     public Integer call() throws DataFormatException { return inflater.inflate(…); }};
  +   Future<Integer> future = executor.submit(callable);
  +   int cnt = 0;
  +   try { cnt = future.get(timeout, TimeUnit.MILLISECONDS);
  +   } catch (Exception e) { future.cancel(true);
  +     throw new DataFormatException("Endless blocking");
  +   } finally { executor.shutdown(); }
  +   return cnt; }
```