DScope: Detecting Real-World Data Corruption Hang Bugs in Cloud Server Systems

Ting Dai[1], Jingzhu He[1], Xiaohui (Helen) Gu[1], Shan Lu[2], Peipei Wang[1]

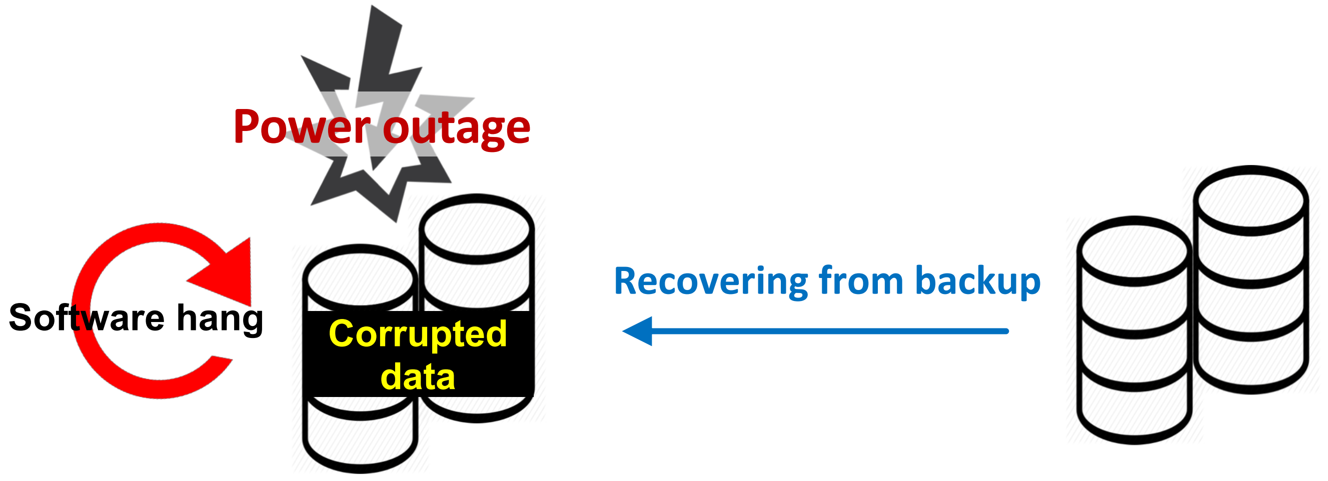
[1]*NC State University* [2]*University of Chicago*

ACM Symposium on Cloud Computing

Real-World Data Corruption Problem

BRITISH AIRWAYS

British Airway service was down for hours with financial penalty of £ 100 million.



Primary data center

Backup data center

A Data Corruption Hang Bug Example

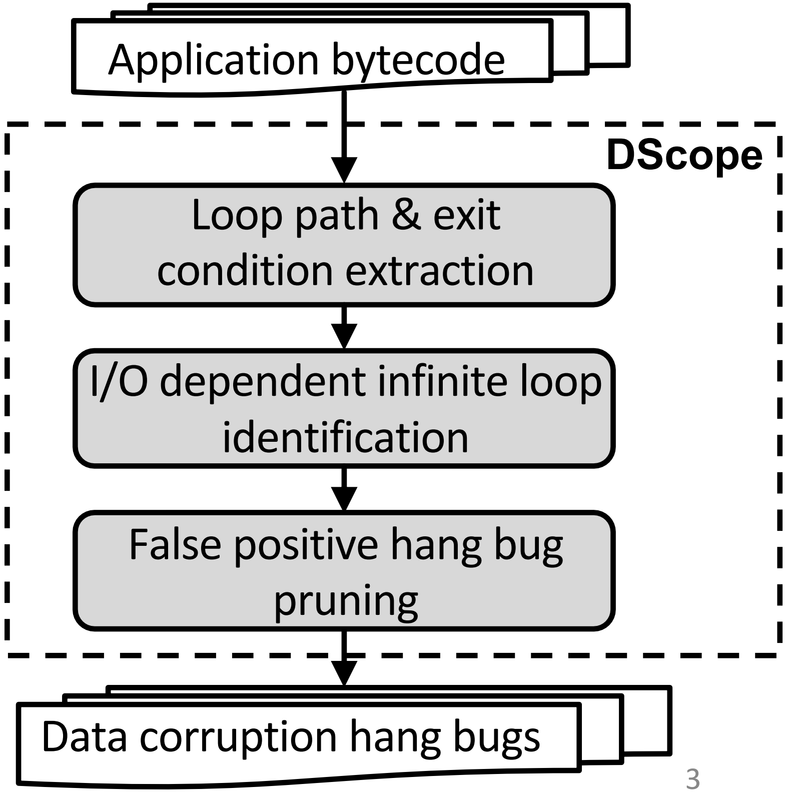
Hadoop-8614

```
public static void skipFully(
        InputStream in, long len) … {
  while (len > 0) {
     long ret = in.skip(len);
     …
     …
     len -= ret;
  }
}
```

Corrupted InputStream

The loop stride (ret) is always 0 when in is corrupted.

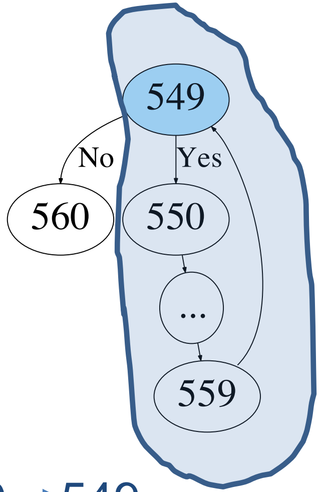
Overview of DScope


Application bytecode
DScope
Loop path & exit condition extraction
I/O dependent infinite loop identification
False positive hang bug pruning
Data corruption hang bugs

Loop Path & Exit Condition Extraction

- **Simple Loops**

```
549    for ( int j = 0; j < length; j++)  {
550      String rack = racks[j] ;
         . . .
559    }
560
```

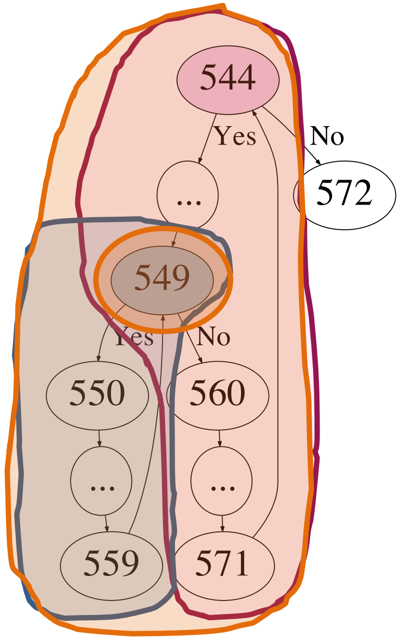


Loop path: 549→ 550→ … →559 →560 →549

Exit condition: j >= length

Loop Path & Exit Condition Extraction

- **Nested Loops**



Loop paths:

Outer: 544 → … → 549 → 560 → … → 571 → 544

Inner: 549 → 550 → … → 559 → 549

Outer: 544 → … 560 → … → 571 → 544

DScope then extracts the exit conditions for each loop path.

Loop Path & Exit Condition Extraction

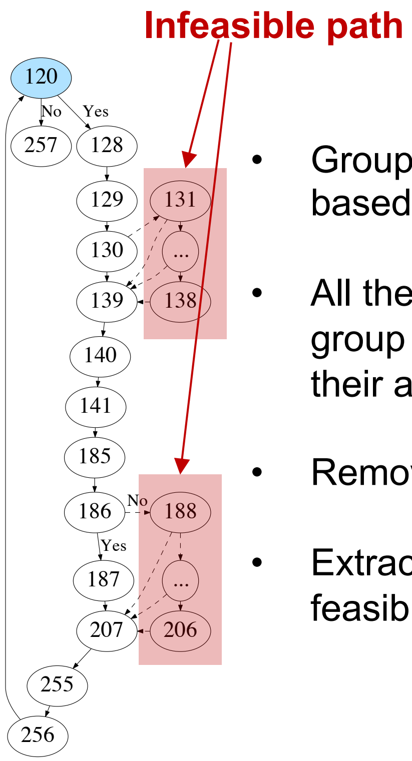
- **Loops with exception handling**

```
120  while (!dataFile.isEOF()) {
       …
129    try {
130      key = decorateKey(…dataFile);
          …
139    } catch (Throwable th) {
140      //ignore exception
141    }
       …
185    try {
186      if (key == null)
187        throw new IOError(…);
          …
207   } catch (Throwable th) {
208     //ignore exception
     } }
```

Corrupted dataFile

throw exception

throw exception



- Group invocation statements based on arguments.
- All the statements in the same group throw exceptions when their arguments get corrupted.
- Remove infeasible loop paths.
- Extract exit conditions of the feasible loop paths.

I/O Dependent Infinite Loop Identification

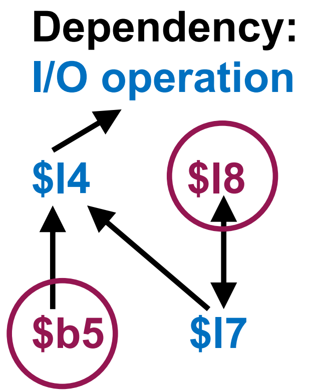
- **Exit conditions *directly* depend on I/O operations**

```
//Soot IR
198  $i1 = r0.<InputStream: read()>(r2) //$i1 is an I/O related variable
199  if $i1 == -1 goto line #203         //``$i1 == -1" is the exit condition
     ...
202  goto line #198
```

I/O Dependent Infinite Loop Identification

- **Exit conditions *indirectly* depend on I/O operations**

```
//Soot IR
3  if l8 >= l0 goto line #12 //``l8 >= l0'' is the exit condition
   ...
5  $l2 = l0 - l8
6  $l4 = $r2.<InputStream: skip>($l2) //$l4 is an I/O related variable
7  $b5 = $l4 cmp 0L
8  if $b5 == 0 goto line #12 //``$b5 == 0" is the exit condition
9  $l7 = $l8 + $l4
10 i8 = $l7
11 goto line #3
```



I/O Dependent Infinite Loop Identification

- **Exit conditions depend on complex I/O related variables**
  - DScope performs an integrated analysis by linking variable information from IR code, Java source code, and Java bytecode.
  - User annotated I/O variables.

False Positive Filtering

Hadoop v2.5.0 WritableUtils.java

```
public static long readVLong(DataInput stream)…{
     byte firstByte = stream.readByte();
     int len = decodeVIntSize(firstByte);
      …
     for (int idx = 0; idx < len-1; idx++) {
          …
} }
```

len is I/O dependent

It's a FP because the loop stride is always 1 and the upper bound (len-1) is fixed.

- **False positive condition**:
  - The loop stride is always **positive** when the loop index has a fixed **upper** bound;
  - The loop stride is always **negative** when the loop index has a fixed **lower** bound.

Loop Stride and Bound Inference

- Stride and bounds are denoted by
  - Numeric primitives

```
for (int idx = 0; idx < len-1; idx++) {
    …
}
```

Bound (len-1)

Stride (1)

Loop Stride and Bound Inference

- **Stride and bounds are denoted by**
  - **APIs in 60 commonly used Java classes**

    Forward index   Reverse index   Check bounds

    Reset index   Update bounds

```
RandomAccessReader dataFile;

while (!dataFile.isEOF()) {
    …
    dataSize = dataFile.readLong();
}
```

Bound checking

Stride forwarding

Evaluation

System	Description	# of bugs
Cassandra	Distributed database management system	2
Compress	Libraries for I/O ops on compressed file	2
Hadoop Common	Hadoop utilities and libraries	10
Mapreduce	Hadoop big data processing framework	5
HDFS	Hadoop distributed file system	4
Yarn	Hadoop resource management platform	4
Hive	Data warehouse	12
Kafka	Distributed streaming platform	1
Lucene	Indexing and search server	2

- **Implemented a prototype of DScope using Soot;**
- **State-of-the-art static bug detectors**:
  - **Findbugs**
  - **Infer**

Bug Detection Results

System		DScope		Findbugs	Infer
		TP	FP	TP	TP
Cassandra	v2.0.8	2	1	0	1
Compress	v1.0	2	2	0	-
Hadoop Common	v0.23.0	4	6	0	0
	v2.5.0	6	6	0	0
Mapreduce	v0.23.0	3	0	0	0
	v2.5.0	2	0	0	0
HDFS	v0.23.0	1	1	0	0
	v2.5.0	3	5	1	-
Yarn	v0.23.0	2	2	1	0
	v2.5.0	2	5	0	0
Hive	v1.0.0	7	6	0	-
	v2.3.2	5	1	0	0
Kafka	v0.10.0.0	1	1	0	0
Lucene	V2.1.0	2	1	0	0
Total		42	37	2	1

Data Corruption Hang Bug Types

- Type 1: Error codes returned by I/O operations directly affect loop strides.
- Type 2: Corrupted data content indirectly affects loop strides.
- Type 3: Improper exception handling directly affects loop strides.
- Type 4: Improper exception handling indirectly affects loop strides.

Data Corruption Hang Bug Types

- **Type 1: Error codes returned by I/O operations directly affect loop strides.**

Hadoop-8614

```
public static void skipFully(InputStream in, long len) … {
  while (len > 0) {
     long ret = in.skip(len);   Corrupted InputStream
     …
     …
     len -= ret;
} }
```

0

The loop stride (ret) is always 0 when in is corrupted.

Data Corruption Hang Bug Types

- **Type 2: Corrupted data content indirectly affects loop strides.**

HDFS-13514

```
BUFFER_SIZE = conf.getInt();
```

Corrupted configuration file

```
private void readLocalFile(Path path, ...) ... {
    ...
    byte[] data = new byte[BUFFER_SIZE];
    long size = 0;
    while (size >= 0) {
        size = in.read(data);
} }
```

0

empty array

The loop stride (size) is always 0 when conducting read op on an empty array.

False Negative Example

The loop index, stride or bounds are **only** related to specific application I/O functions.

HDFS-5438

Application function

```
while (!fileComplete) {
    fileComplete = dfsClient.namenode.complete(src,
                            dfsClient.clientName, last);
    ...
}
```

Corrupted block

False Positive Example

Hadoop v2.5 BlockReaderLocal.java

```
private int readWithBounceBuffer(
                          ByteBuffer buf…) …{
  do {
     …
     bb = drainDataBuf(buf);
  } while (buf.remaining() > 0);
   …
}
```

Check bounds

```
private int drainDataBuf(
                  ByteBuffer buf) {
     …
   buf.put(dataBuf);
     …
}
```

Forward index

- **The forwarding-index Java APIs and the checking-bounds Java APIs are located in different application function.**

Conclusion

- DScope is a new data corruption hang bug detection tool for cloud server systems.
  - Combines candidate bug discovery and false positive filtering.
  - Evaluated over 9 cloud server systems and detects **42** true data corruption hang bugs including **29** new bugs.

Acknowledgements

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

Thank you