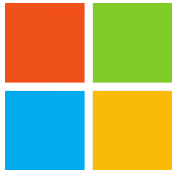
A Hybrid Approach to Cloud System Performance Bug Detection and Diagnosis

Ting Dai

Advisor: Xiaohui (Helen) Gu

NC State University

Real-World Performance Problems



- Nov. 2014, Microsoft Azure storage service outage caused by an infinite loop [link]

- Feb. 2016, Redis cloud outage caused by a resource-intensive infinite loop [link]

amazon

- Jan. 2018, Intel Meltdown patch will slow down your AWS EC2 server [link]

State-of-the-Art

Static analysis

Facebook Infer, Findbugs, PMD, Camarel[ICSE'15], Jin et al.[PLDI'12], etc

- **Advantage**:
  - No runtime overhead
  - Program semantics
- **Disadvantage**:
  - High FP by generic approaches
  - High FN by specific approaches

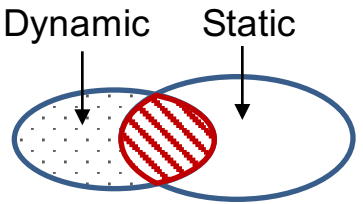
Dynamic analysis

PerfScope[SoCC'14], PerfCompass[TPDS'15], X-ray[OSDI'12], PBI[ASPLOS'13], etc

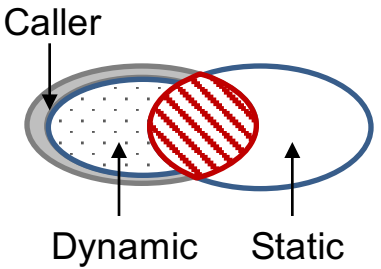
- **Advantage:**
  - More accurate due to runtime behavior
- **Disadvantage:**
  - Limited coverage (FN)

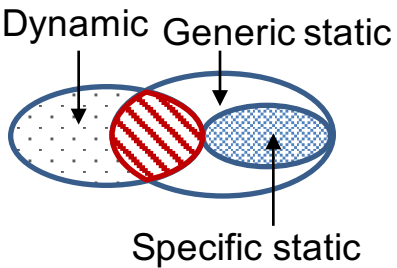
Hybrid Analysis

- **Rationale:**
  - Functions with buggy characteristics which are also experiencing a runtime problem are **most** likely related to a bug.



- **Design choice:**
  - A suspicious caller function can cause its callee functions behave abnormally runtime.
  - Generic static approach.

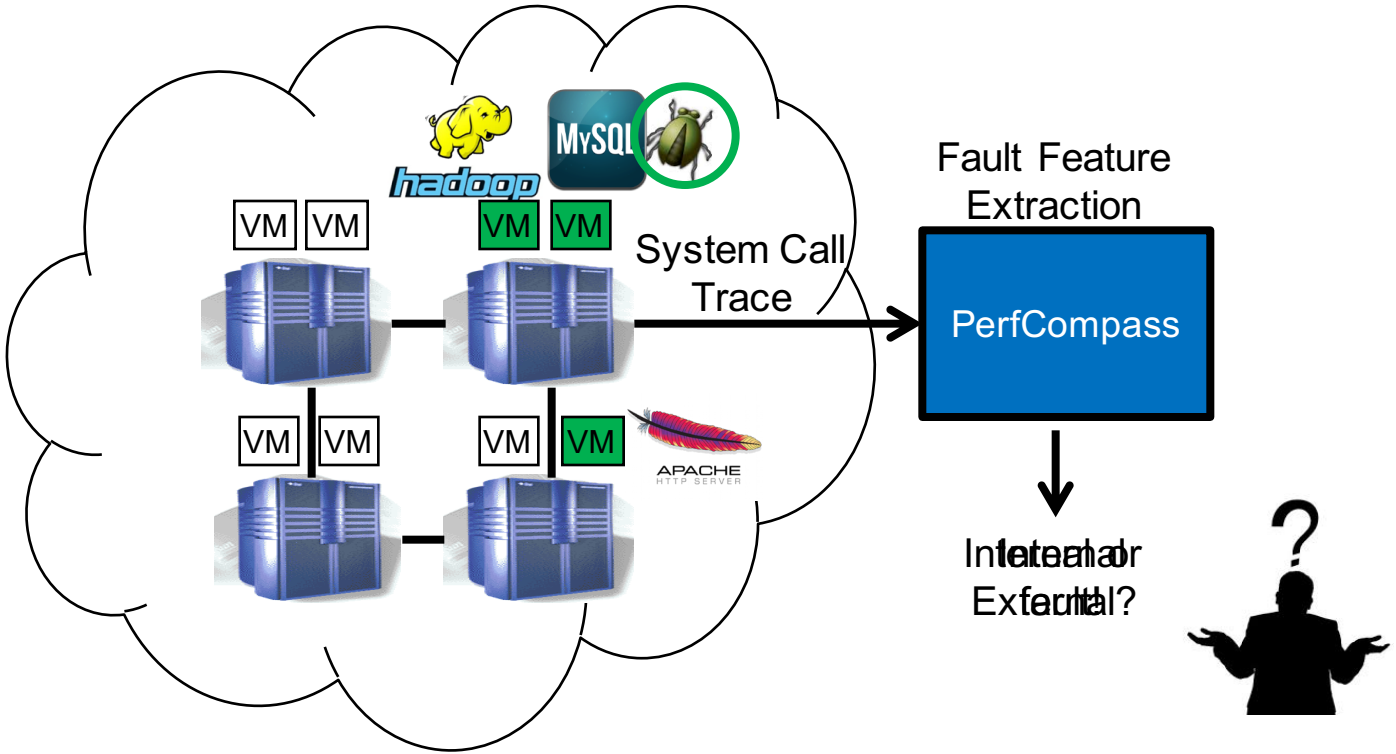




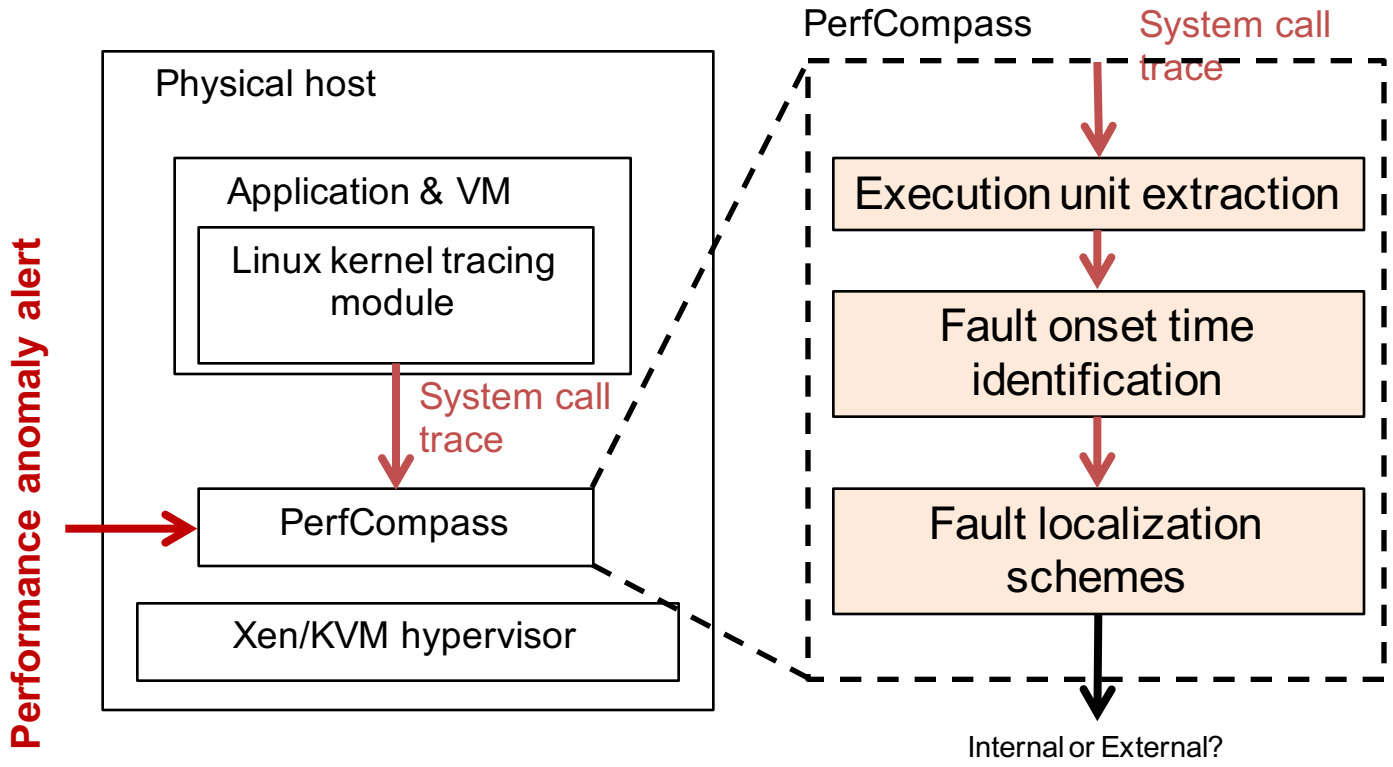
System Overview

PerfCompass	Toward runtime performance anomaly fault localization.
PerfScope	Practical online generic server performance bug inference.
Hytrace	A hybrid approach for diagnosing generic performance bugs.
DScope	Advanced detection for specific performance bugs --- data corruption hang bugs.

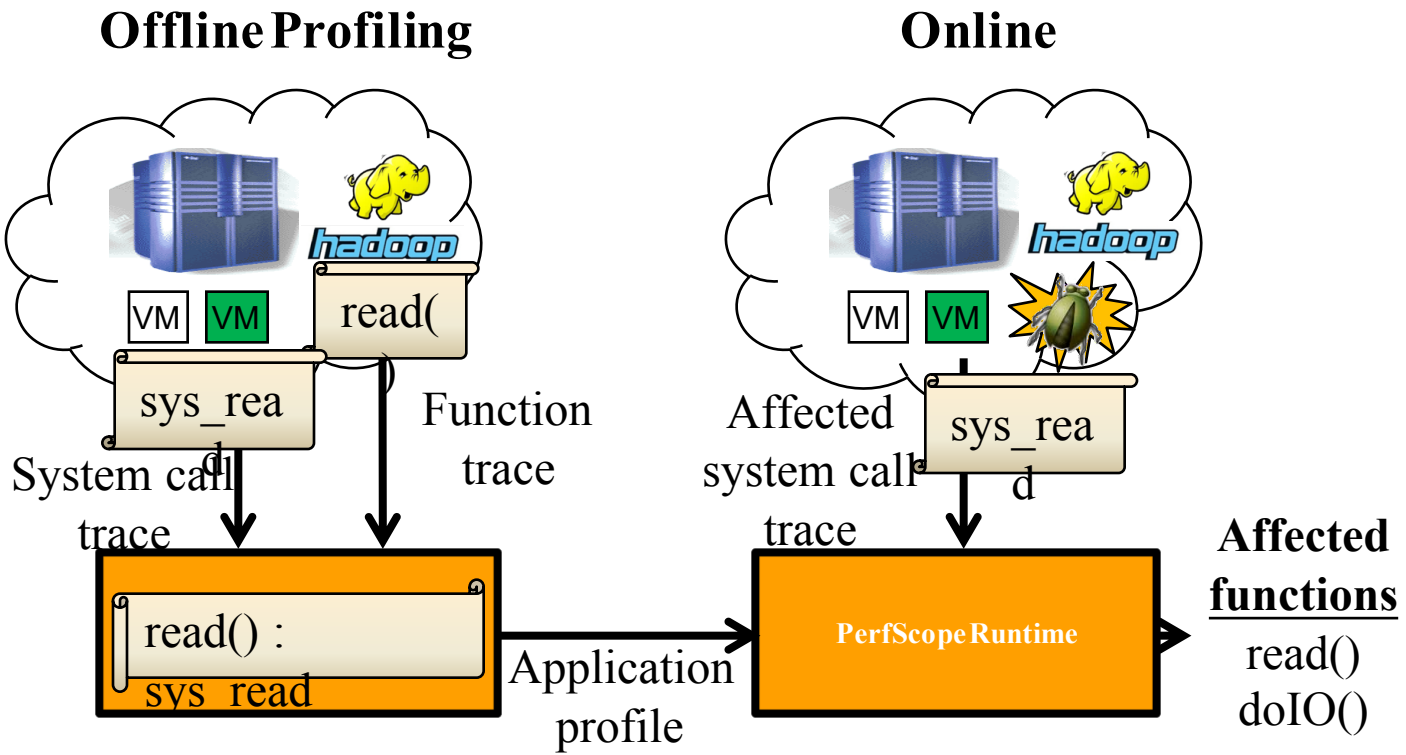
Overview of PerfCompass



Overview of PerfCompass



Overview of PerfScope



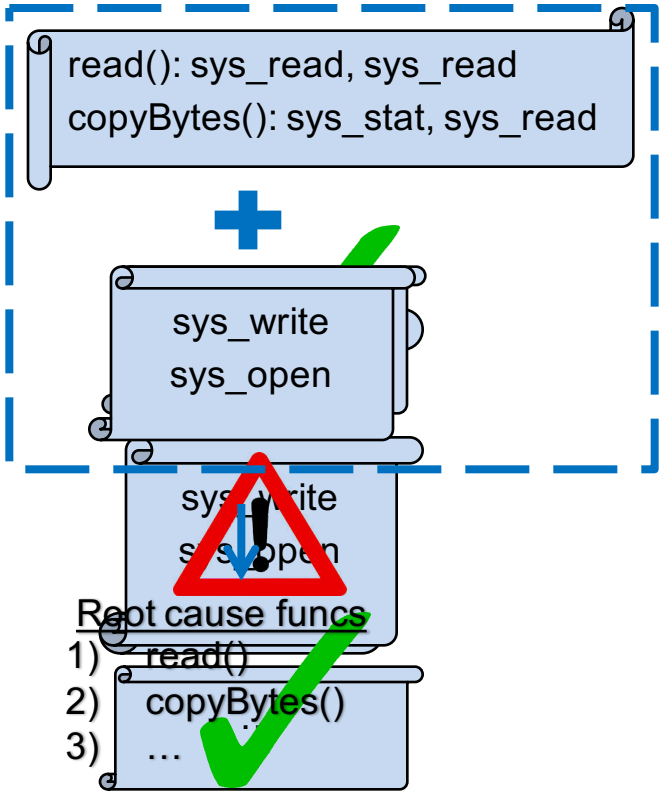
Overview of PerfScope

Offline

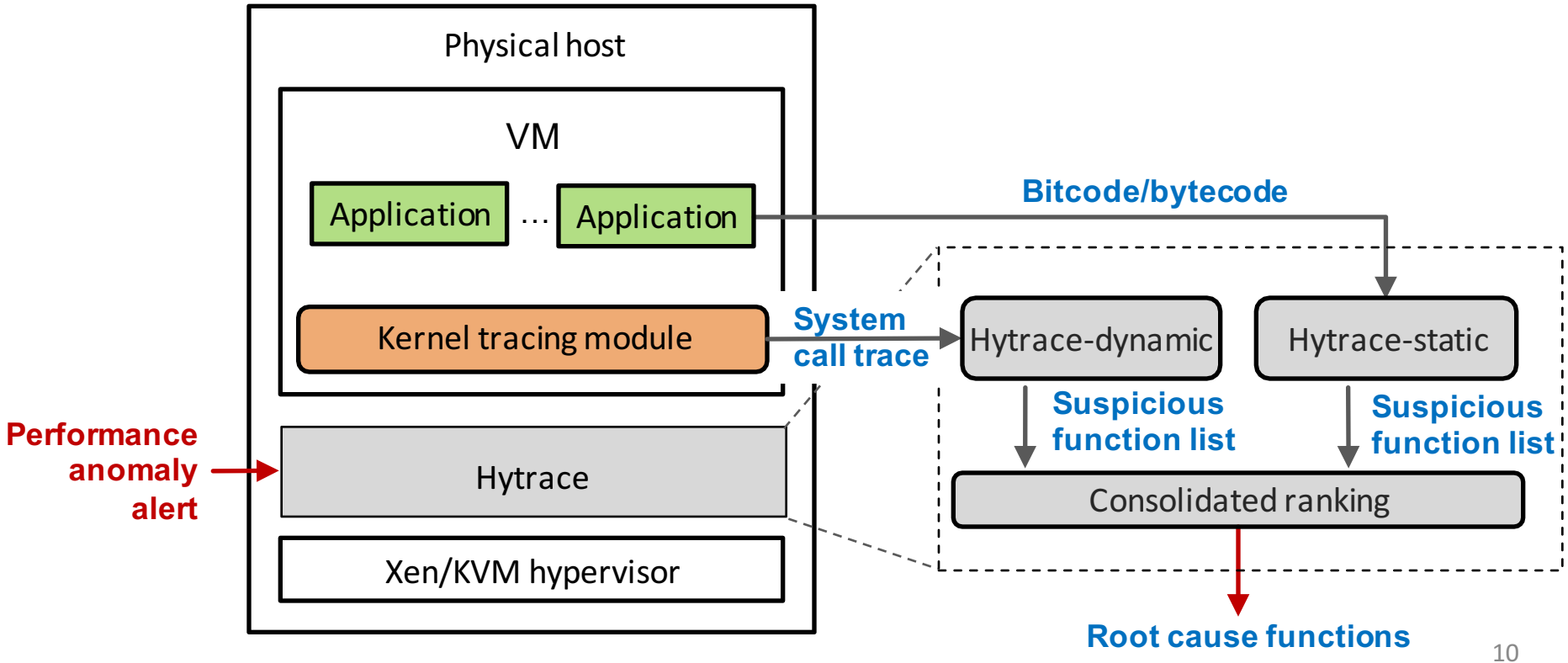
- Binary-based signature profiling

Runtime

- Execution unit extraction
- Abnormal execution unit identification
- Faulty function matching & ranking



Overview of Hytrace



Hytrace Static Analysis

Apache-37680

```
status = apr_socket_opt_set( …, 1);  //non-blocking config

while (! die_now) {
   status = listensocks[offset].accept_func(…);
   …
   else if (status != APR_SUCCESS) {
      continue;
   }
   …
   die_now = 1;
}
```

EAGAIN

- **Rule 1: Constant parameter function calls.**

Hytrace Static Analysis

- Rule 1: Constant parameter function calls.
- Rule 2: Null parameter function calls.
- Rule 3: Unsafe function calls.
- Rule 4: Unchanged loop exit condition variables.
- Rule 5: Uncovered branch.

- Hytrace allows users to easily add new rules with few code changes as long as these rules match our design principle.

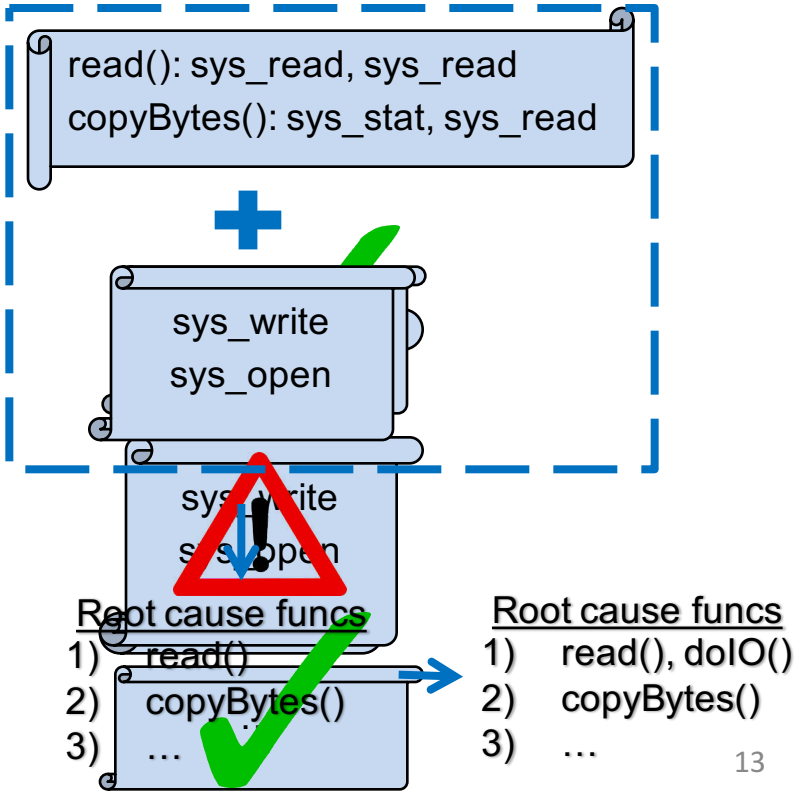
Hytrace Dynamic Analysis

Offline

- Binary-based signature profiling

Runtime

- Execution unit extraction
- Abnormal execution unit identification
- Faulty function matching & ranking
- Candidate function list extension



Consolidated Ranking

- Combine both, high precision & coverage.
- Two components are complementary to each other.
- Program semantic & run-time behavior information.

Evaluation

System	Description	# of bugs
Apache	Apache HTTP server	13
Lighttpd	Web server	7
Memcached	Distributed memory object caching system	1
MySQL	Relational database management system	19
Squid	Caching and forwarding HTTP web proxy	13
Cassandra	Distributed database management system	27
HDFS	Hadoop distributed file system	18
Mapreduce	Hadoop big data processing framework	28
Tomcat	Java Servlet Container	7
Total		133

- **Implemented a prototype of Hytrace using LLVM & Findbugs compiler infrastructures;**
- **Reproduced 14 bugs;**
- **State-of-the-art static/dynamic bug detectors**:
  - **Findbugs (default ruleset)**
  - **Infer (default ruleset)**
  - **Camarel**
  - **PerfScope**

Evaluation (cont.)

Bug name	Description	Symptom
Apache-37680	Make a blocking “accept” call with non-blocking configuration.	**hang**
Apache-43238	Set up new connections with non-keep-alive configuration.	**slowdown**
Apache-45856	Call fopen on file > 2 GB on 32-bit systems.	**hang**
Lighttpd-1212	Keep processing same event when the return value errno is mishandled.	**hang**
Lighttpd-1999	Keep reading and discarding response data while processing header info.	**slowdown**
Memcached-106	Keep reading a non-existent package when the read buffer is overwritten.	**hang**
MySQL-54332	2 threads execute INSERT DELAYED but one of them has a locked table.	**hang**
MySQL-65615	5 x slowdown in the table insertions after truncating a large table.	**slowdown**
Cassandra-5064	ALTER TABLE command keeps flushing empty Memtable.	**hang**
HDFS-3318	HDFS client keeps reading a > 2 GB file when the file length is an integer.	**hang**
Mapreduce-3738	Endless wait for an atomic variable to be set.	**hang**
Tomcat-53450	Tomcat tries to upgrade a read lock to a write lock.	**hang**
Tomcat-53173	Keep dropping incoming requests when the count is improperly updated.	**hang**
Tomcat-42753	Keep processing the same Comet events on a misconfigured req.	**hang**

3 **slowdown** bugs

11 **hang** bugs

True Positive Results

Bug name	Hytrace	Hytrace-dynamic	Hytrace-static	Infer (all)	Infer (perf)	Findbugs (all)	Findbugs (perf)	Caramel	PerfScope
Apache-37680	✓	✓	✓	×	×	-	-	×	✓
Apache-43238	✓	✓	✓	×	×	-	-	×	✓
Apache-45856	✓	✓	✓	×	×	-	-	×	✓
Lighttpd-1212	✓	✓	✓	×	×	-	-	×	✓
Lighttpd-1999	✓	✓	✓	×	×	-	-	×	✓
Memcached-106	✓	✓	✓	-	-	-	-	×	✓
MySQL-54332	✓	✓	✓	-	-	-	-	×	✓
MySQL-65615	✓	✓	✓	-	-	-	-	×	✓
Cassandra-5064	✓	✓	✓	✓	✓	×	×	-	✓
Mapreduce-3738	✓	✓	✓	✓	✓	×	×	-	✓
HDFS-3318	✓	✓	✓	×	×	×	×	-	✓
Tomcat-53450	✓	✓	✓	×	×	×	×	-	✓
Tomcat-53173	✓	✓	✓	×	×	×	×	-	✓
Tomcat-42753	✓	✓	✓	-	-	×	×	-	✓
AVG	100%	100%	100%	14%	14%	0%	0%	0%	100%

False Positive Results

Bug name	Hytrace	Hytrace-dynamic	Hytrace-static	Infer (all)	Infer (perf)	Findbugs (all)	Findbugs (perf)	Caramel	PerfScope
Apache-37680	17	22	40013	18	3	-	-	4	14
Apache-43238	6	12	42273	18	2	-	-	2	8
Apache-45856	5	10	32128	18	2	-	-	4	40
Lighttpd-1212	2	3	4705	181	61	-	-	0	0
Lighttpd-1999	4	4	5057	171	56	-	-	0	1
Memcached-106	2	4	3983	-	-	-	-	0	3
MySQL-54332	6	11	98408	-	-	-	-	22	2
MySQL-65615	2	21	99076	-	-	-	-	8	4
Cassandra-5064	1	8	2982	2904	2904	322	24	-	3
Mapreduce-3738	7	17	9646	5077	5077	1261	170	-	11
HDFS-3318	2	13	10767	2367	2367	1401	168	-	4
Tomcat-53450	8	24	4198	4638	4638	477	53	-	1
Tomcat-53173	15	53	3997	4624	4624	422	51	-	13
Tomcat-42753	2	12	4279			889	238	-	28
AVG	6	15	25822	2002	1973	795	117	5	9

Hytrace Static Rules Coverage Results

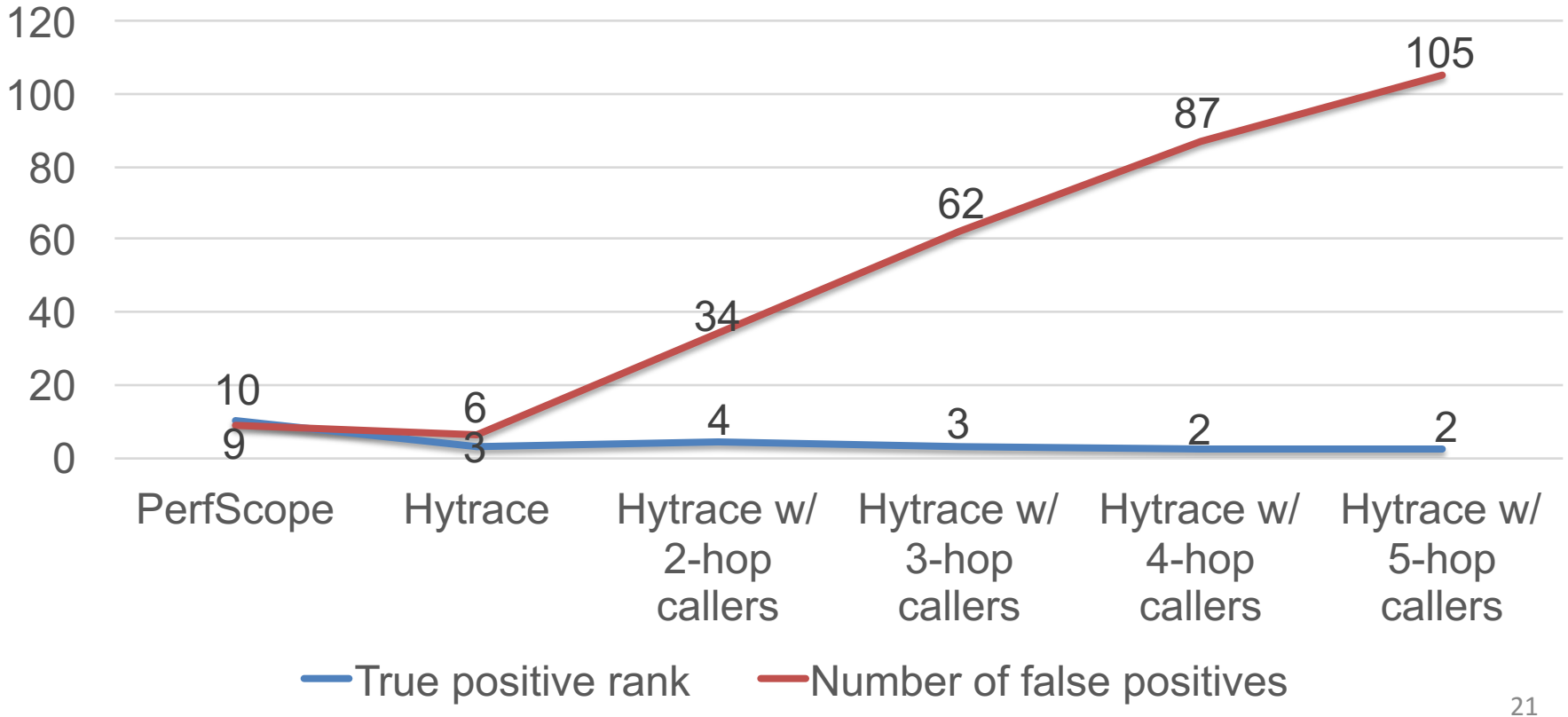
System	Total # bugs	Coverage						# of bugs matched by each rule				
		Hytrace -static	Infer (all)	Infer (perf)	Findbugs (all)	Findbugs (perf)	Caramel	R1	R2	R3	R4	R5
Apache	13	100%	0%	0%	-	-	0%	12	9	2	3	13
Lighttpd	7	100%	0%	0%	-	-	0%	7	2	0	2	6
Memcached	1	100%	0%	0%	-	-	0%	1	1	0	0	1
MySQL	19	100%	11%	5%	-	-	5%	18	18	2	7	17
Squid	13	100%	0%	0%	-	-	0%	13	6	0	1	13
Cassandra	27	100%	44%	44%	0%	37%	-	9	3	-	26	1
HDFS	18	100%	39%	39%	0%	17%	-	13	4	-	17	6
Mapreduce	28	100%	59%	59%	48%	57%	-	21	13	-	26	14
Tomcat	7	100%	43%	43%	14%	43%	-	6	2	-	3	1

Hytrace-static favors generality over precision.

Rank of Root Cause Functions

Bug name	Hytrace						PerfScope Rank
	Rank	R1	R2	R3	R4	R5	
Apache-37680	7	✓	✓	×	×	✓	15
Apache-43238	7	✓	✓	×	×	✓	9
Apache-45856	1	✓	×	✓	×	✓	41
Lighttpd-1212	1	✓	×	×	×	×	1
Lighttpd-1999	2	✓	×	×	×	✓	2
Memcached-106	2	✓	✓	×	×	✓	4
MySQL-54332	2	✓	✓	×	×	×	3
MySQL-65615	2	✓	×	×	×	✓	5
Cassandra-5064	2	✓	×	×	✓	×	4
Mapreduce-3738	4	✓	✓	×	✓	×	12
HDFS-3318	2	✓	×	×	✓	✓	5
Tomcat-53450	1	×	×	×	✓	×	2
Tomcat-53173	10	✓	×	×	×	✓	14
Tomcat-42753	2	✓	✓	×	×	×	29
AVG	3	93%	43%	7%	29%	57%	10

Hytrace with Multiple Hop Callers



Result Summary

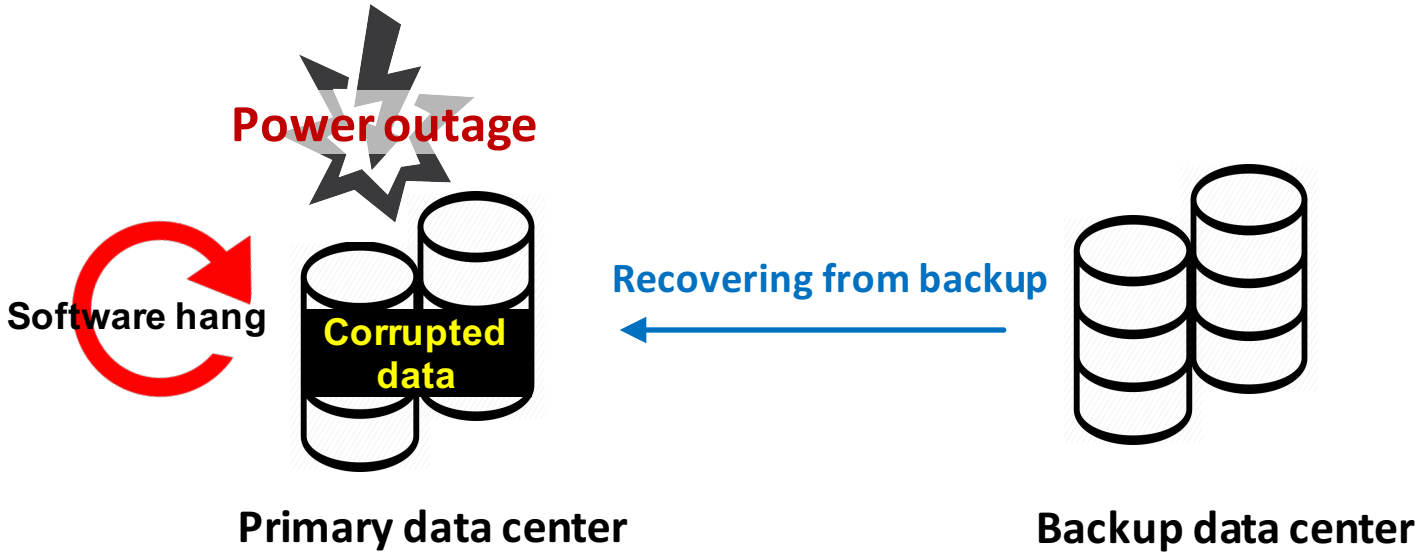
- Hytrace is a hybrid performance diagnosis approach.
  - Combines offline static analysis and online dynamic bug inference.
  - Evaluated over **133** performance bugs on 9 cloud server systems.
  - Reduces FP functions & improve the rank of bug related functions.
  - Imposes less than **3%** CPU overhead to production cloud environments.

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

Real-World Data Corruption Problem

BRITISH AIRWAYS

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



A Data Corruption Hang Bug Example

Hadoop-8614

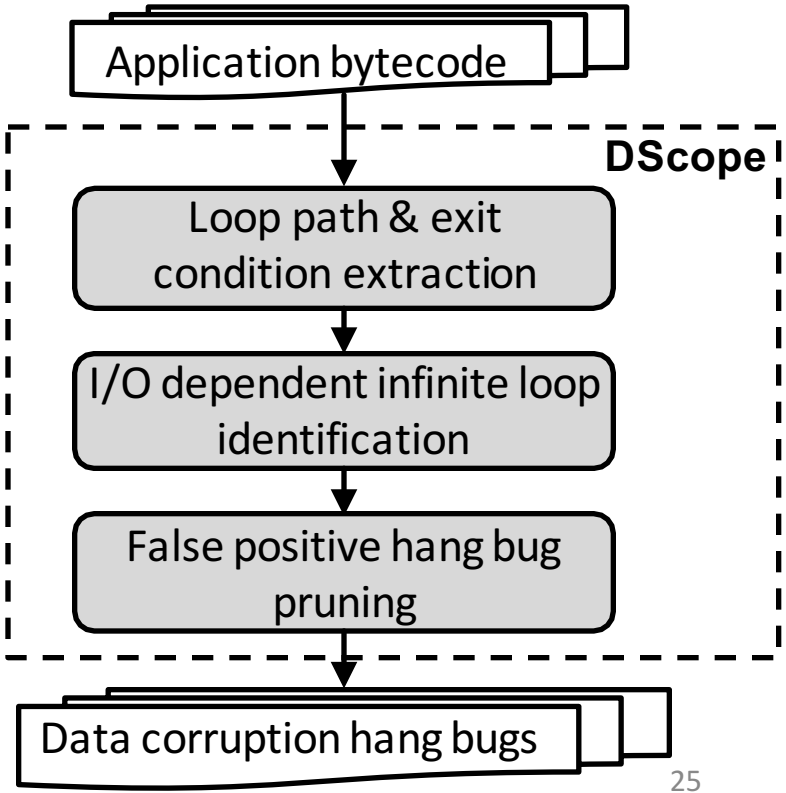
```
183  public static void skipFully(
             InputStream in, long len) … {
184    while (len > 0) {
185      long ret = in.skip(len);
           …
           …
189      len -= ret;
190    }
191  }
```

Corrupted InputStream

0

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

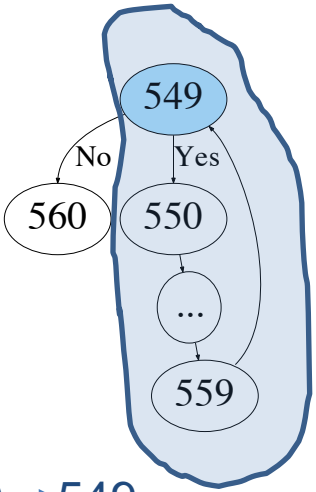
Overview of DScope



Loop Path & Exit Condition Extraction

- **Simple Loops**

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

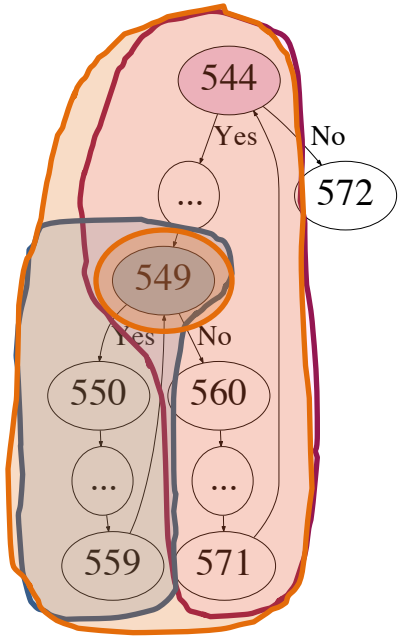


Loop path: 549→ 550→ … →559 →560 →549

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.

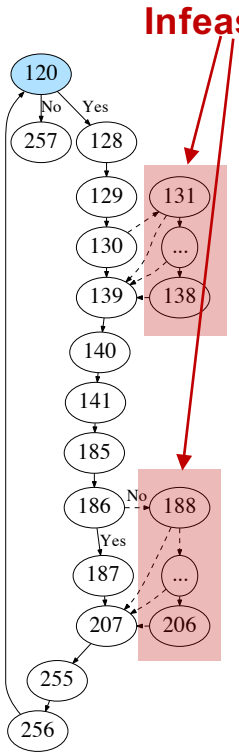
Loop Path & Exit Condition Extraction

- **Loops with exception handling**

```
120  while (!dataFile.isEOF()){
       …
129    try {                          Corrupted dataFile
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
       } }
```

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

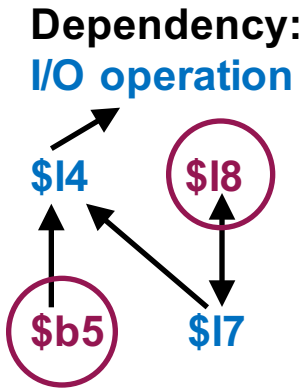
- **Exit conditions directly depend on I/O operations**

```
//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

```
307  public static long readVLong(DataInput stream)…{
308      byte firstByte = stream.readByte();
309      int len = decodeVIntSize(firstByte);
           …
314      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

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

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

```
194 BUFFER_SIZE = conf.getInt(); Corrupted configuration file

78  private void readLocalFile(Path path, ...) … {
        ...
84      byte[] data = new byte[BUFFER_SIZE];
85      long size = 0;
86      while (size >= 0) {
87          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

```
1668   while (!fileComplete) {
1669       fileComplete = dfsClient.namenode.complete(src,
                                dfsClient.clientName, last);
           ...
1689   }
```

Corrupted block

False Positive Example

Hadoop v2.5 BlockReaderLocal.java

```
472  private int readWithBounceBuffer(
                          ByteBuffer buf…) …{
481     do {
          …
502       bb = drainDataBuf(buf);
512     } while (buf.remaining() > 0);
          …
514  }
```

Check bounds

```
277 private int drainDataBuf(
                  ByteBuffer buf) {
          …
286     buf.put(dataBuf);
          …
291 }
```

Forward index

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

Result Summary

- 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.