

Automating Failure Diagnosis for Distributed Systems

Yongle Zhang
Purdue University



Most Internet services live on the cloud

**Google cloud is down, affecting
Amazon AWS S3 outage is breaking**

**Apple's iCloud recovers after a
four-hour outage**

zdnet.com

September 17, 2018

**Microsoft provides preliminary
report on its September 4 cloud
outage**

Ken Yeung / The Next Web:

**Dropbox says its service is back up and running
outage (Updated)** — Update 2 - 12 January: Dropbox now

npr.org

March 1, 2017

**Amazon Cloud Outage Disrupts
Traffic For Websites, Apps**

Karissa Bell / Mashable:

Adobe Creative Cloud Has Been Down for Almost

Creative Cloud is in the midst of a massive outage that has lasted for nearly 24 hours, impacting affected users throughout the United States, Europe and Asia. — Subscribers to the Creative Cloud plan, which includes Photoshop ...

**Cloudflare blames 'bad software'
deployment for today's outage**

Office 365, Azure users are locked

**VMware Joins Cloud Outage Party With Cloud
Foundry Blackout**

Yahoo Mail Takes Big Hit In Cloud Outage

**Another Cloud Outage Strikes Microsoft BPOS,
Exchange Online**

**Google Docs Goes Dark In Evening Cloud
Outage**

**New Amazon Cloud Outage Takes Down Netflix,
Foursquare**

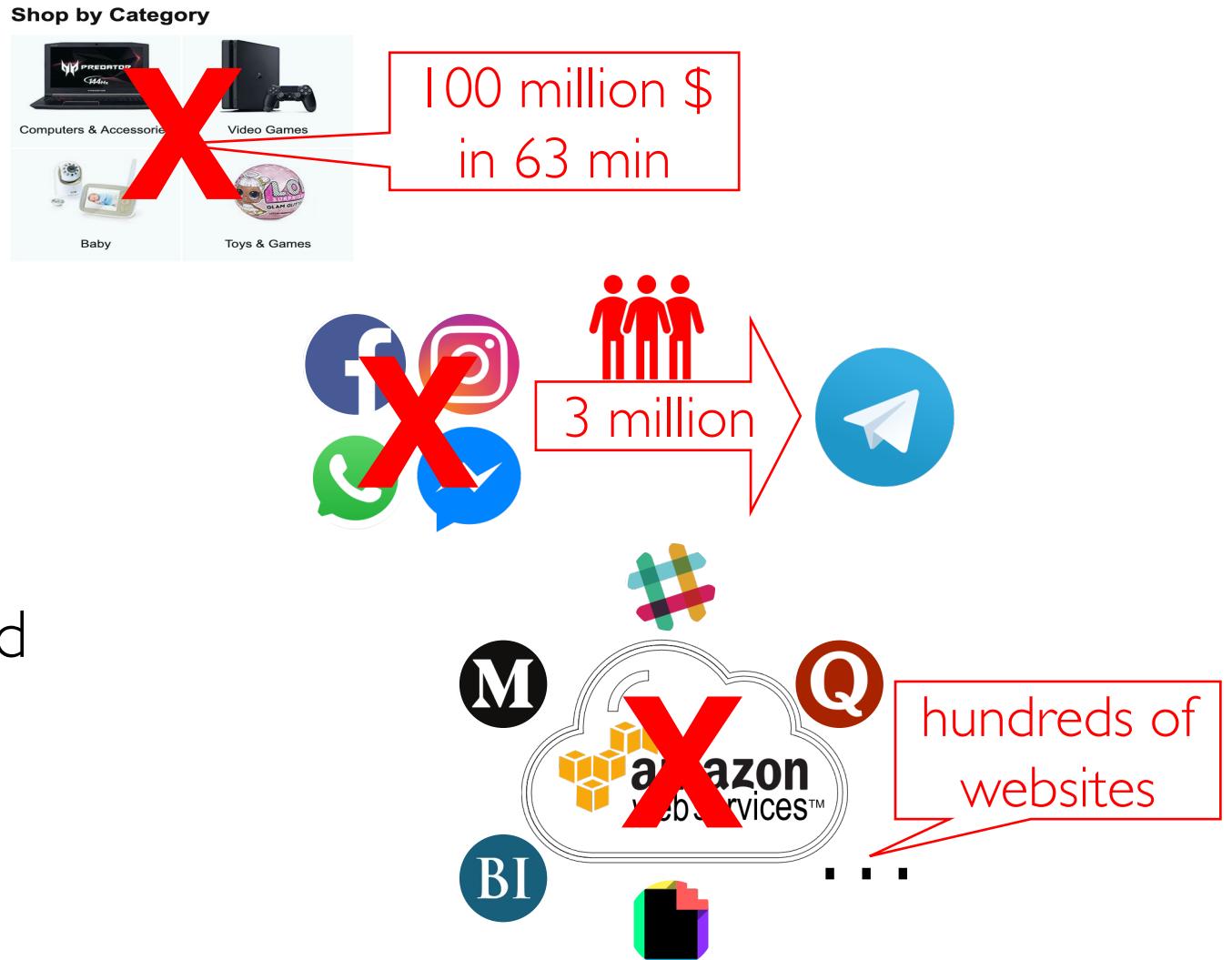
**Amazon EC2 Goes Dark In Morning Cloud
Outage**

Tom Warren / The Verge:

**YouTube, Snapchat, Gmail, Discord, and many other
services are down throughout US during a major Google Cloud service outage** — YouTube, Snapchat, Gmail, Netflix, and a number of other web services suffered from major outages in the US to

Cloud failures have severe consequences

- Significant cost
 - E.g., 2018 Amazon Primeday^[1]
- Frustrates & loses users
 - E.g., 03/13/2019 Facebook^[2]
- Disrupts all services on the cloud
 - E.g., 02/2017 AWS^[3]



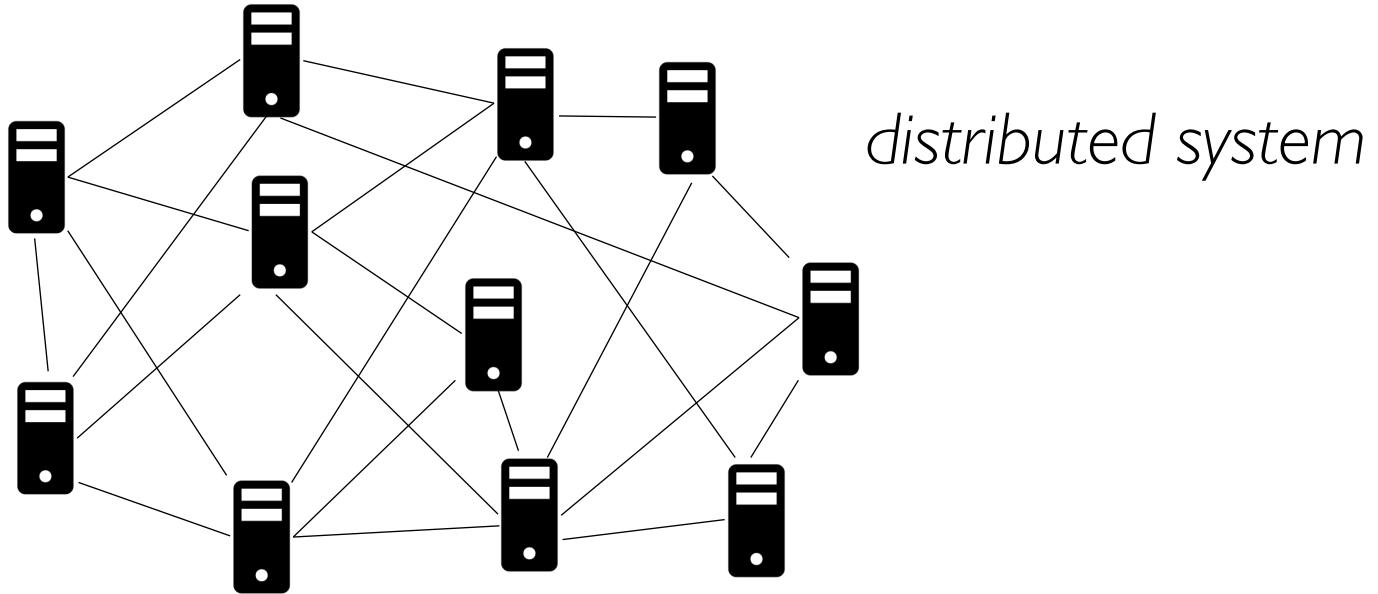
[1]: <https://www.cbronline.com/news/amazon-outage-lost-sales>

[2]: <https://techcrunch.com/2019/03/14/telegram-gets-3m-new-signups-during-facebook-apps-outage/>

[3]: <https://venturebeat.com/2017/02/28/aws-is-investigating-s3-issues-affecting-quora-slack-trello/>

Failure diagnosis is important

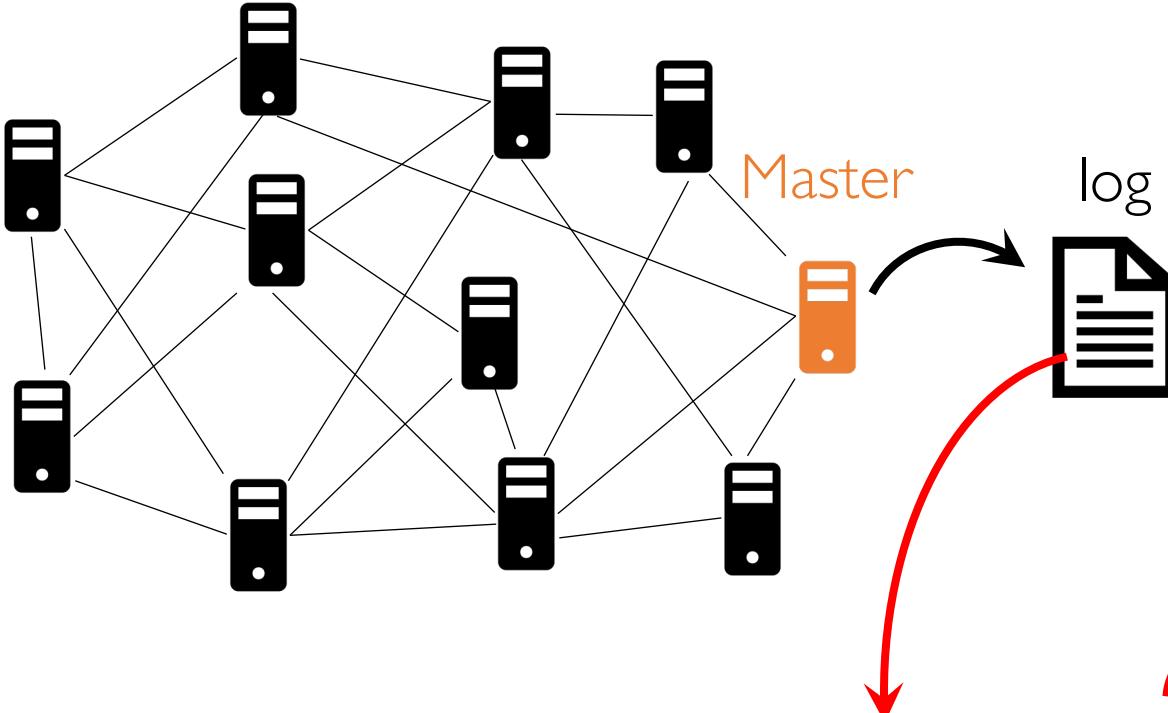
Failure diagnosis for cloud systems is hard



Diagnosing HDFS-10453 took a month

- Step 2: locate the root cause
- Challenge: complex failure execution
- Tried numerous rounds of printf-debugging → took 1 month

Hadoop distributed file system



```
void replicateBlock(Block blk) {  
    int numReplicas = blk.getNumReplicas();  
    int numEffectiveReplicas = blk.getEffectiveReplicas();  
    List<DataNode> containingNodes = blk.getContainingNodes();  
    BlockReplicationWork rw = new BlockReplicationWork(blk);  
    int scheduledWork = 0;  
    synchronized (rw) {  
        if (numReplicas > numEffectiveReplicas) {  
            for (DataNode dn : blk.getReplicaNodes()) {  
                BlockReplicationWork item = rw.getReplica(dn);  
                if (item != null) {  
                    if (item.isUnderConstruction() || item.getBlock().equals(blk)) {  
                        numEffectiveReplicas++;  
                        item.setPriority(scheduledWork++);  
                    }  
                }  
            }  
        }  
        requiredReplication = blk.getBlockReplication();  
        chosenSourceDataNode = chooseSourceDataNode(requiredReplication, rw);  
        chosenReplicaNodes = chooseReplicaNodes(chosenSourceDataNode, requiredReplication, numReplicas);  
        if (chosenReplicaNodes == null) {  
            if (blk.isCorrupt()) {  
                log.error("Block " + blk + " cannot be replicated from any node");  
            }  
            continue;  
        }  
        // livereplicasNodes can include READONLY or SHARED replicas, which are  
        // used for liveverifications. If enough replicas are available, then  
        // numEffectiveReplicas = numReplicas. Otherwise, numEffectiveReplicas < numReplicas.  
        if (numEffectiveReplicas < requiredReplication) {  
            numReplicas -= requiredReplication - numEffectiveReplicas;  
            neededReplications = removeReplicas(blk, effectiveReplicas);  
            if (neededReplications > 0) {  
                log.error("Failed to replicate block " + blk + " because it needs " +  
                         numReplicas + " replicas but only " + numEffectiveReplicas + " are available");  
            }  
        }  
        if (numEffectiveReplicas < requiredReplication) {  
            if (numReplicas < numEffectiveReplicas) {  
                else {  
                    if (numReplicas < numEffectiveReplicas) {  
                        numReplicas = numEffectiveReplicas; // Needed on a new rack  
                        rw.addNewReplica(blk, chosenReplicaNodes, numReplicas);  
                    }  
                }  
            }  
        }  
    }  
    finally {  
        nameSystem.writeLock();  
    }  
    log.debug("collected replication work");  
    final ReplicationWorkNodes rw = new ReplicationWorkNodes();  
    if (distributedFileSystem.replicatedBlockCount() >= numReplicas) {  
        DataNode dn = distributedFileSystem.getLocalDataNode();  
        while (distributedFileSystem.getReplicaCount(dn) == numReplicas) {  
            DataNode dn2 = distributedFileSystem.chooseTargetingNodes(dn);  
            if (dn2 != dn) {  
                for (DataNode dn3 : dn2.getReplicaNodes()) {  
                    log.debug("chooseTargets for block " + blk + " target " + dn3);  
                    chooseReplicaNodes(blk, dn3);  
                    if (numReplicas > 0) {  
                        log.error("HDFS-10453: THE GLOBAL_LOCK IS NOT HELD AND THE chooseReplicaNodes is called, " +  
                                 "rw.chooseReplicaNodes(block, target) is triggered");  
                    }  
                }  
            }  
        }  
    }  
}
```

if (numNeeded > 0)
Log.error("Failed to replicate.");

2019-02-08 16:19:17,126 ERROR Failed to replicate.

HDFS-I 0453 root cause

ReplicationMonitor thread

```
1. void replicateBlock(Block blk) {  
2.     int numNeeded = 3 - blk.nReplica;  
3.     while(numNeeded > 0) {  
4.         node = chooseNextRandomNode(..);  
5.         if (blk.size <= node.capacity){  
6.             ... // replicate b to this node  
7.             numNeeded--;  
8.         }  
9.     }  
10.    if (numNeeded > 0)  
11.        Log.error("Failed to replicate.");  
12. }
```

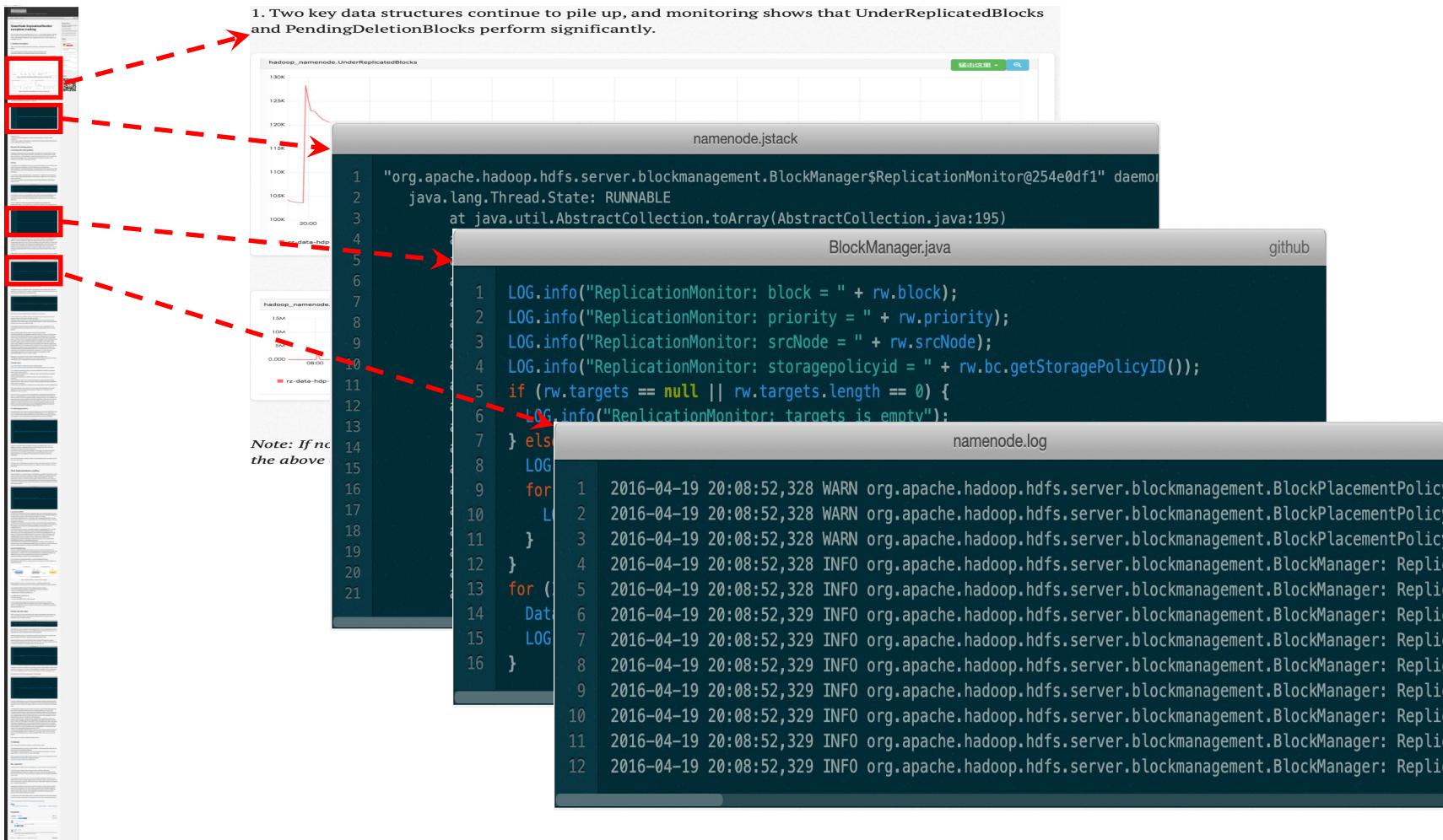
DeleteBlock thread

```
13. void deleteBlock(Block blk) {  
14.     blk.size = Long.MAX_VALUE;  
15.     FileSystem.remove(blk);  
16. }
```

- A data race

Diagnosing HDFS-10453 took a month

- Developers got frustrated & wrote a blog^[1]



[1]: NameNode ReplicationMonitor anomaly diagnosis, <https://hexiaoqiao.github.io/blog/2016/09/13/namenode-replicationmonitor-exception-trace/>

Automating diagnosis

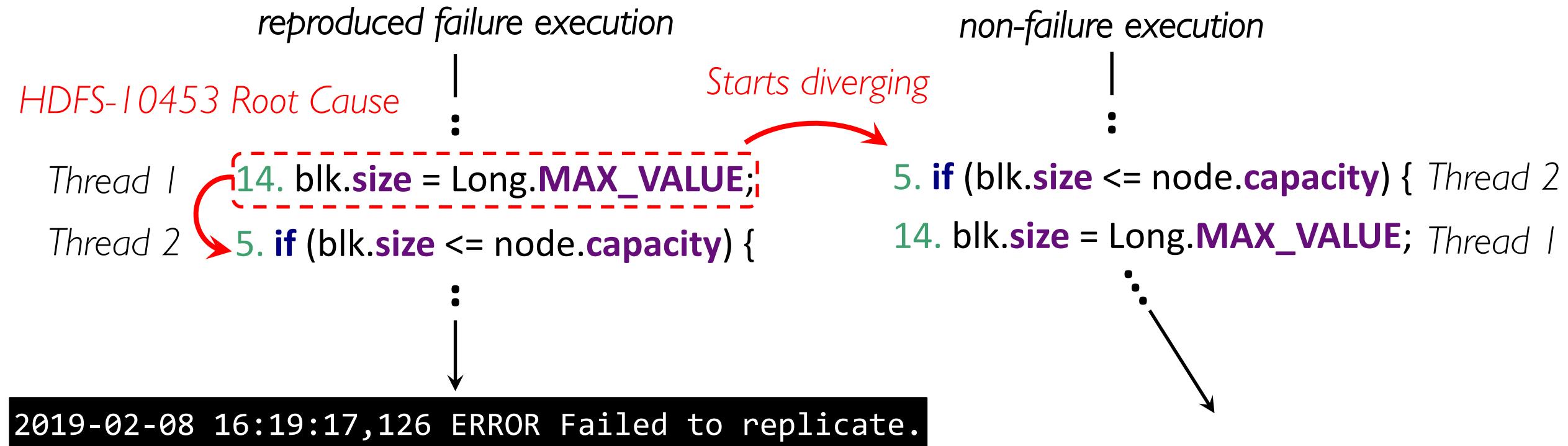
- Pensieve [SOSP'17]: automates failure reproduction
 - Input: logs + bytecode + symptom
 - Output: **minimum** steps to reproduce failures

```
initCluster(4);  
create("a.txt",3);  
stopDataNode(1);  
delete("a.txt");
```

HDFS-10453 reproduction

Automating diagnosis

- Kairux [SOSP'19]: automates root cause localization
 - reproduced execution + symptom → root cause + non-failure execution



“software bugs - the most common cause of cloud incidents” [Liu et al., HotOS’2019]

Approach & impact

Approach ➤ Replicate how developers diagnose failures

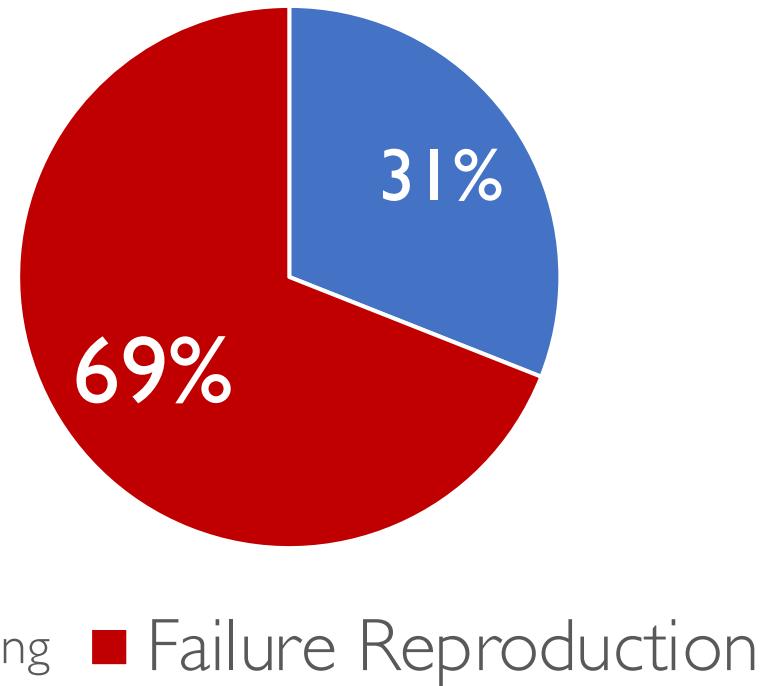
- Exploits human intuitions and principles
- **Scales** diagnosis techniques to distributed systems

Outline

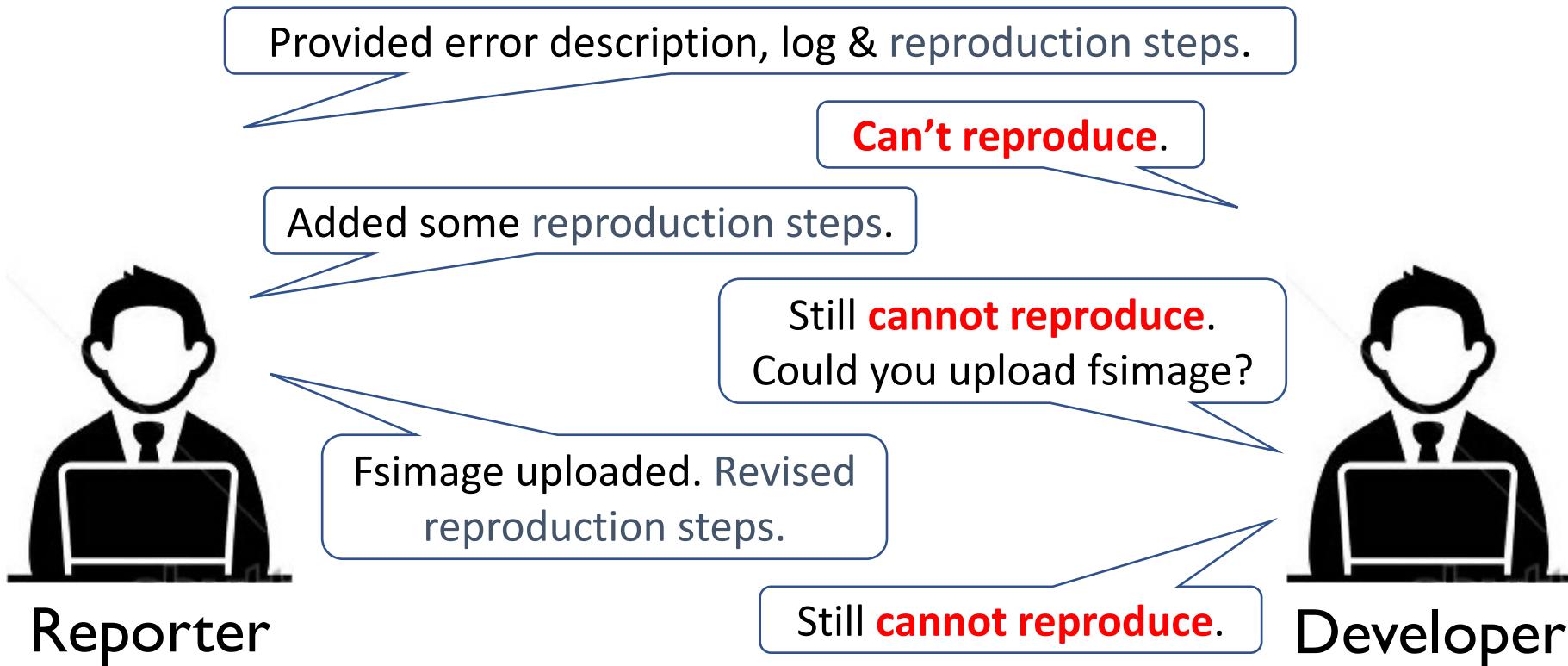
- Introduction
- Pensieve [SOSP'17]: automates failure reproduction
 - logs + bytecode + symptom → minimum steps to reproduce failures
- Kairux [SOSP'19]: automates root cause localization
- Bigger picture

Failure reproduction is time-consuming

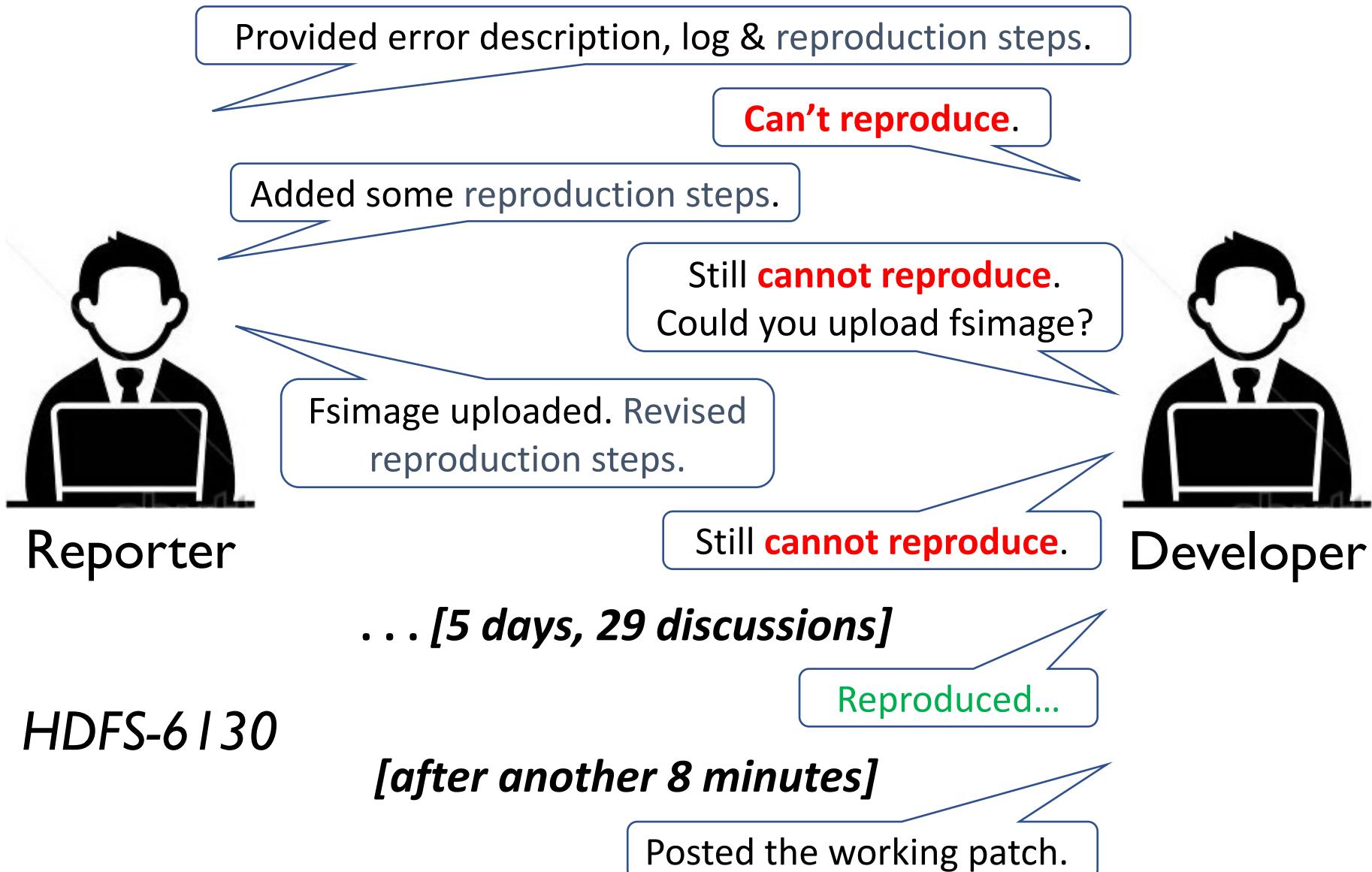
Debugging Time for
60 HDFS, HBase, ZooKeeper Bugs



A Real-World Debugging Procedure



A Real-World Debugging Procedure



Existing Solutions Are Limited

- Record-and-replay (deterministic replay)
 - Intrusive: modifies existing software stack
 - Incurs performance overhead
- Symbolic Execution
 - E.g., ESD [Zamfir EuroSys'10], SherLog [Yuan ASPLOS'10].
 - Pros: precise & non-intrusive
 - Cons: hard to scale to large systems

Scalability of Symbolic Execution

```
1. for(int i=0;i<blocks.length;i++){  
2.   if(blocks[i].genStamp!=VALID_GS)  
3.     log("invalid block. . .");  
4. }
```

Simplified code snippet for HDFS-4022

Scalability of Symbolic Execution

```
1. for(int i=0;i<blocks.length;i++){  
2.   if(blocks[i].genStamp!=VALID_GS)  
3.     log("invalid block. . .");  
4. }
```

Simplified code snippet for HDFS-4022

- Enumerates every possible execution path

Scalability of Symbolic Execution

```
1. for(int i=0;i<blocks.length;i++){  
2.   if(blocks[i].genStamp!=VALID_GS)  
3.     log("invalid block. . .");  
4. }
```



Simplified code snippet for HDFS-4022

- Enumerates every possible execution path

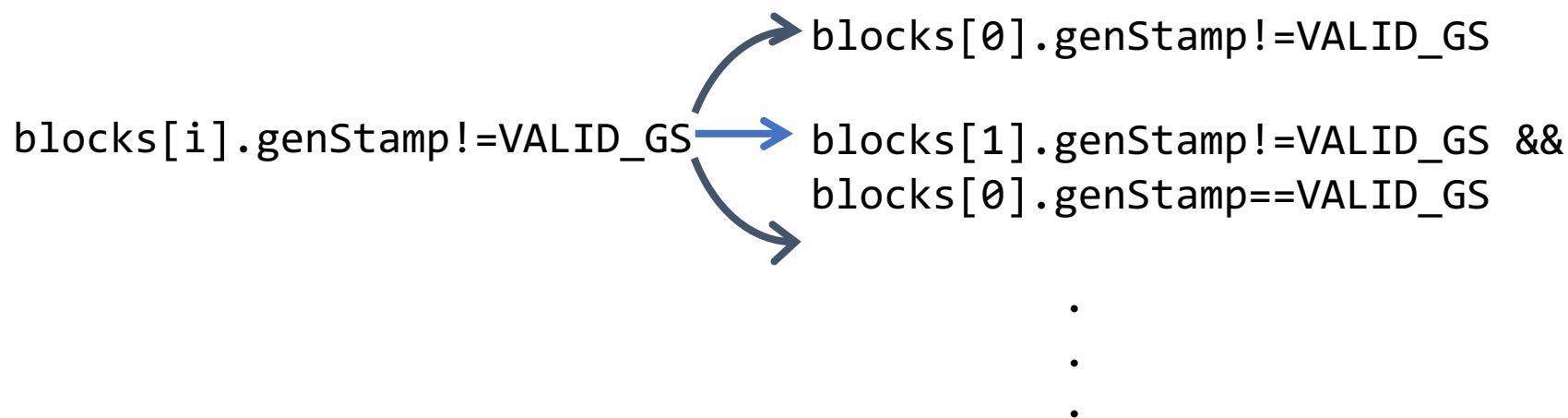
blocks[i].genStamp!=VALID_GS

Scalability of Symbolic Execution

```
1. for(int i=0;i<blocks.length;i++){  
2.   if(blocks[i].genStamp!=VALID_GS)  
3.     log("invalid block. . .");  
4. }
```

Simplified code snippet for HDFS-4022

- Enumerates every possible execution path



Scalability of Symbolic Execution

```
1. for(int i=0;i<blocks.length;i++){  
2.   if(blocks[i].genStamp!=VALID_GS)  
3.     log("invalid block. . .");  
4. }
```

Simplified code snippet for HDFS-4022

- Enumerates every possible execution path

blocks[0].genStamp!=VALID_GS
OR
blocks[i].genStamp!=VALID_GS → blocks[1].genStamp!=VALID_GS &&
blocks[0].genStamp==VALID_GS
OR
.
.
.

Scalability of Symbolic Execution

```
1. for(int i=0;i<blocks.length;i++){  
2.   if(blocks[i].genStamp!=VALID_GS)  
3.     log("invalid block. . .");  
4. }
```

Simplified code snippet for HDFS-4022

- Enumerates every possible execution path

<i>Failure</i>	<i>Total Branch Instructions</i>	<i>S.E. Stopped At (Instructions)</i>	<i>Condition Size</i>	<i>Pensieve instruction</i>
HDFS-4022	72,943,652	693	109,018,324	166

Core Idea – Partial Trace Observation

Developers almost never debug a failure by reconstructing its complete execution path.

Instead, they construct a simplified trace which only contains events that are likely to be causally relevant to the failure.

How do developers debug HDFS-4022?

```
for(int i=0;i<blocks.length;i++)  
    if(blocks[i].genStamp!=VALID_GS)  
        log("invalid block. . .");
```



How do developers debug HDFS-4022?

```
void rollLog(. . .){  
    b.genStamp=logGS;  
}
```

```
if(this.stage==APPEND){  
    log("Appending" + b);  
    b.genStamp=newGS;  
}
```

```
for(int i=0;i<blocks.length;i++)  
    if(blocks[i].genStamp!=VALID_GS)  
        log("invalid block. . .");
```



How do developers debug HDFS-4022?

```
void rollLog(. . .){  
    b.genStamp=logGS;  
}
```

```
if(this.stage==APPEND){  
    log("Appending" + b);  
    b.genStamp=newGS;  
}
```

```
for(int i=0;i<blocks.length;i++)  
    if(blocks[i].genStamp!=VALID_GS)  
        log("invalid block. . .");
```

How do developers debug HDFS-4022?

```
if(this.stage==APPEND){  
    log("Appending" + b);  
    b.genStamp=newGS;  
}
```

```
for(int i=0;i<blocks.length;i++)  
    if(blocks[i].genStamp!=VALID_GS)  
        log("invalid block. . .");
```

How do developers debug HDFS-4022?

```
if(this.stage==APPEND){  
    log("Appending" + b);  
    b.genStamp=newGS;  
}
```

```
for(int i=0;i<blocks.length;i++)  
    if(blocks[i].genStamp!=VALID_GS)  
        log("invalid block. . .");
```

How do developers debug HDFS-4022?

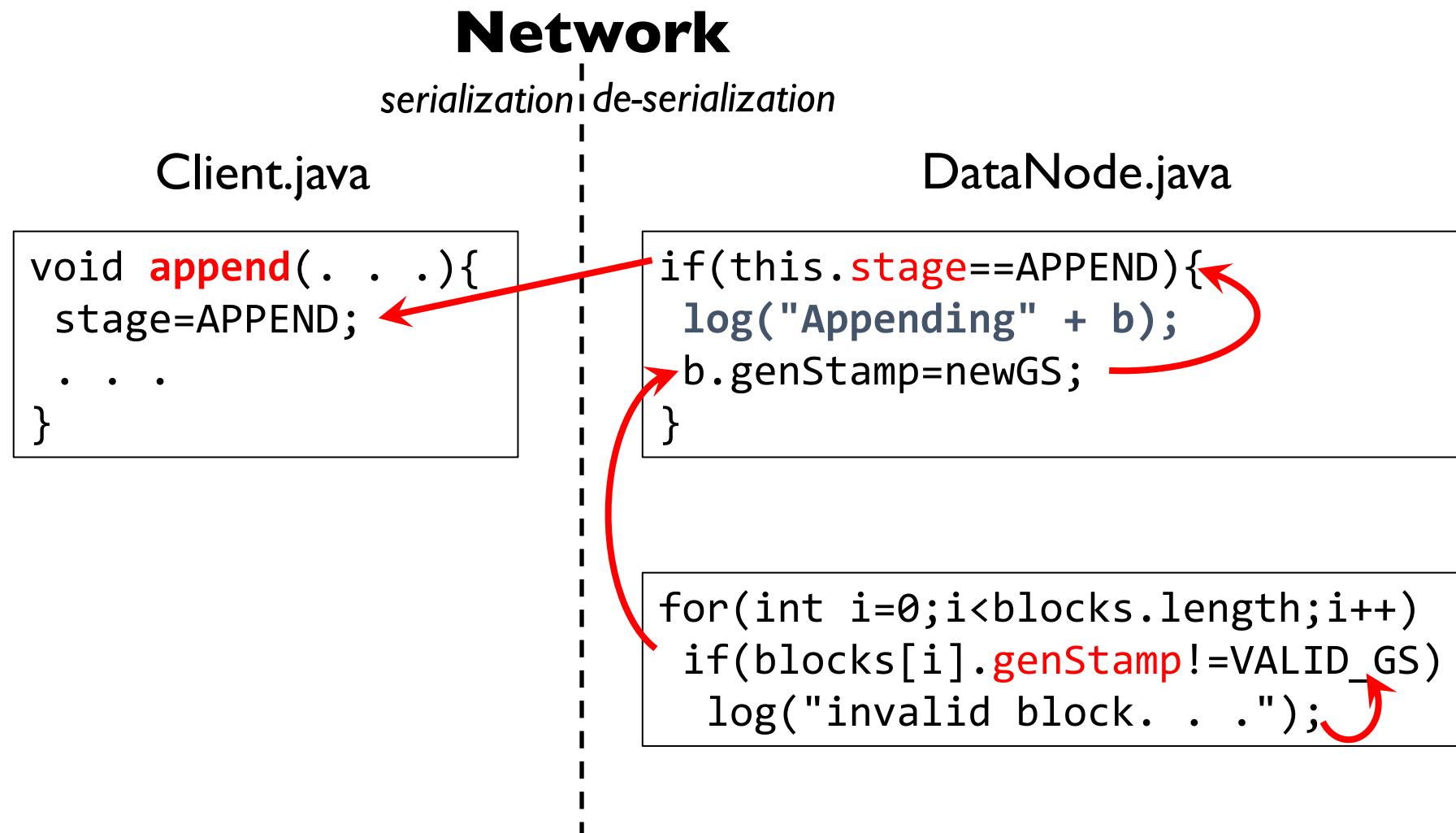
Client.java

```
void append(. . .){  
    stage=APPEND; ←  
    . . .  
}
```

```
if(this.stage==APPEND){  
    log("Appending" + b);  
    b.genStamp=newGS;  
}
```

```
for(int i=0;i<blocks.length;i++)  
    if(blocks[i].genStamp!=VALID_GS)  
        log("invalid block. . .");
```

How do developers debug HDFS-4022?



How do developers debug HDFS-4022?

Network

serialization | de-serialization

Client.java

```
void append(. . .){  
    stage=APPEND; ←  
    . . .  
}
```

DataNode.java

```
if(this.stage==APPEND){  
    log("Appending" + b);  
    b.genStamp=newGS;  
}
```

```
for(int i=0;i<blocks.length;i++)  
    if(blocks[i].genStamp!=VALID_GS)  
        log("invalid block. . .");
```

Got one user command
(append) by looking at **8**
instructions!

Pensieve: automated failure reproduction

Command Line Input (HDFS-4022)

```
./pensieve
-jar ./hadoop-hdfs-2.0.0-alpha.jar // Java bytecode
-log ./HDFS-logs/                // failure logs
-error ./HDFS-logs/dn2.log#800    // symptoms
```

dn2.log



800: 2019-02-08 16:19:17,126 ERROR invalid block



Output: reproduction

```
initCluster(1);                      // start 1-machine cluster
create("a.txt",2);                   // create 2-replica file
append("a.txt","X");                // append to file
addDataNode();                      // add a datanode on the fly
```

Evaluation of Pensieve

- Evaluated on 18 cases from JVM distributed systems
 - HDFS, HBase, ZooKeeper, Cassandra
 - with noisy logs generated from manual reproduction
- Overall Result
 - Successfully reproduces 72%

Case study: HDFS-4022

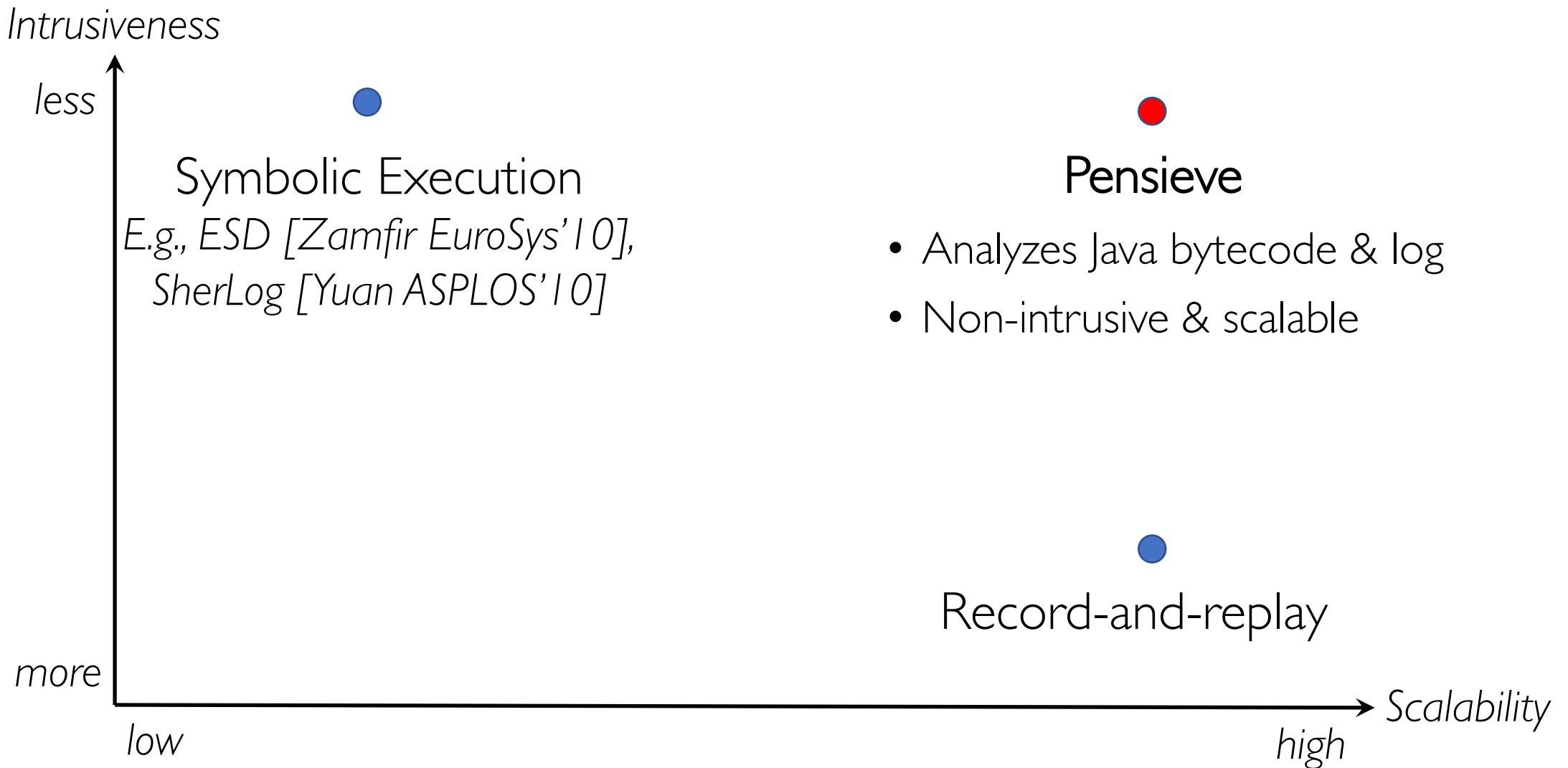
```
initCluster(4);
create("a.txt",3);
stopDataNode(1);
append("a.txt","data");
```

Developers' reproduction

```
initCluster(3);
setConfig("policy", "ALWAYS");
create("a.txt",2);
stopDataNode(1);
append("a.txt","data");
```

Pensieve's reproduction – **fewer nodes!**

Related work: failure reproduction



Pensieve

- Partial Trace Observation: the natural way human beings reproduce failures

Developers construct a simplified trace which only contains events that are likely to be causally relevant to the failure.

- Pensieve: automated failure reproduction
 - Scales to real-world distributed systems
 - Non-intrusive & relies on logs

Outline

- Introduction
- Pensieve [SOSP'17]: automates failure reproduction
- Kairux [SOSP'19]: automates root cause localization
 - Assumption: failure is reproduced
 - Answers “what is a root cause?”
- Bigger picture

Kairux: root cause localization

- Input
 - reproduced failure execution e_f
- Output
 - the location of one single root cause in e_f
 - a non-failure execution that avoids the root cause

Existing works are neither precise nor scalable

Understandability (execution context that explains the root cause)

Probabilistic approach
Execution searching
E.g., spectrum-based [Jones et al., ICSE'02],
E.g., Mage [Lucek et al., SOSP'07],
statistical debugging [Hiblit et al., PLDI'05],
BugEx [Robler et al., PSSA'12],
Holmes [Johnson et al., ICSE'20]

```
void func(...) { search space
    if (a1 > 0) {...} T/F
    ...
    if (an > 0)      T/F
    Log.error("ERROR");
}
```

- Instruments program to record predicates
- Based on a precise, scalable definition of root cause
- Finds single root cause location with minimal differences
- Reduces search space, scales to distributed systems
- Root cause candidates: strongest statistical differences
- Explains the root cause

10. if (numNeeded > 0):
11. Delta debugging
Log.error("Failed to replicate.");

- Predicate-switching
- [Zhang et al., ICSE'06]

→ Scalability

What is a root cause?

What is the fundamental property of a root cause that allows us to build a tool to automatically search for it?

What is a root cause?

The most basic reason for a failure
which, if corrected, would have
prevented the failure from occurring.

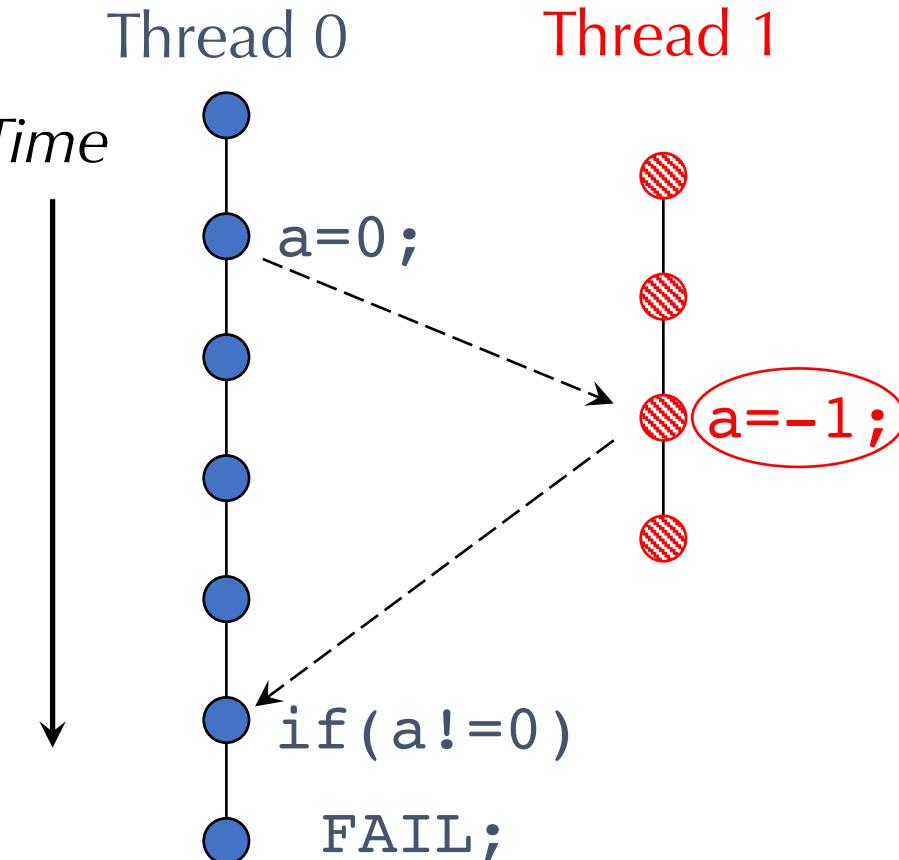
– Wilson et al. ASQ Quality Press 1993

What is a root cause?

- 1 The root cause is an instruction in the failure execution which, if changed, would have resulted in a correct execution.

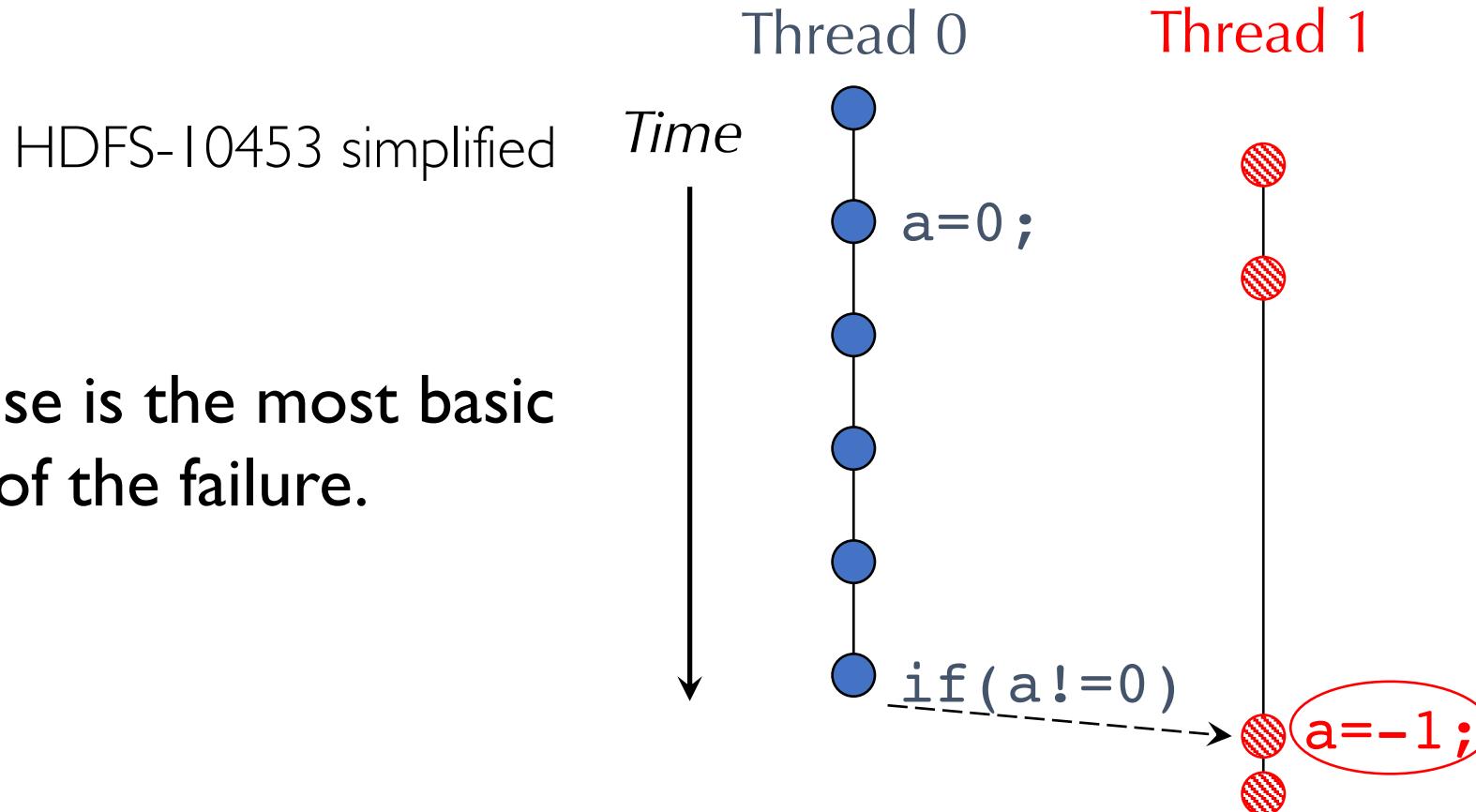
HDFS-10453 simplified

- 2 The root cause is the most basic cause of the failure.



What is a root cause?

- 1 The root cause is an instruction in the failure execution which, if changed, would have resulted in a correct execution.



- 2 The root cause is the most basic cause of the failure.

What is a root cause?

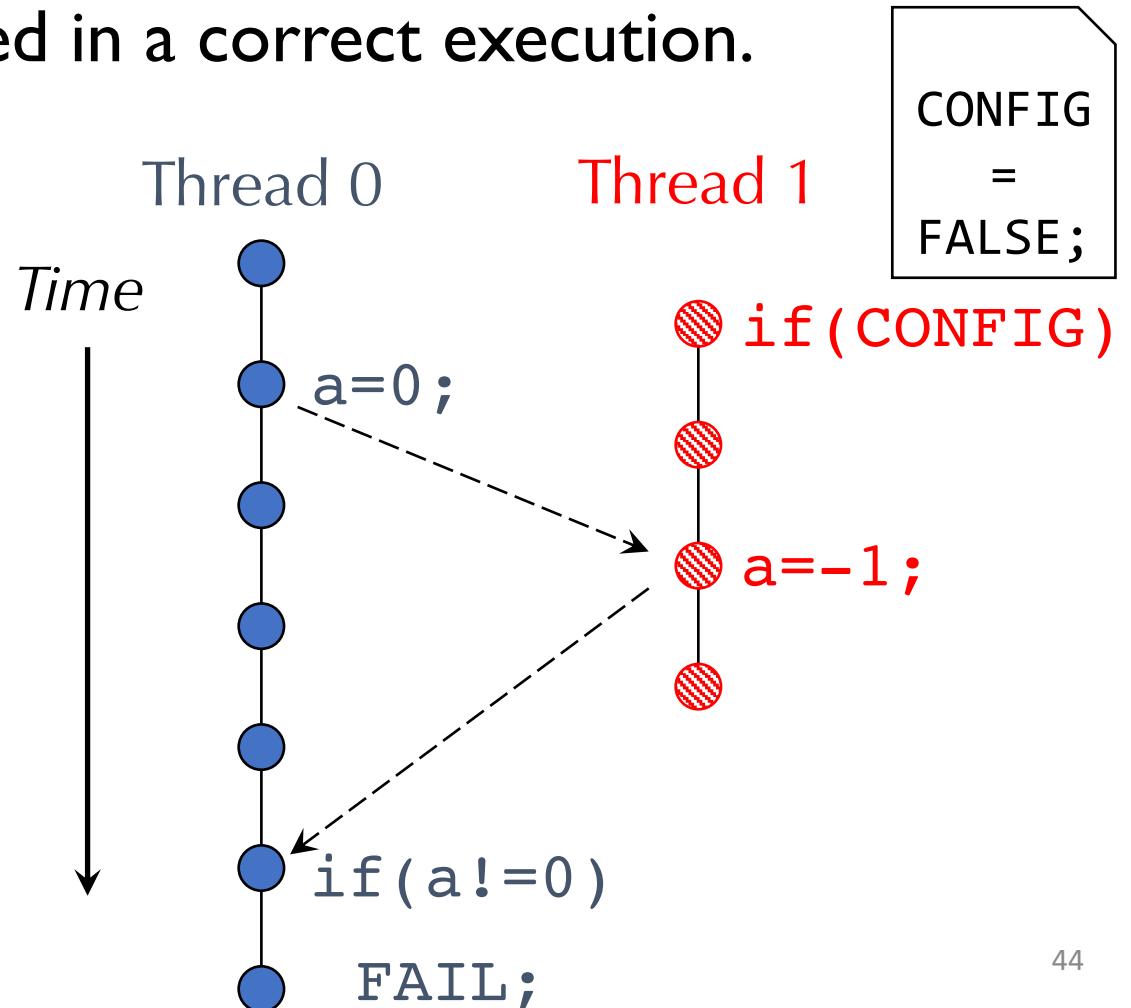
1

- The root cause is an instruction in the failure execution which, if changed, would have resulted in a correct execution.

HDFS-10453 simplified

2

- The root cause is the most basic cause of the failure.



What is a root cause?

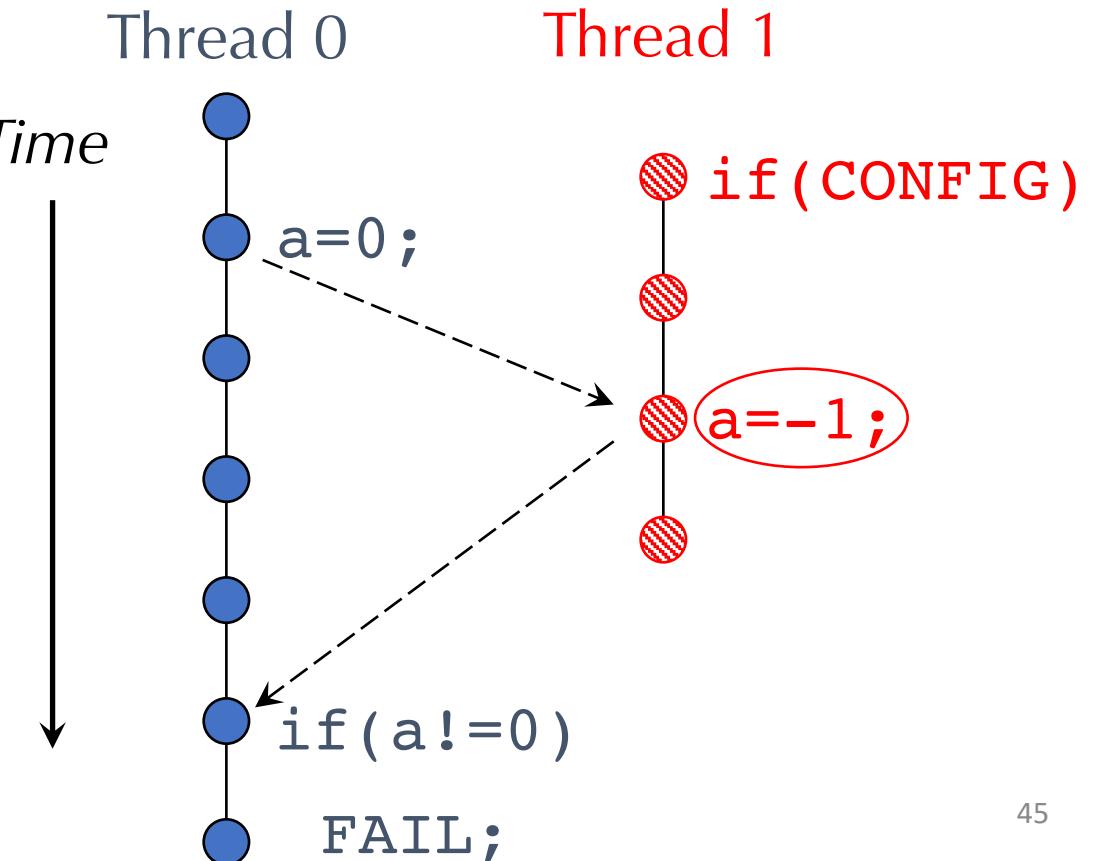
1

- The root cause is an instruction in the failure execution which, if changed, would have resulted in a correct execution.

2

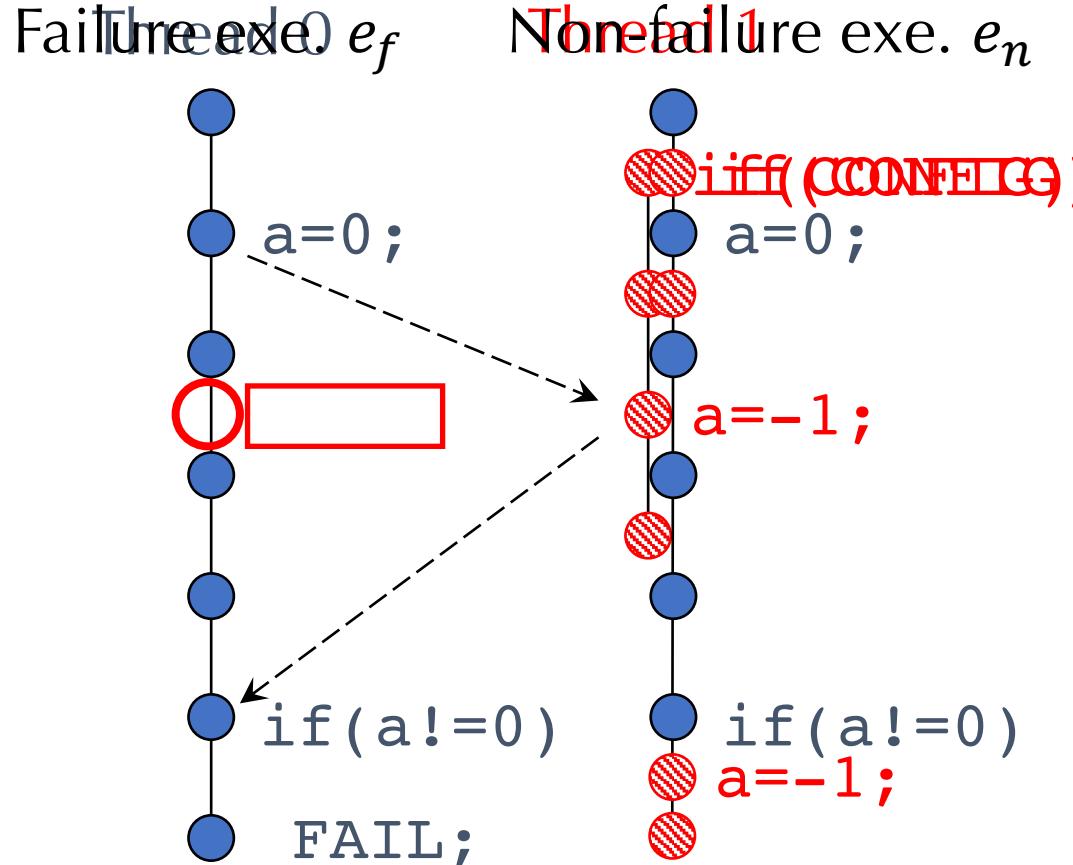
- The root cause is the most basic cause of the failure.

HDFS-10453 simplified

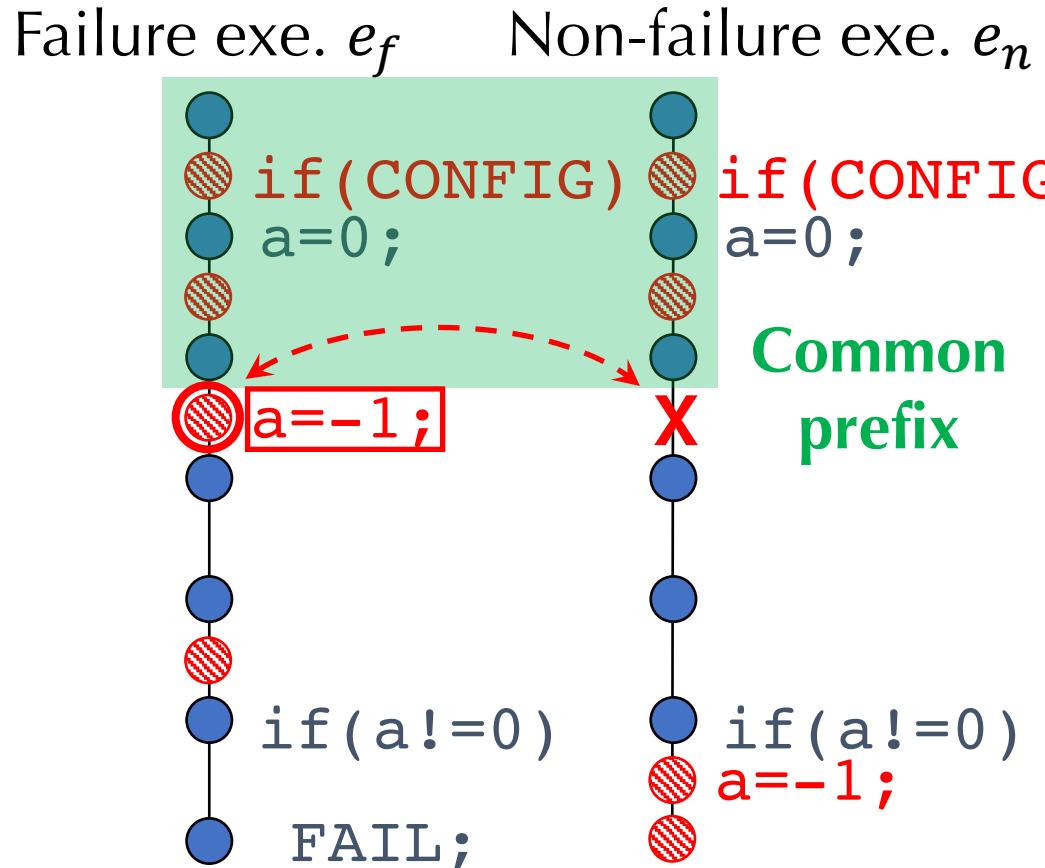


Refined Definition of Root Cause

The root cause is the **first** instruction in the failure execution e_f that once executed would **inevitably** result in the failure

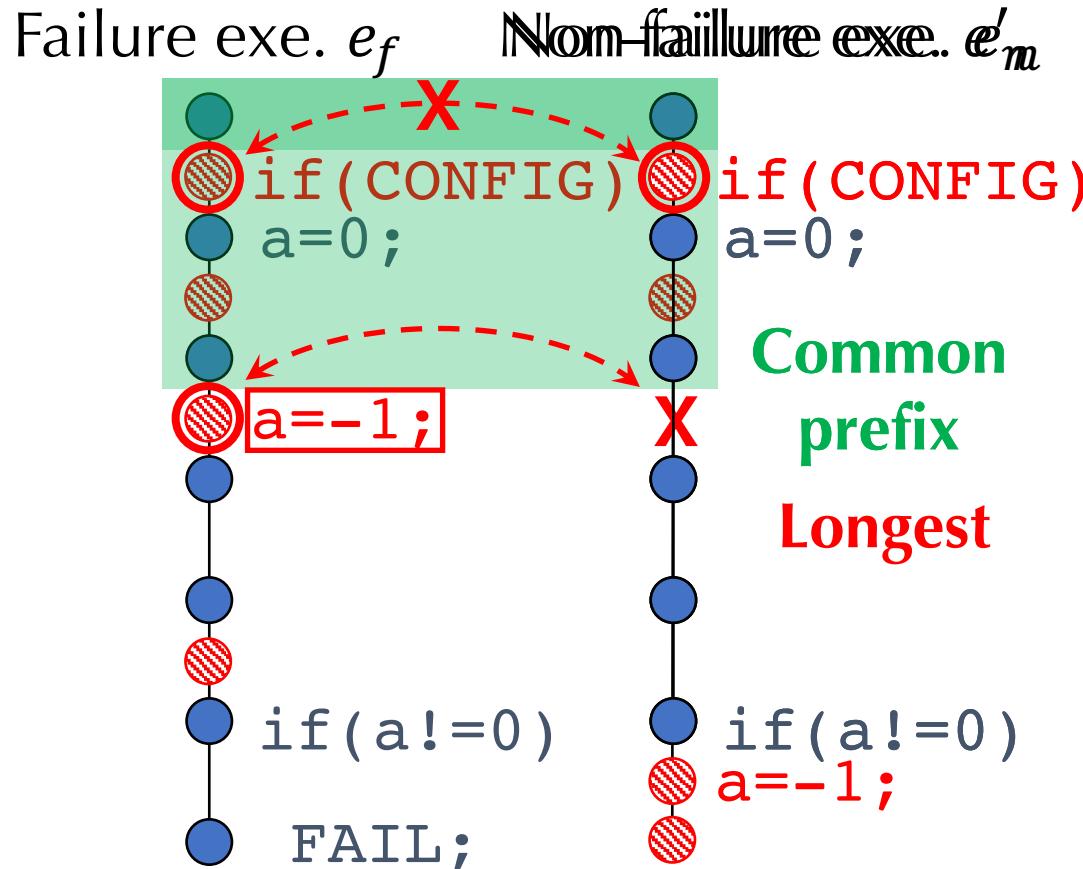


Inflection Point



Their inflection point is the first instruction in e_f that differs from the instruction at the same point in e_n .

Inflection Point Hypothesis



Their inflection point is the first instruction in e_f that differs from the instruction at the same point in e_n .

Σ_n : the set of all valid non-failure executions.

The root cause is located at the inflection point between e_f and $e_n \in \Sigma_n$ that has the **longest common prefix** with e_f .

Inflection Point Hypothesis – highly scalable

- A non-failure execution with minimal execution trace differences
 - Search space: 2^n
- A non-failure execution with the longest common prefix
 - Search space: n

```
void func(...) {  
    if (a1 > 0) {...} T/F } ]  
    ...  
    if (an > 0) T/F ]  
    Log.error("ERROR");  
}
```

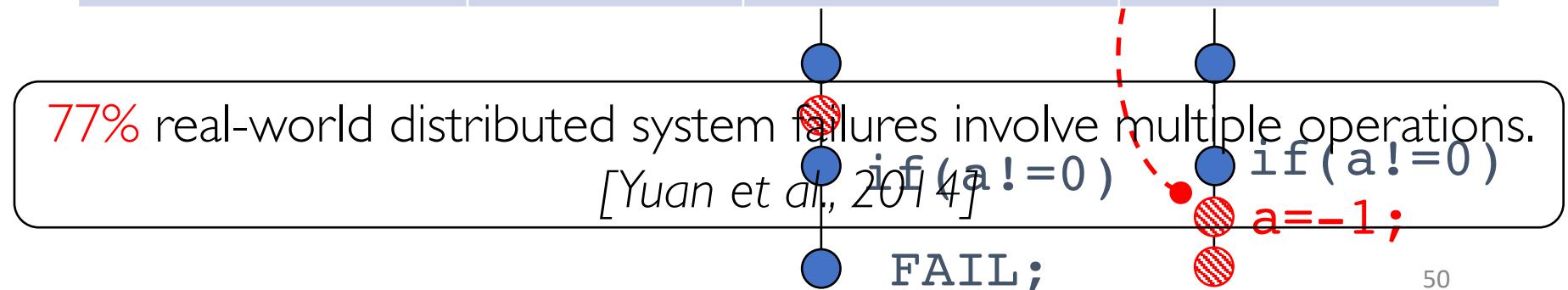
	Failure exe. e_f	Non-failure exe. e_n	Skipped exe.s
search space	T F ...	T ...	F ...
	2^n
	T

Kairux – Automated root cause localization

Construct the non-failure execution e_n that has the longest common prefix.

- Key ideas
 - Adaptive dynamic slicing to obtain partial order
 - Use test cases
 - Stitch test cases to cover more operations
 - Modify existing test cases

System	Class cov.	Function cov.	Statement cov.
Cassandra 3.11.4	88.2%	76.8%	73.6%
HDFS 3.1.2	87.7%	90.4%	90.1%
ZooKeeper 3.4.11	90.6%	90.3%	88.4%



Evaluation of Kairux

- Evaluated Kairux on 10 cases from JVM distributed systems
 - HDFS, HBase, ZooKeeper
 - One case with noisy operations generated from manual reproduction
- Successfully finds inflection points for 7
 - 3 unsuccessful cases:
 - root cause locates in an operation not covered by test cases
- Reduces the # of instructions to diagnose

	# instr. in dynamic slice	# instr. in longest prefix	# tests combined
Avg.	309	165	1.43

Kairux

- Inflection Point Hypothesis: precise, scalable definition of root cause

The root cause is located at the inflection point between e_f and $e_n \in \Sigma_n$ that has the longest common prefix with e_f .

- Kairux
 - transforms the inflection point hypothesis to a practical tool
 - effective in locating root cause in real distributed system failures

Outline

- Introduction
- Kairux [SOSP'19]: automates root cause localization
- Pensieve [SOSP'17]: automates failure reproduction
- Bigger picture

Most Internet services live on the cloud

**Google cloud is down, affecting
Amazon AWS S3 outage is breaking**

**Apple's iCloud recovers after a
four-hour outage**

zdnet.com

September 17, 2018

**Microsoft provides preliminary
report on its September 4 cloud
outage**

Ken Yeung / The Next Web:

**Dropbox says its service is back up and running
outage (Updated)** — Update 2 - 12 January: Dropbox now

npr.org

March 1, 2017

**Amazon Cloud Outage Disrupts
Traffic For Websites, Apps**

Karissa Bell / Mashable:

Adobe Creative Cloud Has Been Down for Almost

Creative Cloud is in the midst of a massive outage that has lasted for nearly 24 hours, impacting affected users throughout the United States, Europe and Asia. — Subscribers to the Creative Cloud plan, which includes Photoshop ...

**Cloudflare blames 'bad software'
deployment for today's outage**

Office 365, Azure users are locked

**VMware Joins Cloud Outage Party With Cloud
Foundry Blackout**

Yahoo Mail Takes Big Hit In Cloud Outage

**Another Cloud Outage Strikes Microsoft BPOS,
Exchange Online**

**Google Docs Goes Dark In Evening Cloud
Outage**

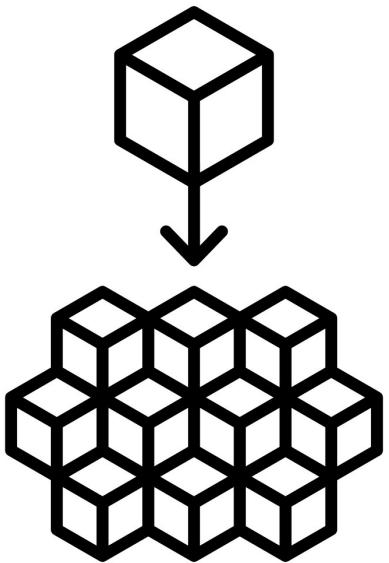
**New Amazon Cloud Outage Takes Down Netflix,
Foursquare**

**Amazon EC2 Goes Dark In Morning Cloud
Outage**

Tom Warren / The Verge:

**YouTube, Snapchat, Gmail, Discord, and many other
services are down throughout US during a major Google Cloud service outage** — YouTube, Snapchat, Gmail, Netflix, and a number of other web services suffered from major outages in the US to

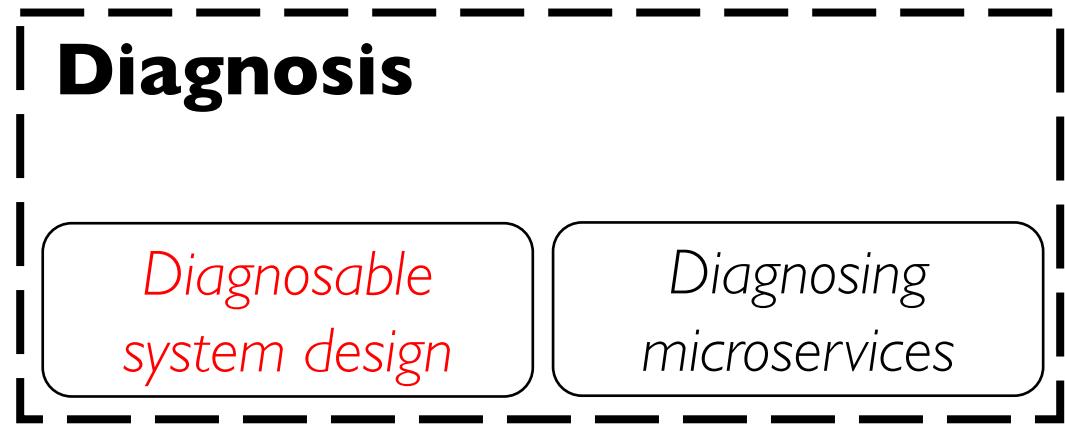
Automated diagnosis on microservices



- My previous work
 - exploits human intuitions and principles
 - **scales** diagnosis techniques to distributed systems
- Challenges in diagnosing microservices – increasing complexity
 - More layers, more interactions between layers & components
 - Dynamically created & destroyed instances
 - Heterogeneous language, hardware
- No existing tools work!



Diagnosable system design



- Challenge: increasing complexity of software systems
 - Hard to understand & diagnose
- Towards diagnosable system design
 - What is the critical information needed for diagnosis?

Backup slides