NTT

Innovative R&D by NTT

Tackling non-determinism in Hadoop

Testing and debugging distributed systems with Earthquake

GitHub : https://github.com/osrg/earthquake/
Twitter: @EarthquakeDMCK

Akihiro Suda <suda.akihiro@lab.ntt.co.jp>
NTT Software Innovation Center

What I will talk about

- What are distributed systems? (e.g., Hadoop)
  Why is it difficult to test them?

- Bugs we found/reproduced with *Earthquake*
  https://github.com/osrg/earthquake/

- How *Earthquake* controls non-determinism

- Lessons we learned

WHAT ARE DISTRIBUTED SYSTEMS?
WHY IS IT DIFFICULT TO TEST THEM?

Big Data Analysis
→Improve business strategy

hadoop Spark

Machine Learning
→Recommend attractive contents to users

mahout Apache SystemML

DISTRIBUTED SYSTEMS, DISTRIBUTED SYSTEMS EVERYWHERE

NoSQL & NewSQL
→Highly available and reliable service

APACHE HBASE riak

Clustered Containers
→Scalable DevOps

kubernetes by Google MESOS

And there are many more..

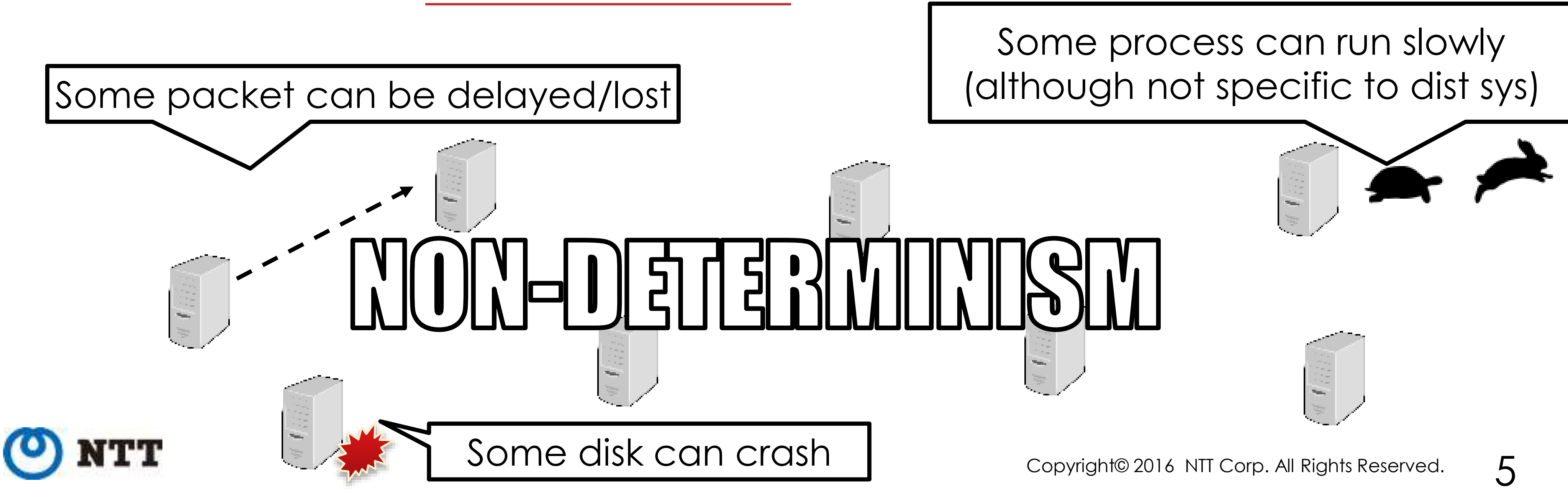
Clip arts are from openclipart.org

Why is it difficult to test distributed systems?

Scalability is the charm of distributed systems

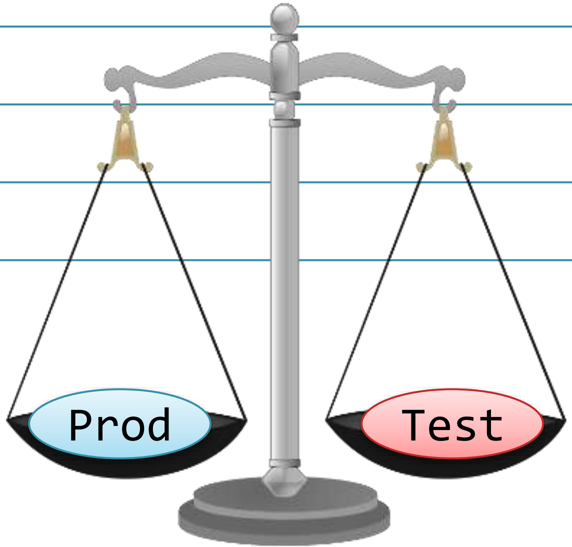
But... it forces the system to be composed of many machines and many software..

→Something in the system *definitely* goes awry!
We need to test whether the system is tolerant of awry things, but it's difficult to non-deternimism


Some packet can be delayed/lost
Some process can run slowly
(although not specific to dist sys)
NON-DETERMINISM
Some disk can crash

Good News: distributed systems are well tested!

Software	Production code (LOC)	Test code (LOC)
MapReduce	95K	87K
YARN	178K	121K
HDFS	152K	150K
ZooKeeper	33K	27K
Spark	167K	128K
Flume	46K	34K
Cassandra	168K	78K
HBase	571K	222K
Kubernetes	309K	153K
etcd	34K	27K



But..

Data are measured at 14/01/2016, using CLOC

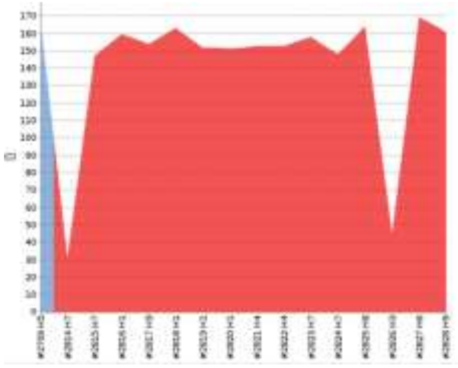
Bad News: they are still buggy

- over 3 bug reports per day, on average
- 50% of bugs take several months to get resolved
- 26% of bugs are "flaky", i.e., hard to reproduce due to non-deternimism

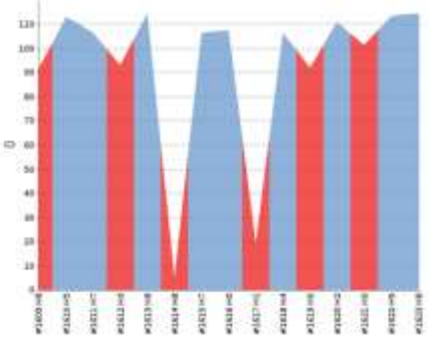
What Bugs Live in the Cloud? A Study of 3000+ Issues in Cloud Systems [Gunawi et.al., SoCC'14]
Simple Testing Can Prevent Most Critical Failures.. [Yuan et. al., OSDI'14]

Data are captured at 14/01/2016

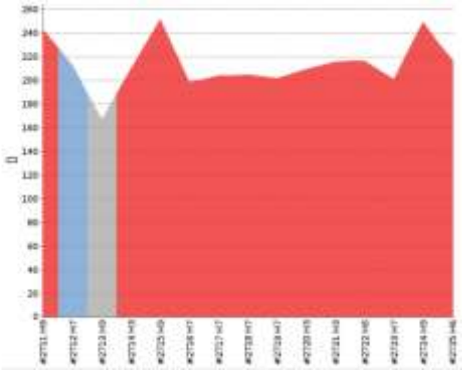
https://builds.apache.org/job/%s-trunk/



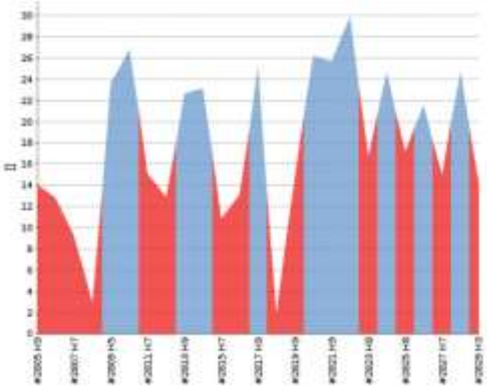
MapReduce



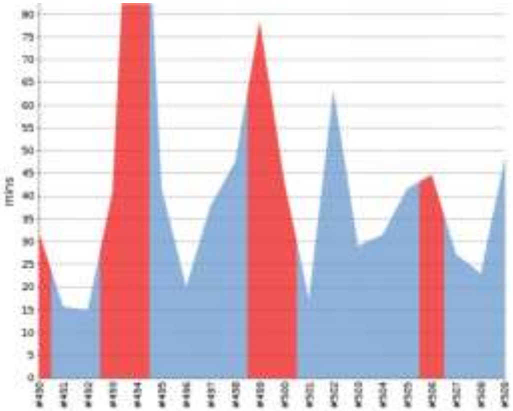
YARN



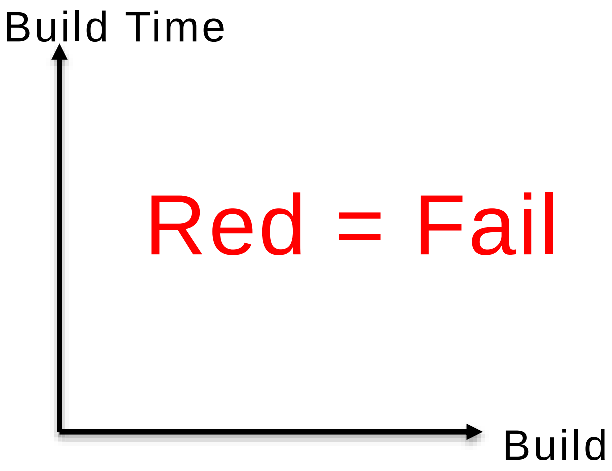
HDFS



ZooKeeper



HBase


Build Time

Red = Fail

Build

Never seen fully successful build, even on my local machine

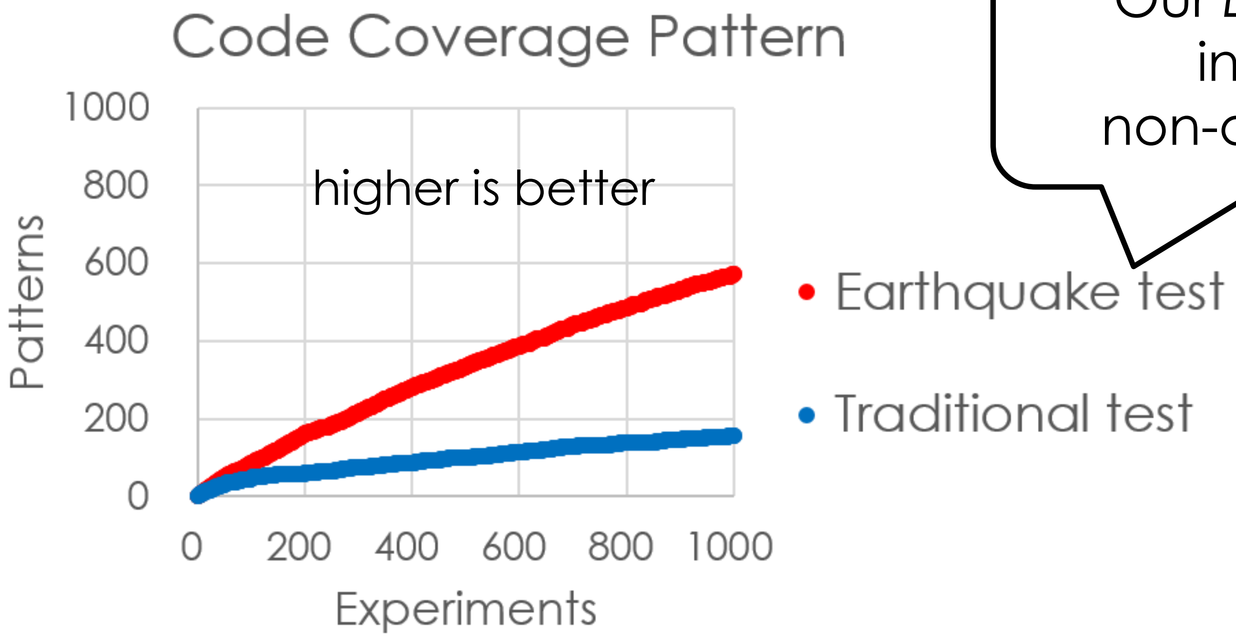
We need to control non-determinism!

- Sometimes we can see test failures on Jenkins

- So.. if we run test codes repeatedly on a PC, we can locally reproduce the bug, right..?

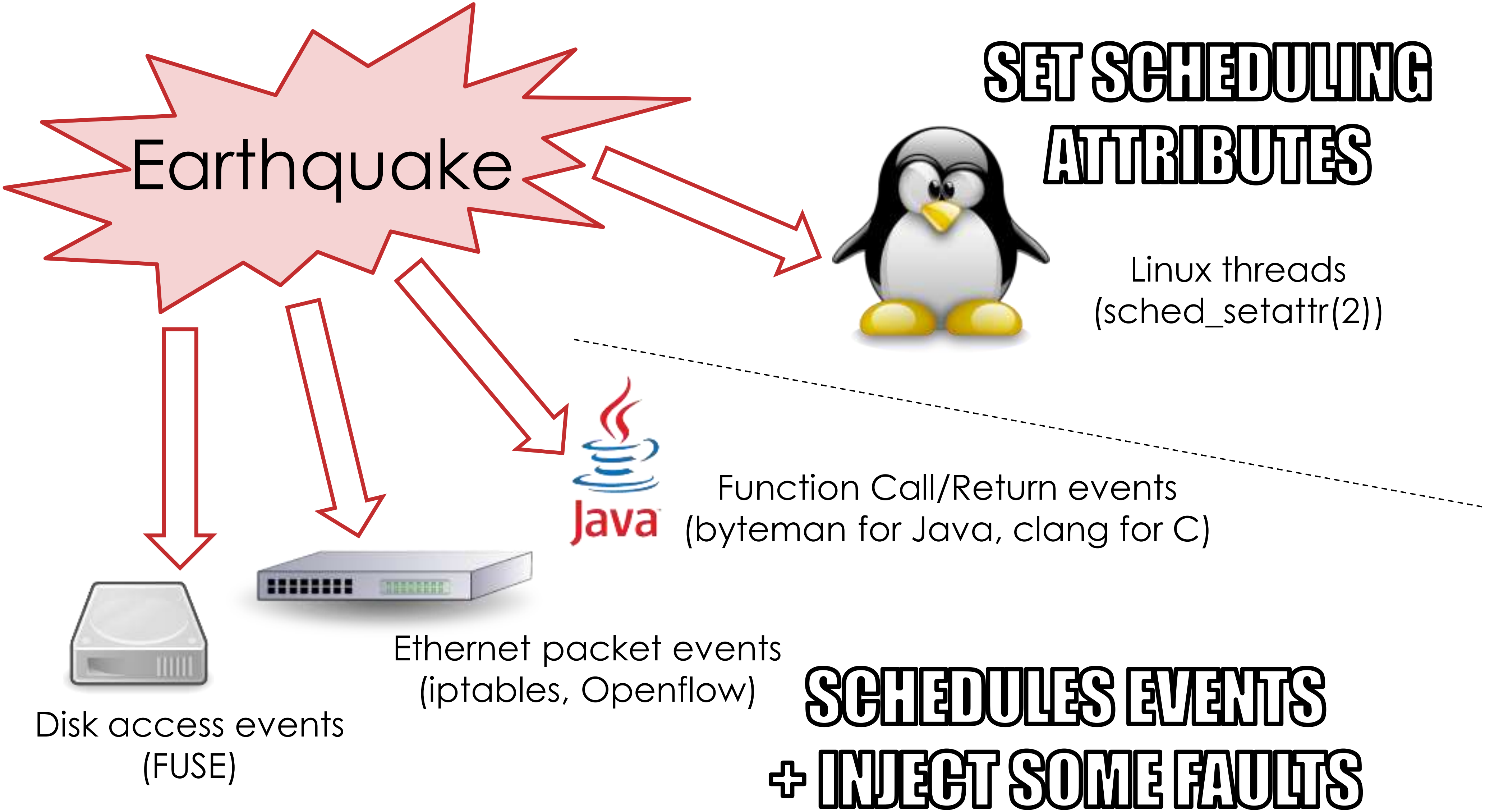
→No! Just repeating tests does *not* make much sense, due to poor non-determinism variation

Our *Earthquake* increases non-determinism



We revisit the plot later

Earthquake: programmable fuzzy scheduler



Earthquake

SET SCHEDULING ATTRIBUTES

Linux threads (sched_setattr(2))

Function Call/Return events (byteman for Java, clang for C)

Ethernet packet events (iptables, Openflow)

Disk access events (FUSE)

SCHEDULES EVENTS + INJECT SOME FAULTS

How to use Earthquake?

For Ethernet scheduling

```
$ go get github.com/osrg/earthquake/earthquake-container
$ sudo earthquake-container run -i -t ubuntu $COMMAND
```

Docker-like CLI
(For non-Dockerized version, please refer to docs)

For Thread scheduling

```
$ git clone https://github.com/AkihiroSuda/microearthquake
$ sudo microearthquake pid $PID
```

Will be unified to `earthquake-container` in February

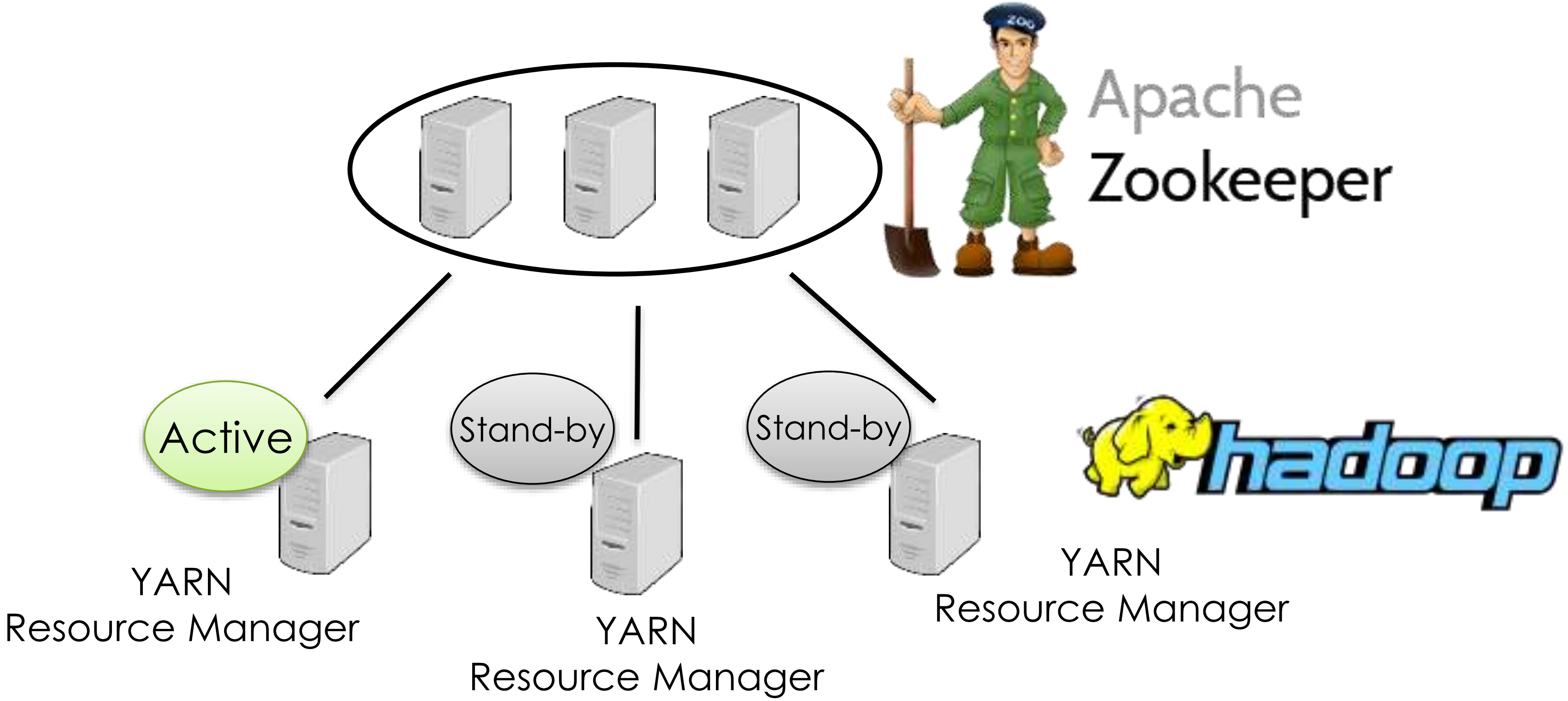
Details of Earthquake
is followed later

BUGS WE FOUND/REPRODUCED WITH EARTHQUAKE

Reproduction codes are included in the github repo:
https://github.com/osrg/earthquake/tree/master/example

ZooKeeper: distributed lock service for Hadoop (and so on)

- **Provides distributed lock for HA in Hadoop**
  - HA of Hadoop relies on ZooKeeper
- **Also used by Spark, Mesos, Docker Swarm, Kafka, ...**
- **Similar softwares: etcd, Consul, Atomix,..**

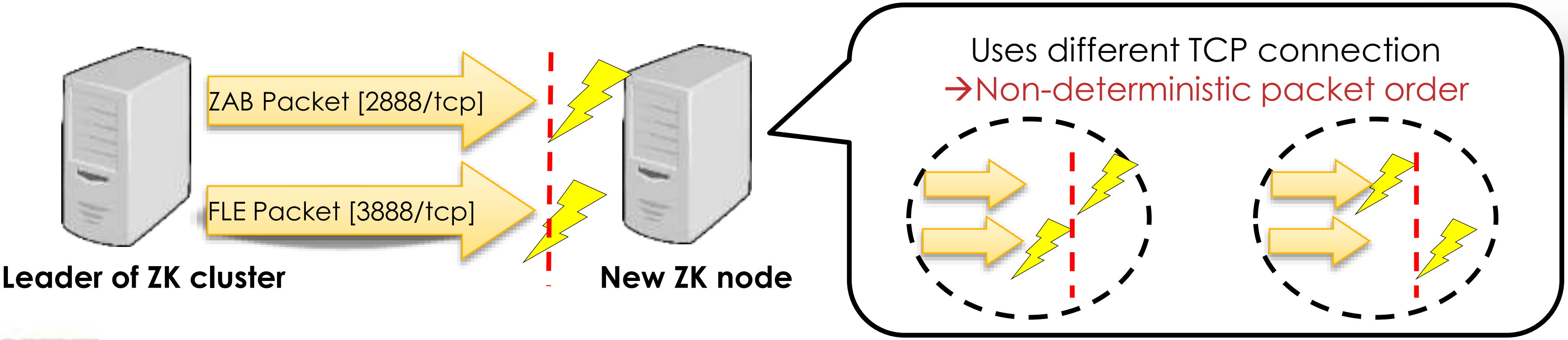


Found https://issues.apache.org/jira/browse/ZOOKEEPER-2212

- **Bug: New node cannot participate to ZK cluster properly**

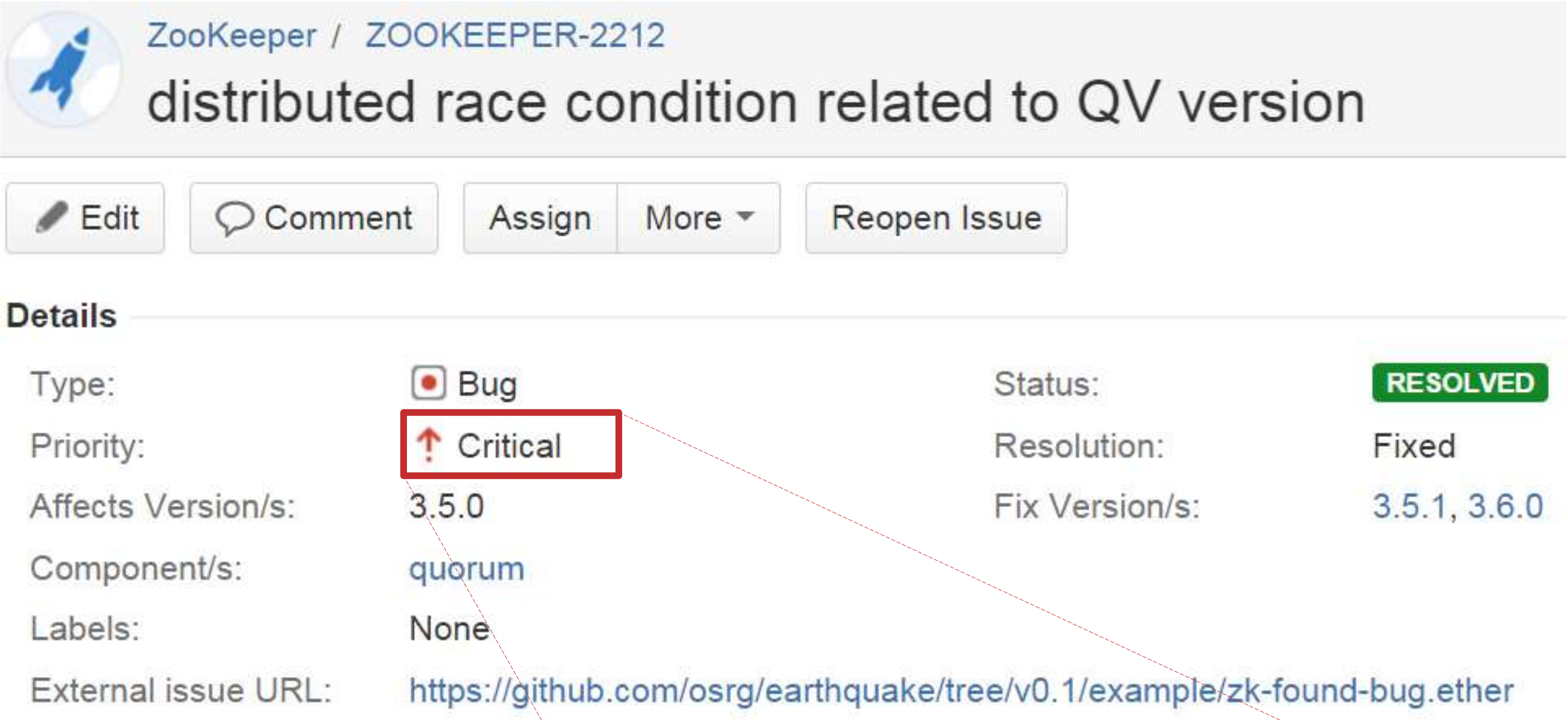
  →New node cannot become a leader of ZK cluster itself

- **Cause: distributed race (ZAB packet vs FLE packet)**
  - ZAB.. broadcast protocol for data
  - FLE.. leader election protocol for ZK cluster itself
  - Code is becoming spaghetti due to many contributions which had not been planned in the first release
    →Interaction between ZAB and FLE is not obvious



Data are captured at 22/01/2016

https://issues.apache.org/jira/browse/ZOOKEEPER-2212



ZooKeeper / ZOOKEEPER-2212

distributed race condition related to QV version

Details

Type: Bug

Priority: ↑ Critical

Affects Version/s: 3.5.0

Component/s: quorum

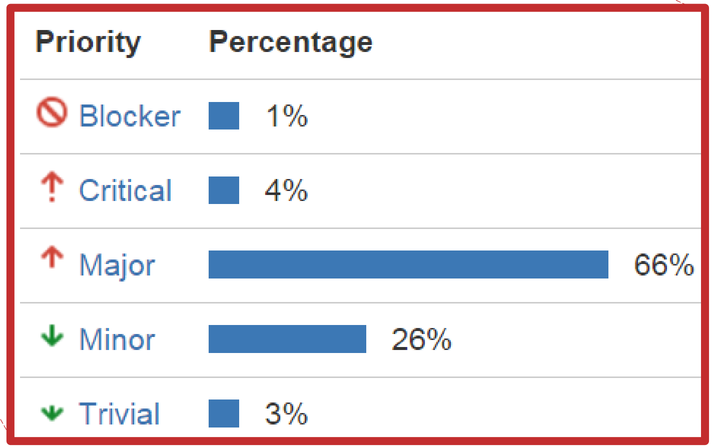
Labels: None

External issue URL: https://github.com/osrg/earthquake/tree/v0.1/example/zk-found-bug.ether

Status: RESOLVED

Resolution: Fixed

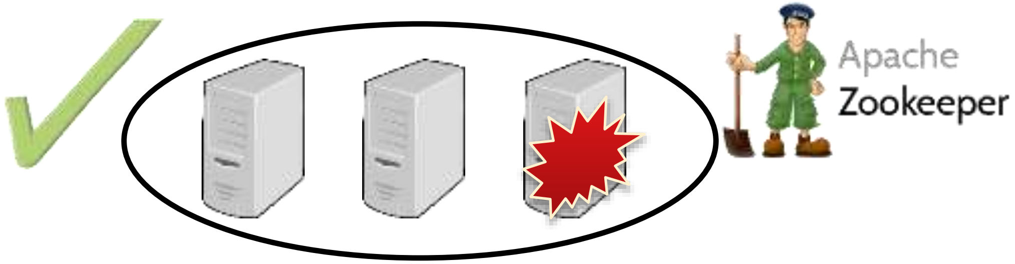
Fix Version/s: 3.5.1, 3.6.0



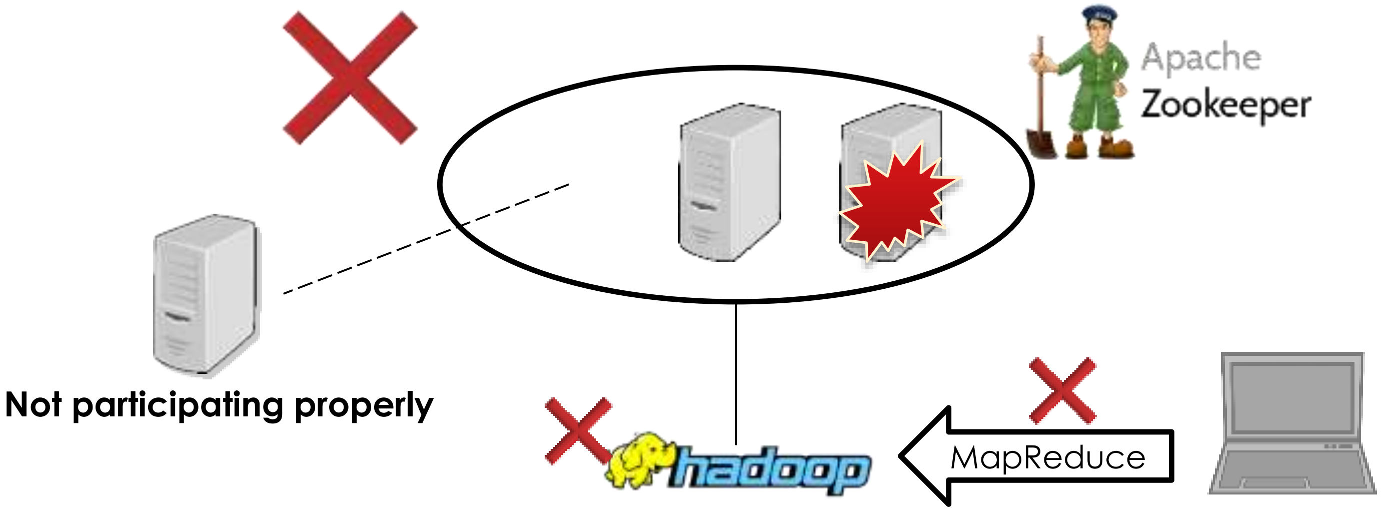
Priority	Percentage
Blocker	1%
↑ Critical	4%
↑ Major	66%
↓ Minor	26%
↓ Trivial	3%

Affect to production

- **Expected: ZK cluster works even when $\lfloor N/2 \rfloor$ nodes crashed**
  (N is an odd number, 3 or 5 in most cases)

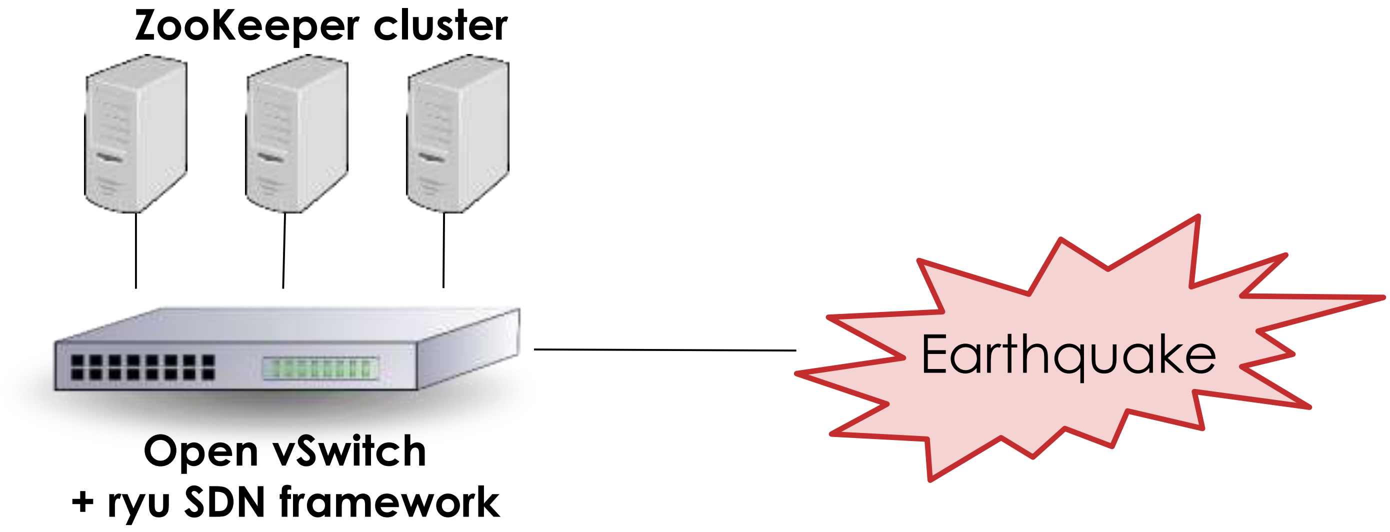


- **Real: it doesn't work!**
  → Example: No MapReduce job can be submitted to Hadoop



How hard is it to reproduce?

- **We permuted some specific Ethernet packets in random order using Earthquake**
  - Unlike Linux netem (or FreeBSD dummynet), Earthquake provides much more programmable interface



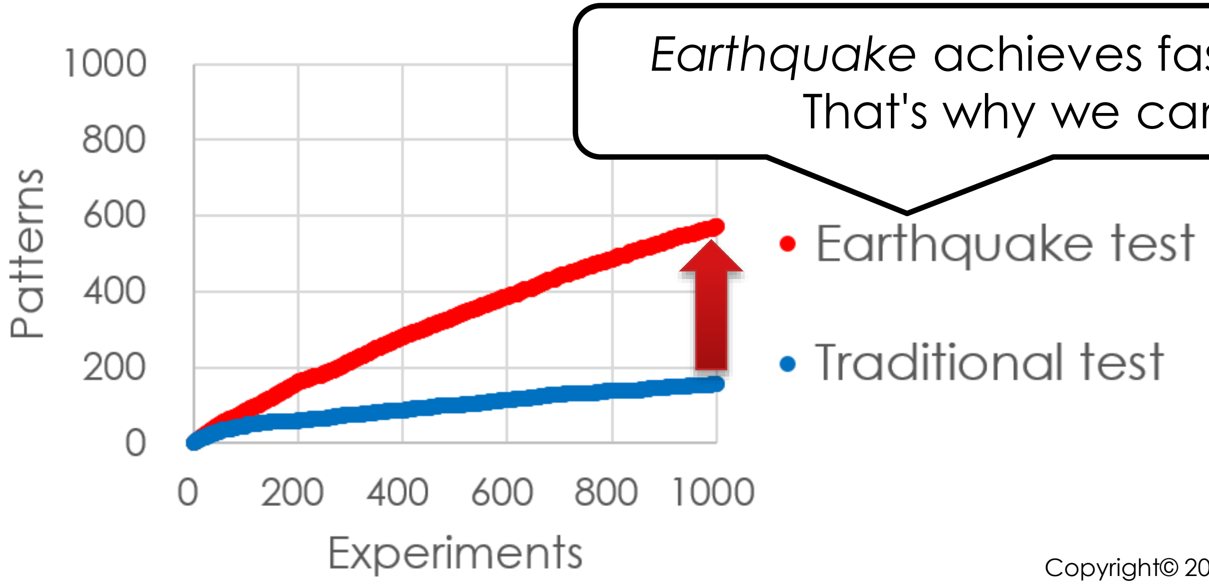
- **Reproducibility: 0.0% → 21.8% (tested 1,000 times)**
  - We could *not* reproduce the bug even after 5,000 times traditional testing (60 hours!)

Distributed Execution Pattern

We define *the distributed execution pattern* based on code coverage:

$$\boldsymbol{P} = \begin{pmatrix} \boldsymbol{p}_{\mathbf{1},\mathbf{1}} & \cdots & \boldsymbol{p}_{\mathbf{1},\boldsymbol{N}} \\ \vdots & \ddots & \vdots \\ \boldsymbol{p}_{\boldsymbol{L},\mathbf{1}} & \cdots & \boldsymbol{p}_{\boldsymbol{L},\boldsymbol{N}} \end{pmatrix}$$

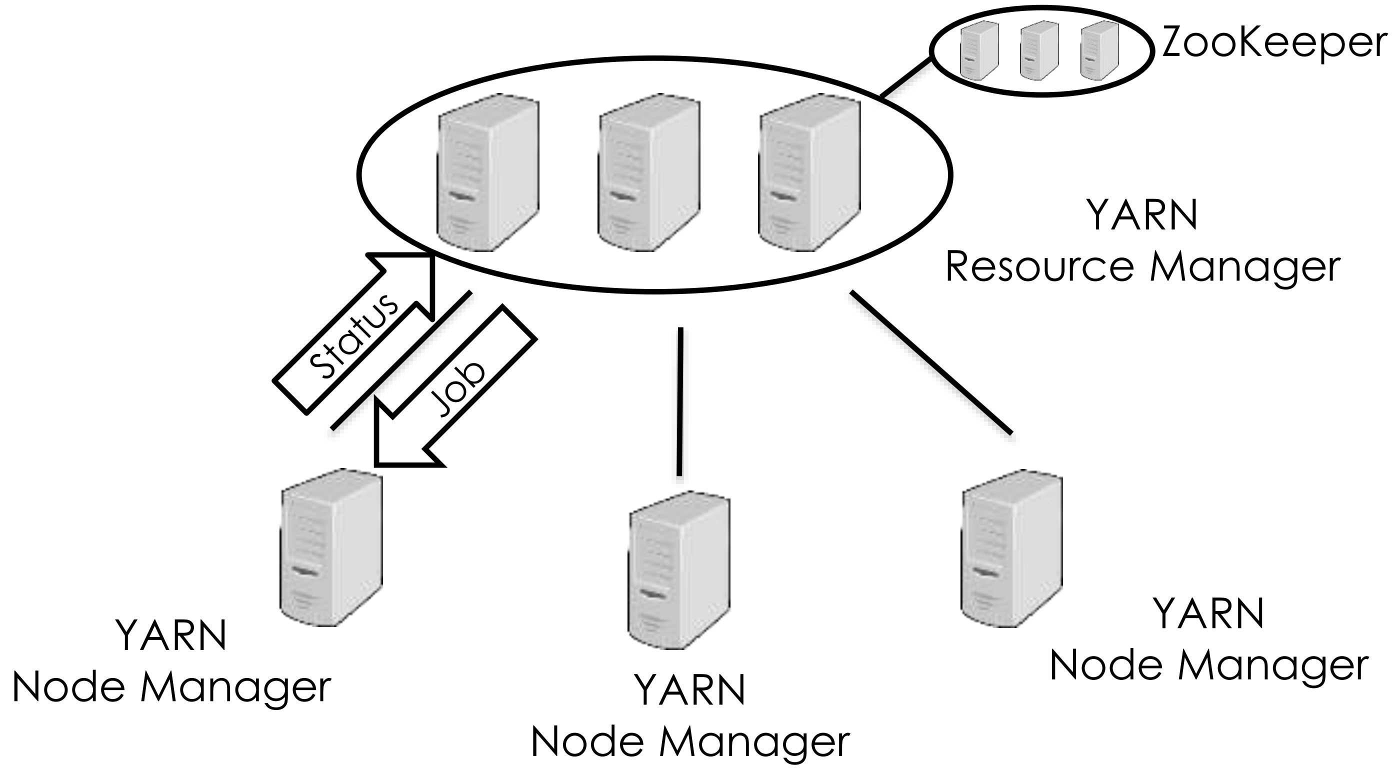
- L: Number of line of codes
- N: Number of nodes
- $p_{i,j}$: 1 if the node j covers the branch in line i, otherwise 0
- We used JaCoCo: Java Code Coverage Library



Earthquake achieves faster pattern growth
That's why we can hit the bug

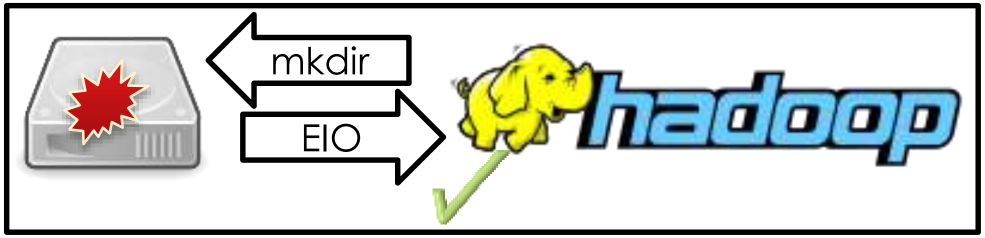
YARN: the kernel of Hadoop

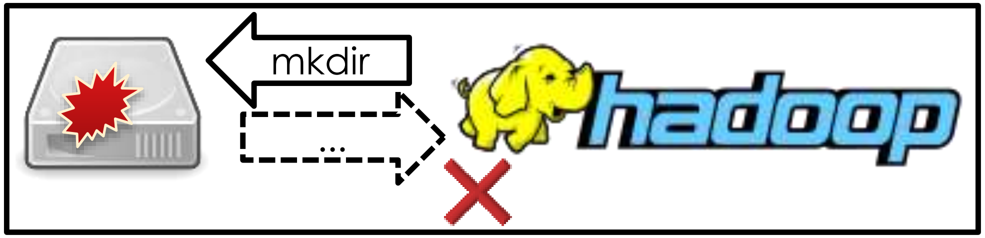
- YARN assigns MapReduce jobs to computation nodes
- Computation nodes reports their health status for fault-tolerant job scheduling



Found https://issues.apache.org/jira/browse/YARN-4301

- **Bug: YARN cannot detect disk failure cases where mkdir()/rmdir() blocks**





- **We noticed that the bug can occur theoretically when we are reading the code, and actually produced the bug using Earthquake**
  - When we should inject the fault is pre-known;
    so we manually wrote a concrete scenario using Earthquake API
  - Much more realistic than mocking

Found/Reproduced flaky tests (excluding simple "timed out" fail)

R: ZOOKEEPER-2080: CnxManager race

- Not reproduced nor analyzed for past 2 years
- Can affect production, not only in JUnit test

R: YARN-1978: NM TestLogAggregationService race

R: YARN-4168: NM TestLogAggregationService race

F: YARN-4543: NM TestNodeStatusUpdater race

F: YARN-4548: RM TestCapacityScheduler race

R: YARN-4556: RM TestFifoScheduler race

R: etcd #4006: kvstore_test race

R: etcd #4039: kv_test race

and more..

Flaky test doesn't matter, as it doesn't affect production?

It still matters!

For developers..
It's a barrier to promotion of CI

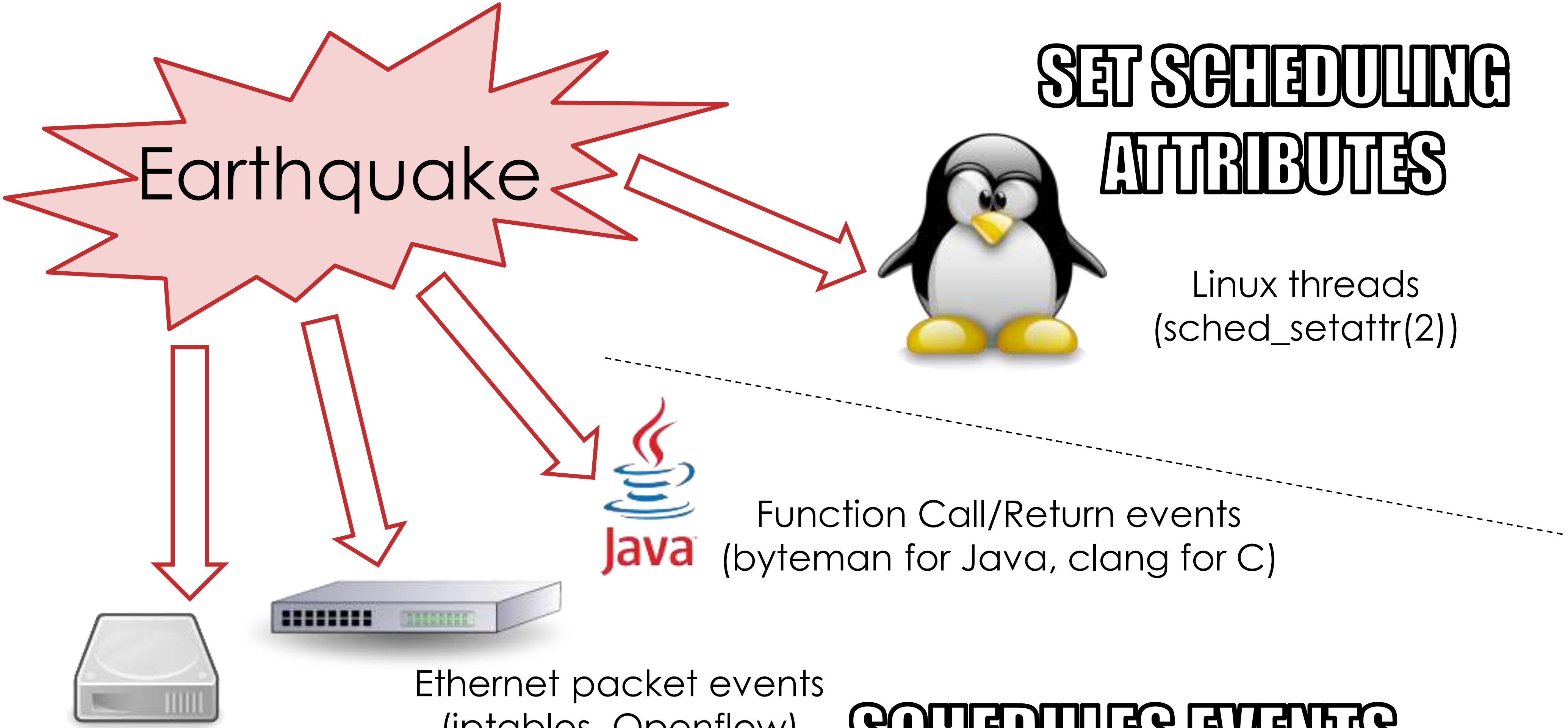
- If many tests are flaky, developers tend to ignore CI failure → overlook real bugs

For users..
It's a barrier to risk assessment for production

- No one can tell flaky tests from real bugs

HOW EARTHQUAKE CONTROLS NON-DETERMINISM

Earthquake: programmable fuzzy scheduler



SET SCHEDULING ATTRIBUTES

Linux threads (sched_setattr(2))

Function Call/Return events (byteman for Java, clang for C)

Ethernet packet events (iptables, Openflow)

Disk access events (FUSE)

SCHEDULES EVENTS + INJECT SOME FAULTS

Three Major Principles of Earthquake

- Non-invasiveness

- Incremental Adoptability

- Language Independence

Principle 1 of 3: Non-invasiveness

- **It's possible to insert extra hook codes for controlling non-determinism**
  - Great academic works: SAMC[OSDI'14], DEMi[NSDI'16],..
- **But we avoid such "invasive" modifications:**
  - We need to test multiple versions
    - For evaluating which version is suitable for production
    - For bisection testing (`git bisect`)
    - →We don't like to rebase the modification patch!
  - We don't like to see false-positives!
    - modification itself can be a false bug
    - debugging false bugs is vain

Principle 2 of 3: Incremental Adoptability

- **Real implementation of distributed systems are very complicated and difficult to understand**

Paper	Pages
Google's MapReduce [OSDI '04]	13
Chubby (==ZooKeeper, etcd) [OSDI'06]	16
BigTable (==HBase) [OSDI'06]	16
Borg (==Kubernetes) [Eurosys'15]	18
Spark [NSDI'12]	14

Software	Production code (LOC)
MapReduce	95K
YARN	178K
HDFS	152K
ZooKeeper	33K
Spark	167K
Flume	46K
Cassandra	168K
HBase	571K
Kubernetes	309K
etcd	34K

- **So we need to start testing without complex knowledge**

- **But after we got the details, we want to incrementally utilize the new knowledge for improving test planning**

Principle 3 of 3: Language Independence

- **Many distributed systems are written in Java**

- **But the world won't be unified under Java**
  - Scala.. Spark, Kafka
  - Go.. etcd, Consul
  - Erlang.. Riak, RabbitMQ
  - Clojure.. Storm, ..
  - C++.. Kudu,
  - ..

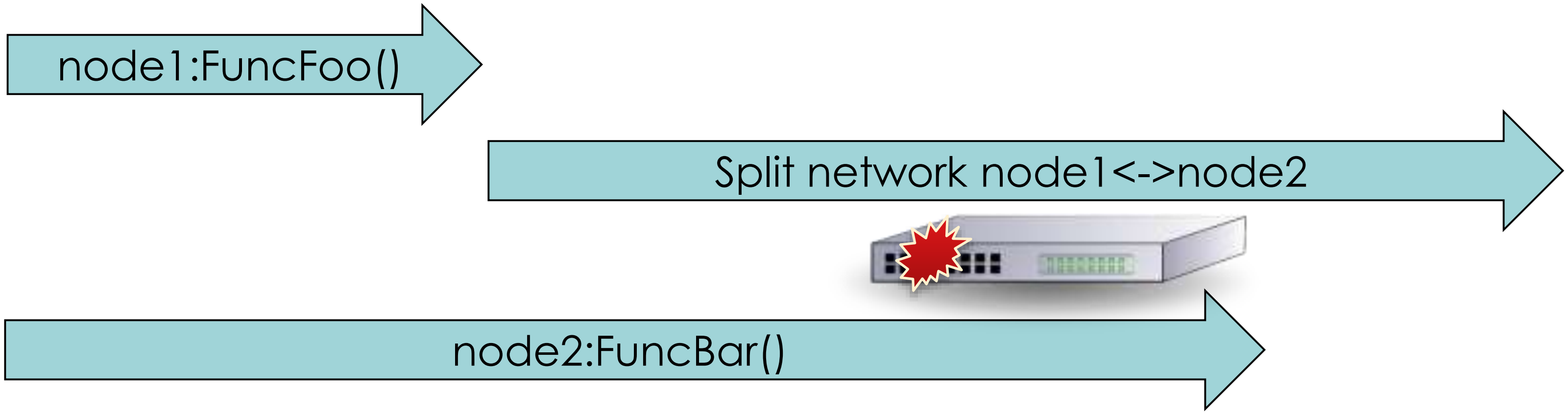
- **We avoid language-dependent technique so that we can test any software written in any language**
  - Earthquake has some Java/C dependent components, but they are just extensions, not mandatory.

How Earthquake schedules events and faults

- **Random is the default behavior**
  - No state transition model is required

Events:
- PacketEvent (iptables, Openflow)
- FilesystemEvent (FUSE)
- JavaFunctionEvent(byteman)
- SyslogEvent ..

- **If you are suspicious of some specific scenarios, you can program the scenarios!**
  - e.g., If FuncFoo() and FuncBar() happen concurrently (in XX msecs window) ..
    - You can ensure that FuncFoo() returns before FuncBar() returns
    - You can inject a fault between FuncFoo() and FuncBar()



Earthquake API (Go)

```
type ExplorePolicy interface {
    QueueNextEvent(Event)
    GetNextActionChan() chan Action
}
```

Events:

- PacketEvent (iptables, Openflow)
- FilesystemEvent (FUSE)
- JavaFunctionEvent(byteman)
- SyslogEvent ..

Actions:

- (Default action)
- PacketFaultAction
- FilesystemFaultAction
- ShellAction ..

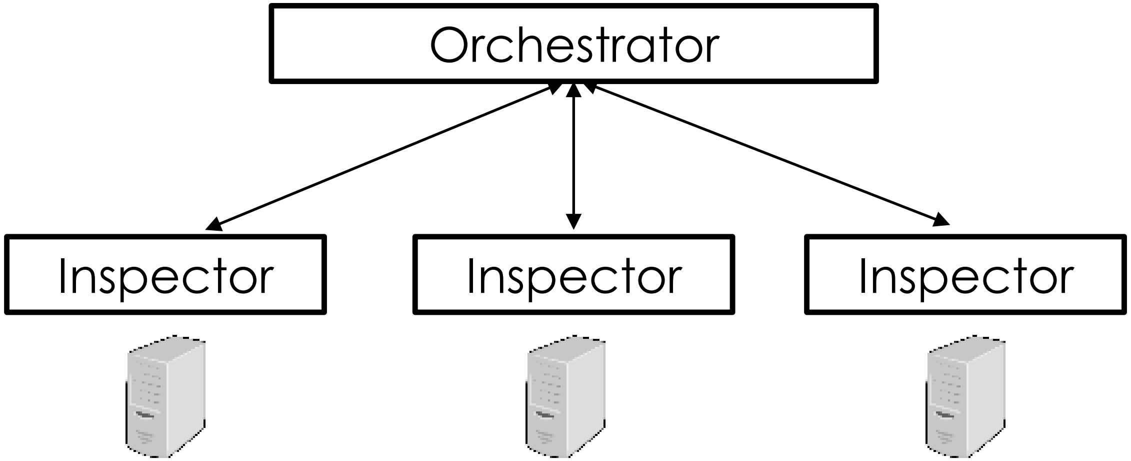
```
func (p *MyPolicy) QueueNextEvent(event Event) {
    action := event.DefaultAction()
    p.timeBoundedQ.Enqueue(action,
      10 * Millisecond, 30 * Millisecond)
}

func (p *MyPolicy) GetNextActionChan() chan Action {.. }

func NewMyPolicy() ExplorePolicy { return &MyPolicy {..} }

func main() {
    RegisterPolicy("mypolicy", NewMyPolicy)
    os.Exit(CLIMain(os.Args))
}
```

Fired in [10ms, 30ms]

Go is poor at DLL..
So user-written policy plugin is statically linked
(as in Docker Machine drivers)

Earthquake API

- **RPC mode**
  - Best for complex scenarios
  - Doesn't scale so much, but it doesn't matter for tests

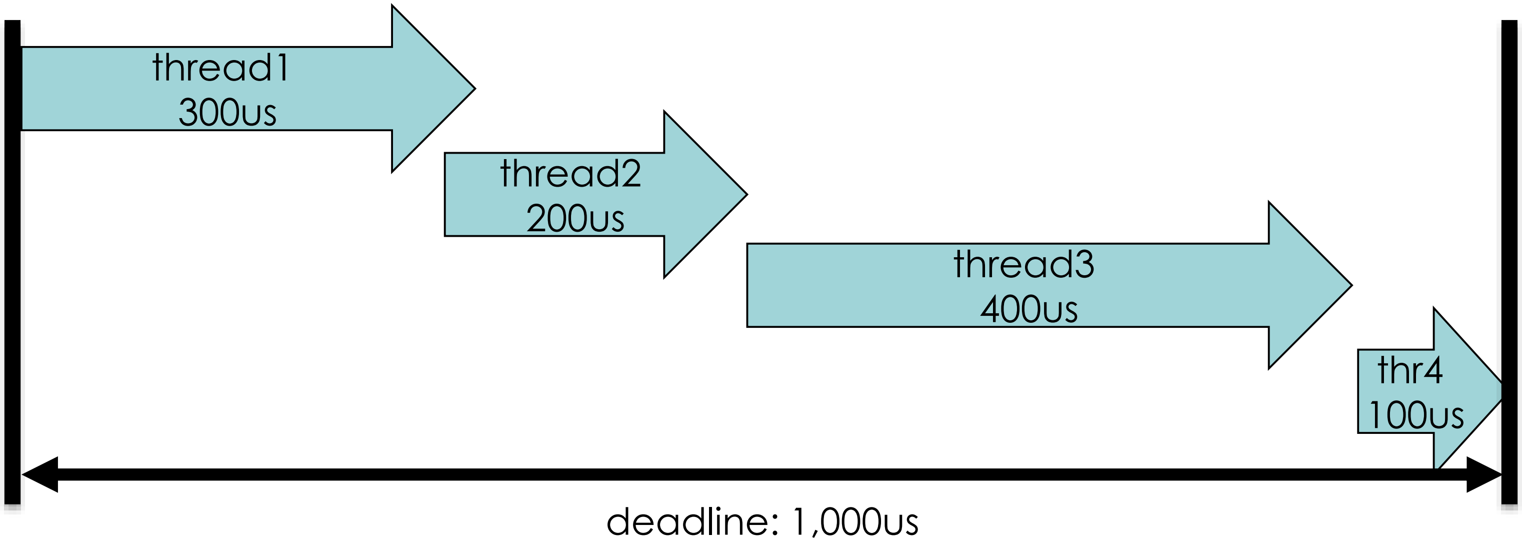


- **Auto-pilot, single-binary mode**
  - Best for the default randomized scenario

How Earthquake sets scheduling attributes?

- **Linux 3.14 introduced SCHED_DEADLINE scheduler**
  - Earthquake is not related to performance testing; but SCHED_DEADLINE is much more configurable than renice(8) with default CFQ scheduler; so we use it

- **Earthquake changes *runtime* of threads, using SCHED_DEADLINE + sched_setattr(2)**
  - Finds runtime set that satisfies the deadline constraint using Dirichlet-distributed random values ($\sum r_i = 1000$)

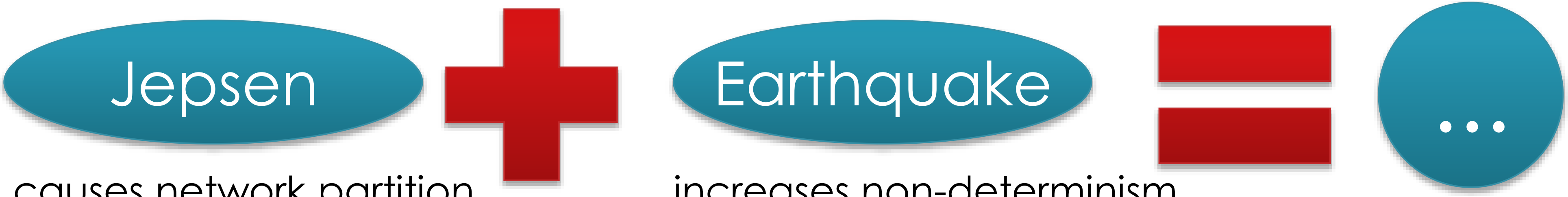


Similar great tool: Jepsen

- **Jepsen**
  - specialized in network partition
  - specialized in testing linearizability (strong consistency)
  - Famous for "Call Me Maybe" blog: http://jepsen.io/

- **Earthquake is much more generalized**
  - The bugs we found/reproduced are basically beyond the scope of Jepsen

- **Earthquake can be also combined with Jepsen!**
  It will be our next work..


Jepsen + Earthquake = …

causes network partition
tests linearizablity

increases non-determinism
(especially of thread scheduling)

LESSONS WE LEARNED

Performance should be split from liveness

"Liveness": something good *eventually* happens

- c.f. "safety": nothing bad happens (typical assertions are safety)

Problem: Flaky test expects something good happens in a certain timeout, which depends on the performance of the machine

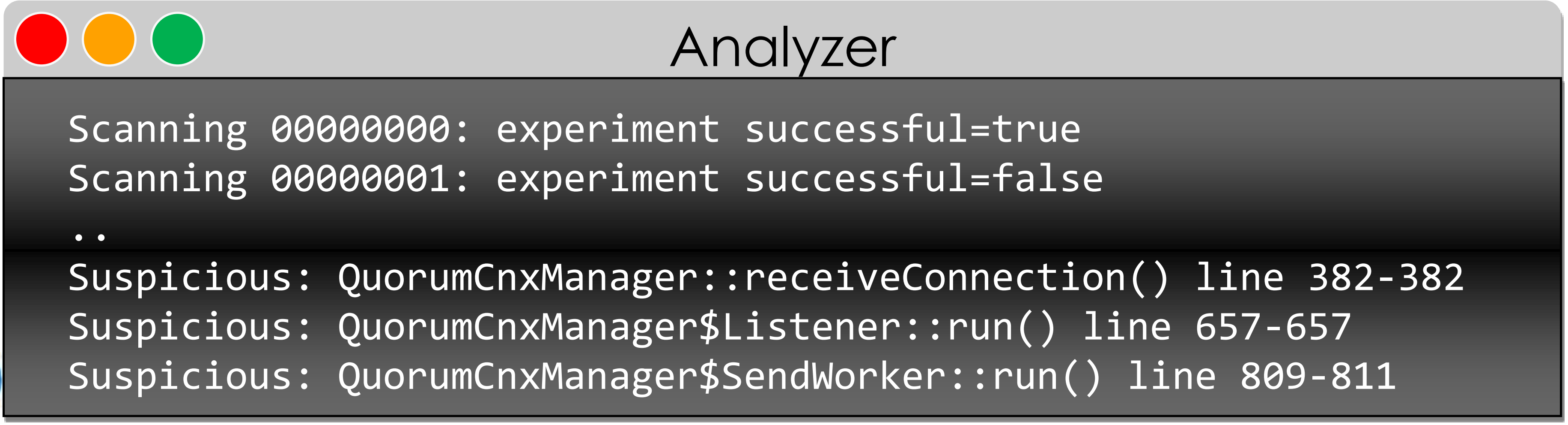
```
invokeAsyncProcess();
sleep(certainTimeout); // some tests lack even this sleep for async proc
assertTrue(checkSomethingGoodHasHappened());
```

Solution: make sure any test succeeds on slow machines
(of course performance testing is important, but it should be separated)

```
invokeAsyncProcess();
bool b;
for (int i = 0; i < MANY_RETRIES; i++) {
    sleep(certainTimeout);
    b = checkSomethingGoodHasHappened(); // idempotent
    if (b) break;
}
assertTrue(b);
```

LOG.debug() is sometimes useless for debugging

- **Of course it is definitely better than nothing**
- **Problem: But sometimes it's useless for distributed systems, because it's too hard for humans to inspect logs from multiple nodes**
  - Unclear ordering
  - Unclear structure (needs an alternative to log4j!)
  - enormous non-interesting logs
- **Solution: So we made a tool that picks up some branches that only occur in failed experiments for analyzing the root cause of the bug**



```
Scanning 00000000: experiment successful=true
Scanning 00000001: experiment successful=false
..
Suspicious: QuorumCnxManager::receiveConnection() line 382-382
Suspicious: QuorumCnxManager$Listener::run() line 657-657
Suspicious: QuorumCnxManager$SendWorker::run() line 809-811
```

Need for integration test framework

Each of the Hadoop components has plenty of JUnit test codes

But we also need integration tests with multiple components!

- e.g. ZOOKEEPER-2347: critical deadlock which leaded to withdrawal of ZooKeeper release v3.4.7
  - Found (not by us nor with Earthquake) in an integration test with ZooKeeper and HBase

Apache BigTop is an interesting project about this!

- But.. how should we prepare realistic workloads systematically？ It's an open question..

CONCLUSION

Conclusion

- Distributed systems are difficult to test and debug, due to non-determinism
- Earthquake controls non-determinism so that you can find and reproduce such "flaky" bugs
- Please try and give us your feedback: https://github.com/osrg/earthquake/

How to use Earthquake?

For Ethernet scheduling

```
$ go get github.com/osrg/earthquake/earthquake-container
$ sudo earthquake-container run -i -t ubuntu $COMMAND
```

Docker-like CLI
(For non-Dockerized version, please refer to docs)

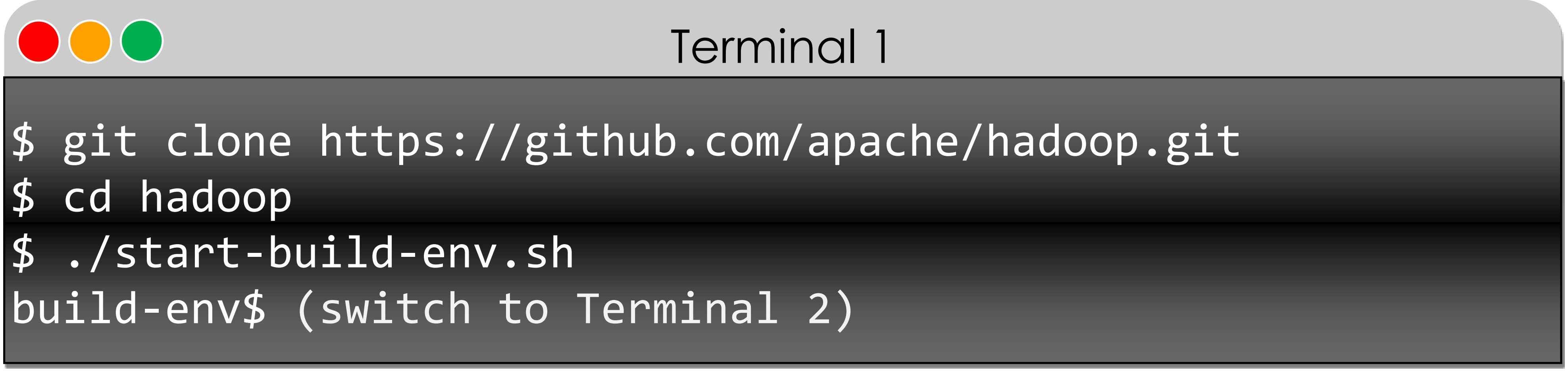
For Thread scheduling

```
$ git clone https://github.com/AkihiroSuda/microearthquake
$ sudo microearthquake pid $PID
```

Will be unified to `earthquake-container` in February

ADDENDUM

Quick start for reproducing flaky bugs in Hadoop [1/3]

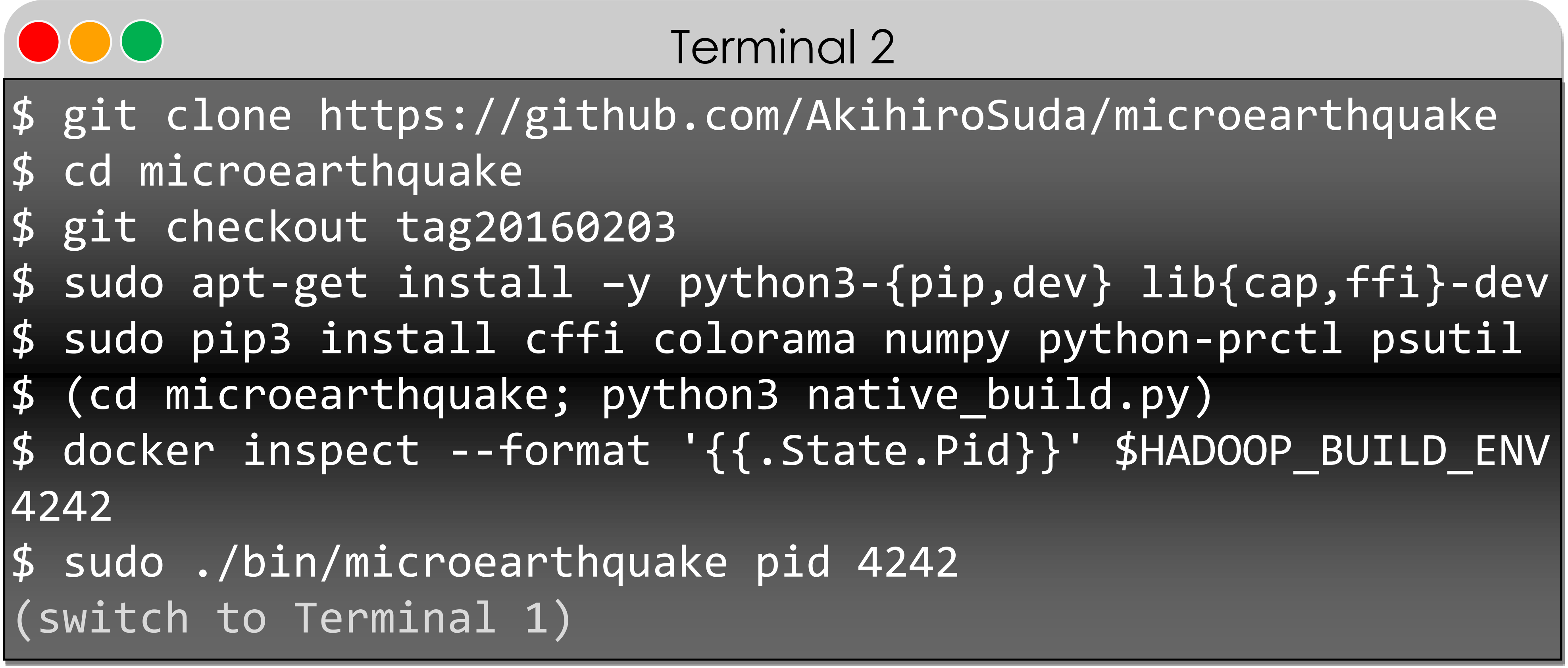


Terminal 1

```
$ git clone https://github.com/apache/hadoop.git
$ cd hadoop
$ ./start-build-env.sh
build-env$ (switch to Terminal 2)
```

- `start-build-env.sh` starts the build environment in a Docker container. Suppose the started container name is $HADOOP_BUILD_ENV.
- If you have a package dependency issue, you may have to run `mvn package -Pdist -DskipTests -Dtar` in the container .

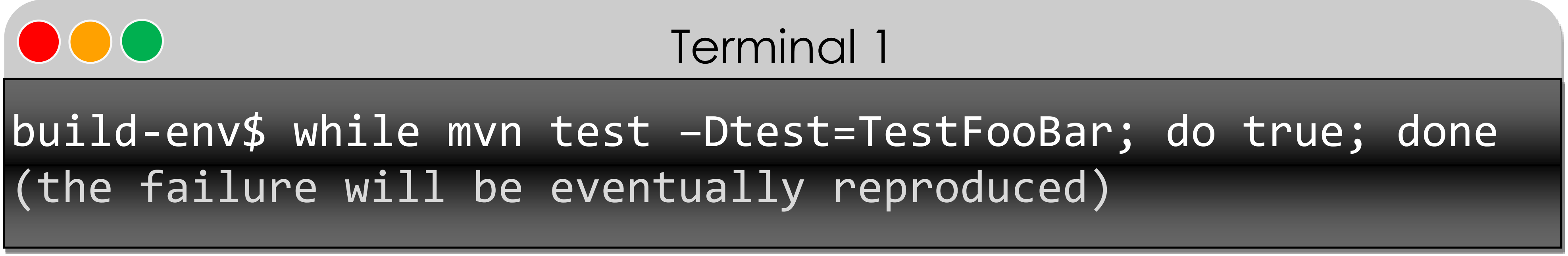
Quick start for reproducing flaky bugs in Hadoop [2/3]



```
$ git clone https://github.com/AkihiroSuda/microearthquake
$ cd microearthquake
$ git checkout tag20160203
$ sudo apt-get install -y python3-{pip,dev} lib{cap,ffi}-dev
$ sudo pip3 install cffi colorama numpy python-prctl psutil
$ (cd microearthquake; python3 native_build.py)
$ docker inspect --format '{{.State.Pid}}' $HADOOP_BUILD_ENV
4242
$ sudo ./bin/microearthquake pid 4242
(switch to Terminal 1)
```

- `microearthauke` fuzzes the thread scheduling for all the threads under the process tree (pid=4242).
- `microearthquake` will be rewritten in Go, and unified to `earthquake-container`. (planned in February)

Quick start for reproducing flaky bugs in Hadoop [3/3]



```
build-env$ while mvn test -Dtest=TestFooBar; do true; done
(the failure will be eventually reproduced)
```

- Suppose TestFooBar.java is failing intermittently on Jenkins.
- You can find failing tests at https://hadoop.apache.org/issue_tracking.html

FAQs

Q. How can I write a test?

A. Basically you don't have to write any new test, as you can use existing xUnit tests.

If you want to test a real cluster rather than an xUnit pseudo cluster, please refer to examples: https://github.com/osrg/earthquake/tree/master/example .

Q. Can I test standalone multi-threaded programs?

A. Yes, by fuzzing thread scheduling.

Q. Do I need to write `ExplorePolicy` manually? It's bothersome.

A. No, basically the default random behavior is enough.

Please feel free to open an issue in GitHub:
https://github.com/osrg/earthquake/issues