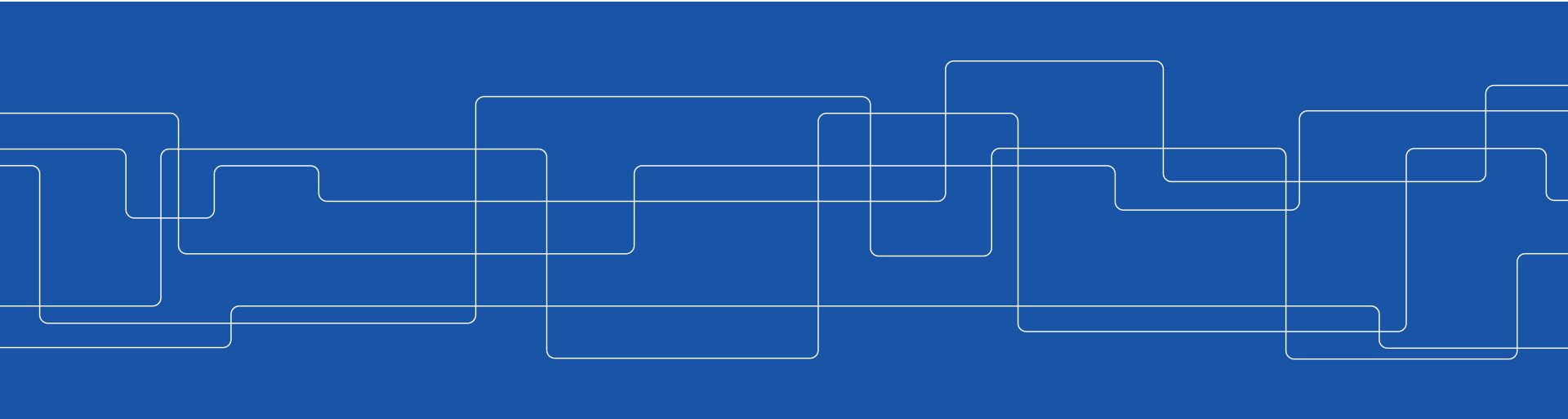
WASP | WALLENBERG AI, AUTONOMOUS SYSTEMS AND SOFTWARE PROGRAM

KTH ROYAL INSTITUTE OF TECHNOLOGY

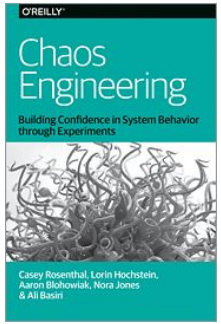
Application Level Chaos Engineering in JVM

Long Zhang, KTH Royal Institute of Technology

longz@kth.se



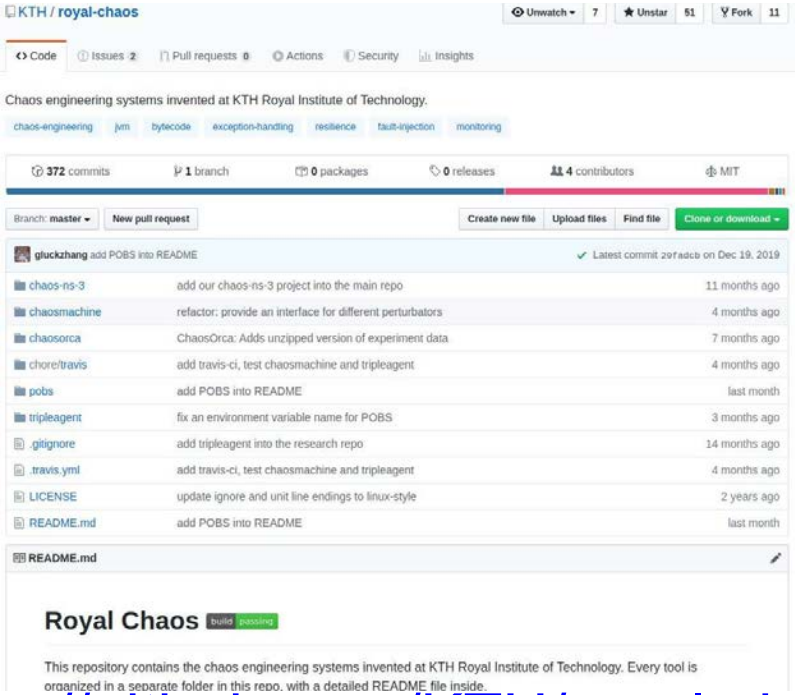
The Space of Chaos Engineering



- Examples of inputs for chaos experiments (Chapter 1, page 4):
  - Simulating the failure of an entire region or datacenter.
  - Partially deleting Kafka topics over a variety of instances to recreate an issue that occurred in production.
  - Injecting latency between services for a select percentage of traffic over a predetermined period of time.
  - Function-based chaos (runtime injection): randomly causing functions to throw exceptions.
  - Code insertion: Adding instructions to the target program and allowing fault injection to occur prior to certain instructions.
  - Time travel: forcing system clocks out of sync with each other.
  - Executing a routine in driver code emulating I/O errors.
  - Maxing out CPU cores on an Elasticsearch cluster.
- The opportunities for chaos experiments are boundless and may vary based on the architecture of your distributed system and your organization's core business value.

https://www.oreilly.com/ideas/chaos-engineering

Royal-Chaos @ GitHub



https://github.com/KTH/royal-chaos

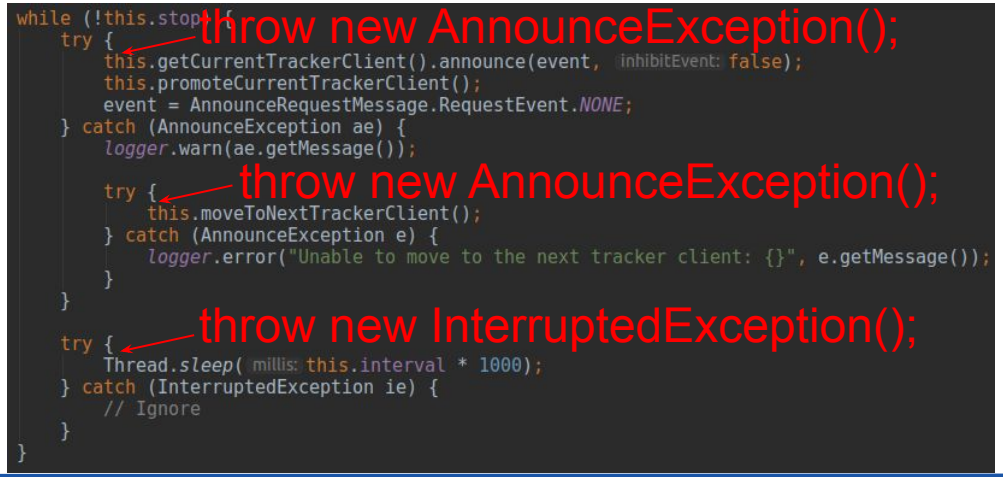
ChaosMachine

A Chaos Engineering System for Live Analysis and Falsification of Exception-handling in the JVM

https://arxiv.org/abs/1805.05246

ChaosMachine - Background Work

- Try-catch block short-circuit testing
  - A corresponding exception at the beginning
  - Make the whole try block invalid




```
while (!this.stop) {
    try {
        this.getCurrentTrackerClient().announce(event, inhibitEvent: false);
        this.promoteCurrentTrackerClient();
        event = AnnounceRequestMessage.RequestEvent.NONE;
    } catch (AnnounceException ae) {
        logger.warn(ae.getMessage());

        try {
            this.moveToNextTrackerClient();
        } catch (AnnounceException e) {
            logger.error("Unable to move to the next tracker client: {}", e.getMessage());
        }
    }

    try {
        Thread.sleep( millis: this.interval * 1000);
    } catch (InterruptedException ie) {
        // Ignore
    }
}
```

The Overview of ChaosMachine

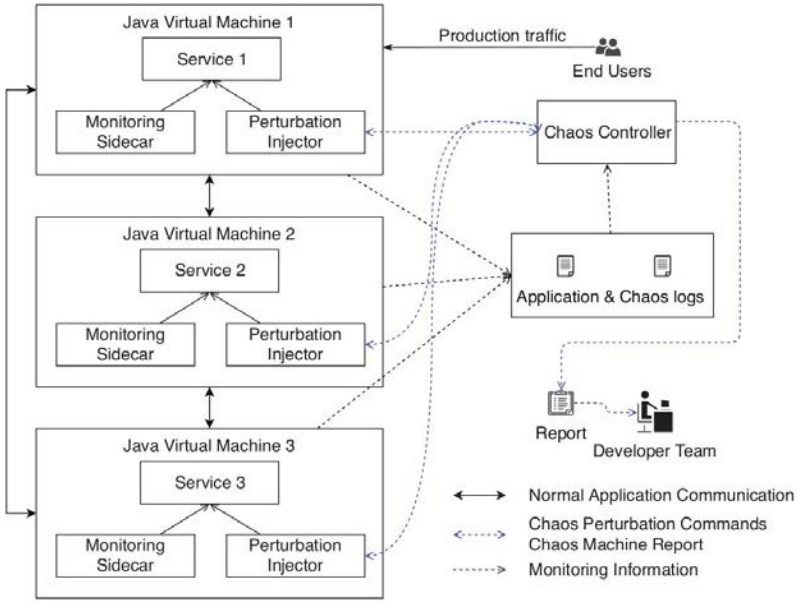


Fig. 1. The components of ChaosMachine

- Input
  - Arbitrary software in Java
  - Hypotheses
- Architecture
  - Monitoring sidecars
  - Perturbation Injectors
  - Chaos Controller
- Output
  - A report and monitoring logs

ChaosMachine - Hypotheses

- Resilience hypothesis
  - The observable behavior of the catch block, executed upon exception, is equivalent to the observable behavior of the try-block when no exception happens.
- Observability hypothesis
  - An exception caught in the catch block results in user-visible effects.
- Debug hypothesis
  - An exception caught in the catch block results in an explicit message in logs.
- Silence hypothesis
  - It fails to provide the expected behavior upon exception while providing no troubleshooting information whatsoever, i.e., it is neither observable nor debuggable.

ChaosMachine - What Can Be Learned

- Try-catch classification
  - Fragile ones
  - Possible resilient ones
- Logs handling mechanisms

ChaosMachine - Experiments On TTorrent

TABLE II
THE RESULTS OF CHAOS EXPERIMENTATION WITH EXCEPTION INJECTION ON 27 TRY-CATCH BLOCKS IN THE TTORRENT BITTORRENT CLIENT

Try-catch block information	Execution Anal./Expl.	Logged	Downl.	Exit status	System metrics	RH	OH	DH	SH
BEValue/getBytes,ClassCastException,0	41 / 1	yes	no	crashed	-		x	x	
BEValue/getNumber,ClassCastException,0	15 / 1	yes	no	crashed	-		x	x	
BEValue/getString,ClassCastException,0	37 / 1	yes	no	crashed	-		x	x	
BEValue/getString,UnsupportedEncodingException,1	37 / 1	yes	no	crashed	-		x	x	
ClientMain/main,CmdLineParser$OptionException,0	1 / 1	yes	no	crashed	-		x		
ClientMain/main,Exception,1	1 / 1	yes	no	crashed	-		x		
Announce/run,AnnounceException,0	1 / 60	yes	no	stalled	-		x	x	
Announce/run,InterruptedException,2	1 / 760	no	yes	normally	more threads			x	
Announce/run,InterruptedException,3	1 / 1	no	yes	normally	no diff	x			
Announce/run,AnnounceException,4	1 / 1	yes	yes	normally	no diff	x		x	
Announce/stop,InterruptedException,0	1 / 1	no	yes	normally	no diff	x			
ConnectionHandler/run,SocketTimeoutException,0	1290 / 1030	no	yes	normally	no diff	x			
ConnectionHandler/run,IOException,1	1290 / 1	yes	yes	stalled	higher cpu			x	
ConnectionHandler/run,InterruptedException,2	1290 / 2	yes	no	stalled	no diff			x	
ConnectionHandler/stop,InterruptedException,0	1 / 1	no	yes	normally	no diff	x			
ConnectionHandler$ConnectorTask/run,Exception,0	50 / 50	yes	no	stalled	no diff			x	
Handshake/craft,UnsupportedEncodingException,0	50 / 48	yes	no	stalled	no diff			x	
PeerExchange/send,InterruptedException,0	90763 / 210	no	no	stalled	no diff				x
PeerExchange/stop,InterruptedException,0	46 / 44	no	yes	normally	no diff	x			
PeerExchange$OutgoingThread/run,InterruptedException,0	90805 / 32984841	no	no	stalled	higher cpu			x	x
PeerExchange$OutgoingThread/run,InterruptedException,1	90763 / 288	no	no	stalled	no diff			x	
PeerExchange$OutgoingThread/run,IOException,2	90805 / 43	yes	no	stalled	no diff			x	
PeerExchange$OutgoingThread/run,IOException,3	90763 / 46	yes	no	stalled	no diff			x	
Piece/validate,NoSuchAlgorithmException,0	2564 / 5427	yes	no	stalled	higher cpu			x	
HTTPAnnounceRespMessage/parse,InvalidBEncodingException,0	3 / 30	yes	no	stalled	no diff			x	
HTTPAnnounceRespMessage/parse,InvalidBEncodingException,1	3 / 30	yes	no	stalled	no diff			x	
HTTPAnnounceResponseMessage/parse,UnknownHostException,2	3 / 30	yes	no	stalled	no diff			x	
total: 27/52	460626 / 32992950	18/27	8/27	7/27	4/27	6/27	7/27	20/27	3/27

- Total try-catch blocks: 52
- Covered by workload: 27
- Possible resilient ones: 6
- Silent ones: 3

TripleAgent

Monitoring, Perturbation and Failure-obliviousness for Automated Resilience Improvement in Java Applications

https://arxiv.org/abs/1812.10706

A Chinese Kungfu in Chaos Engineering

Technique of Ambidexterity, Zhou Botong,
The Legend of the Condor Heroes

https://en.wikipedia.org/wiki/Zhou_Botong

TripleAgent - Example

- An invocation chain: m2 → m1 → m0

```
Class2 {
    public void m2() {
        try {
            new Class1().m1();
        } catch (EA a) {
            ...
        } catch (EB b) {
            ...
        }
    }
}
```

```
Class1 {
    public void m1() throws EA, EB {
        new Class0().m0();
    }
}
```

```
Class0 {
    public void m0() throws EA, EB {
        // a statement
        ...
    }
}
```

TripleAgent - Example

- An invocation chain: m2 → m1 → m0

```
Class2 {
    public void m2() {
        try {
            new Class1().m1();
        } catch (EA a) {
            ...
        } catch (EB b) {
            ...
        }
    }
}
```

```
Class1 {
    public void m1() throws EA, EB {
        new Class0().m0();
    }
}
```

```
Class0 {
    public void m0() throws EA, EB {
        // code injected with code transformation
        PAgent.throwExceptionPerturbation(key1);
        PAgent.throwExceptionPerturbation(key2);
        // a statement
        ...
        ...
    }
}
```

TripleAgent - Example

- An invocation chain: m2 → m1 → m0

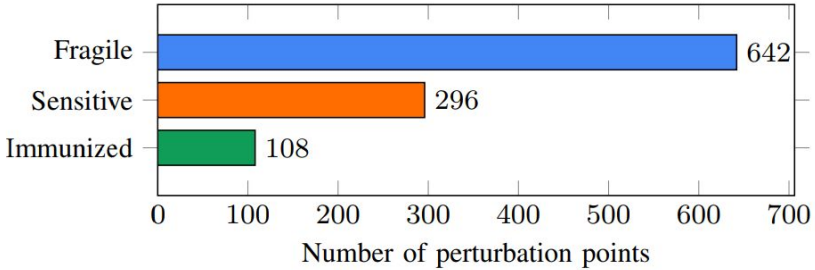
```
Class2 {
    public void m2() {
        try {
            new Class1().m1();
        } catch (EA a) {
            ...
        } catch (EB b) {
            ...
        }
    }
}
```

```
Class1 {
    public void m1() throws EA, EB {
        new Class0().m0();
    }
}
```

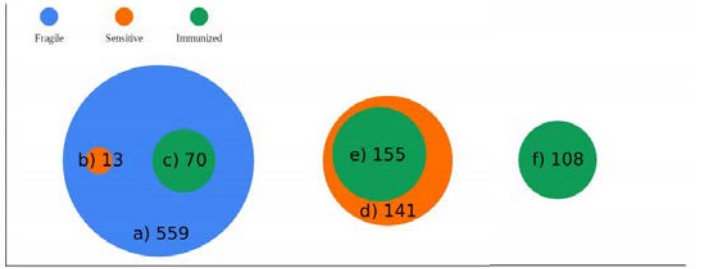
```
Class1 {
    public void foo() throws EA, EB {
        try {
            new Class0().m0();
        } catch (Exception e) {
            if (FOAgent.modelsOn(key)) {
                // the exception is silenced
            } else { throw e; }
        }
    }
}
```

```
Class0 {
    public void m0() throws EA, EB {
        // code injected with code transformation
        PAgent.throwExceptionPerturbation(key1);
        PAgent.throwExceptionPerturbation(key2);
        // a statement
        ...
        ...
    }
}
```

TripleAgent - Evaluation



Category of perturbation points



a) Fragile stays fragile, b) Fragile to sensitive, c) Fragile to immunized
d) Sensitive stays sensitive, e) Sensitive to immunized, f) Immunized stays immunized

Resilience improvement

TripleAgent identifies 238 perturbation points whose resilience could be improved by failure-oblivious methods.

TripleAgent - Overhead

- Application level: the execution time
- Operating system level: CPU usage etc.
- Binary code level: the code bloat

THE OVERHEAD OF AN EXPERIMENT ON TTORRENT

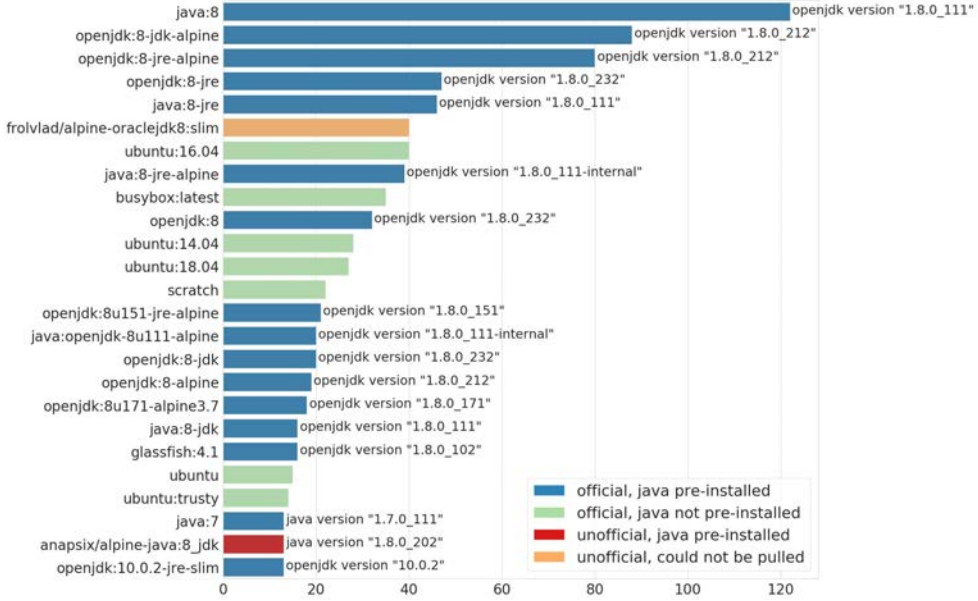
Evaluation Aspects	Original Version	Instrumented Version	Variation
Downloading time	20.4s	21.1s	3.5%
CPU time	15.0s	18.3	22.2%
Memory usage	47M	49M	4.3%
Peak thread count	30	32	6.7%
Relevant class files size	16.7KB	16.8KB	0.6%

POBS

Automatic Observability
for Dockerized Java Applications

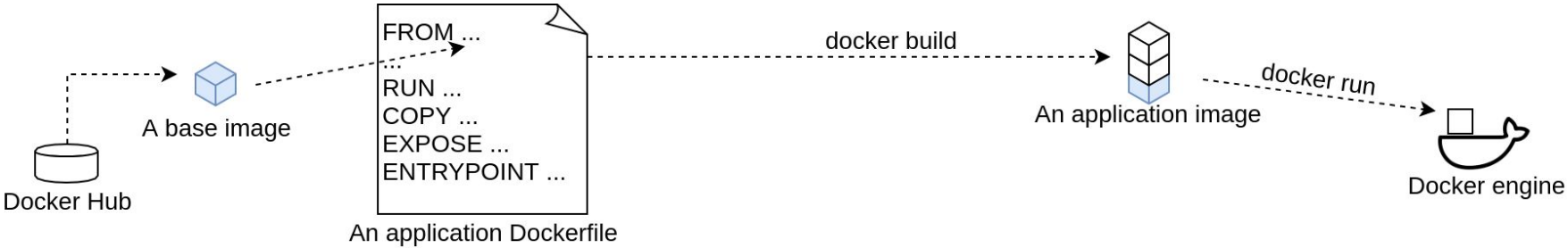
https://arxiv.org/abs/1912.06914

POBS - Empirical Study

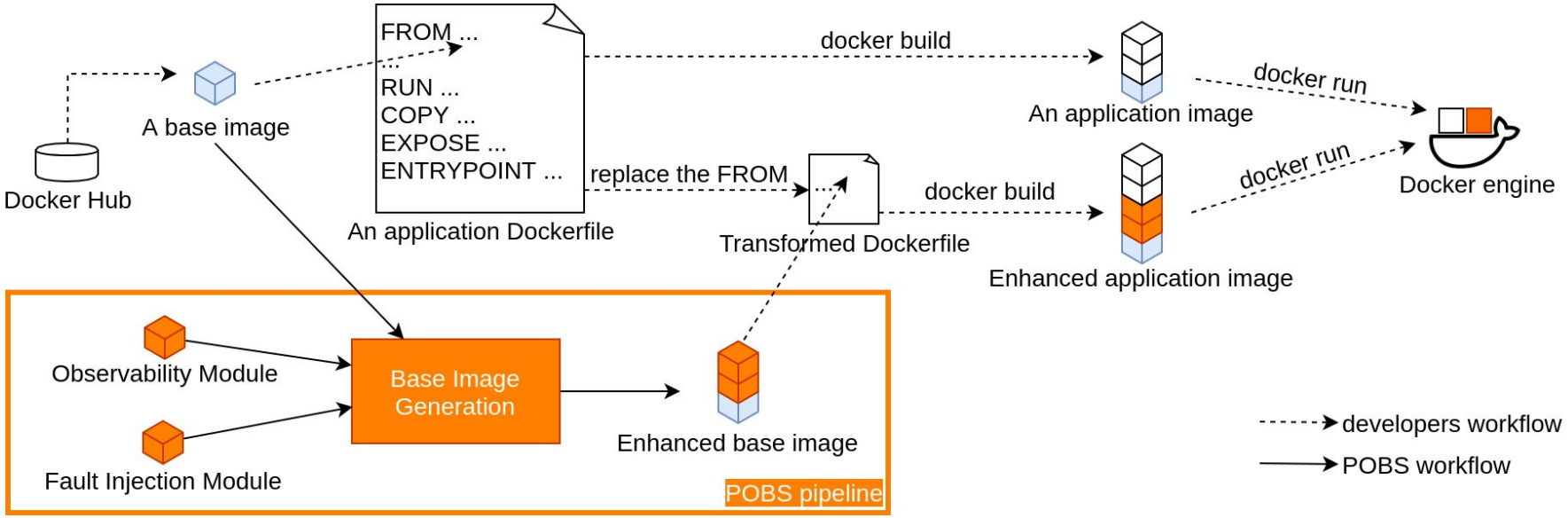


The 25 most popular base images across 1952 Dockerfiles in 589 GitHub java projects

POBS - Design

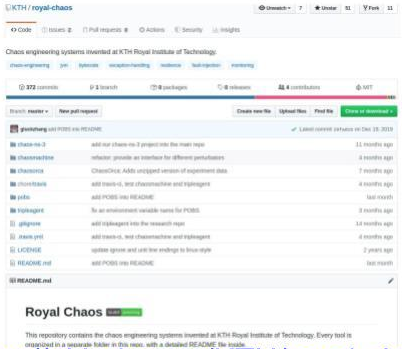

Docker Hub
A base image
FROM ...
...
RUN ...
COPY ...
EXPOSE ...
ENTRYPOINT ...
An application Dockerfile
docker build
An application image
docker run
Docker engine

POBS - Design


FROM ...
...
RUN ...
COPY ...
EXPOSE ...
ENTRYPOINT ...
An application Dockerfile
docker build
An application image
docker run
Docker engine
A base image
Docker Hub
replace the FROM
Transformed Dockerfile
docker build
Enhanced application image
docker run
Observability Module
Fault Injection Module
Base Image Generation
Enhanced base image
POBS pipeline
developers workflow
POBS workflow

DEMO Time!

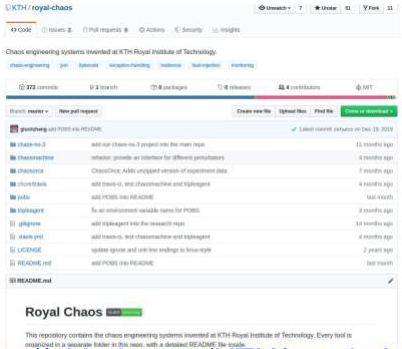
Royal-Chaos @ GitHub



https://github.com/KTH/royal-chaos

Summary

Royal-Chaos @ GitHub



https://github.com/KTH/royal-chaos

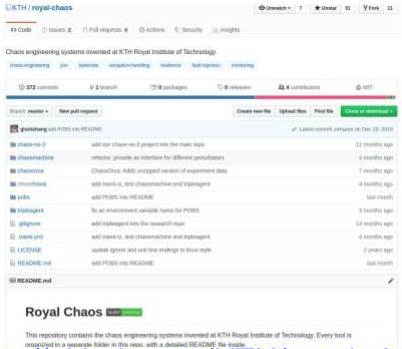
ChaosMachine

A Chaos Engineering System for Live Analysis and Falsification of Exception-handling in the JVM

https://arxiv.org/abs/1805.05246

Summary

Royal-Chaos @ GitHub



https://github.com/KTH/royal-chaos

ChaosMachine

A Chaos Engineering System for Live Analysis and Falsification of Exception-handling in the JVM

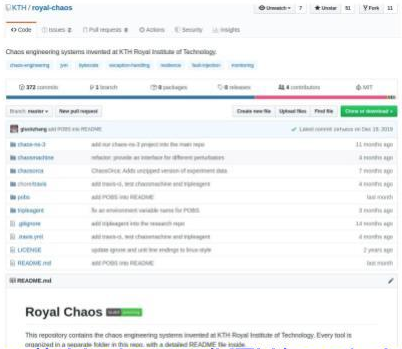
https://arxiv.org/abs/1805.05246

TripleAgent

Monitoring, Perturbation and Failure-obliviousness for Automated Resilience Improvement in Java Applications

https://arxiv.org/abs/1812.10706

Royal-Chaos @ GitHub



https://github.com/KTH/royal-chaos

ChaosMachine

A Chaos Engineering System for Live Analysis and Falsification of Exception-handling in the JVM

https://arxiv.org/abs/1805.05246

TripleAgent

Monitoring, Perturbation and Failure-obliviousness for Automated Resilience Improvement in Java Applications

https://arxiv.org/abs/1812.10706

POBS

Automatic Observability for Dockerized Java Applications

https://arxiv.org/abs/1912.06914

Thanks for listening!

Long Zhang longz@kth.se