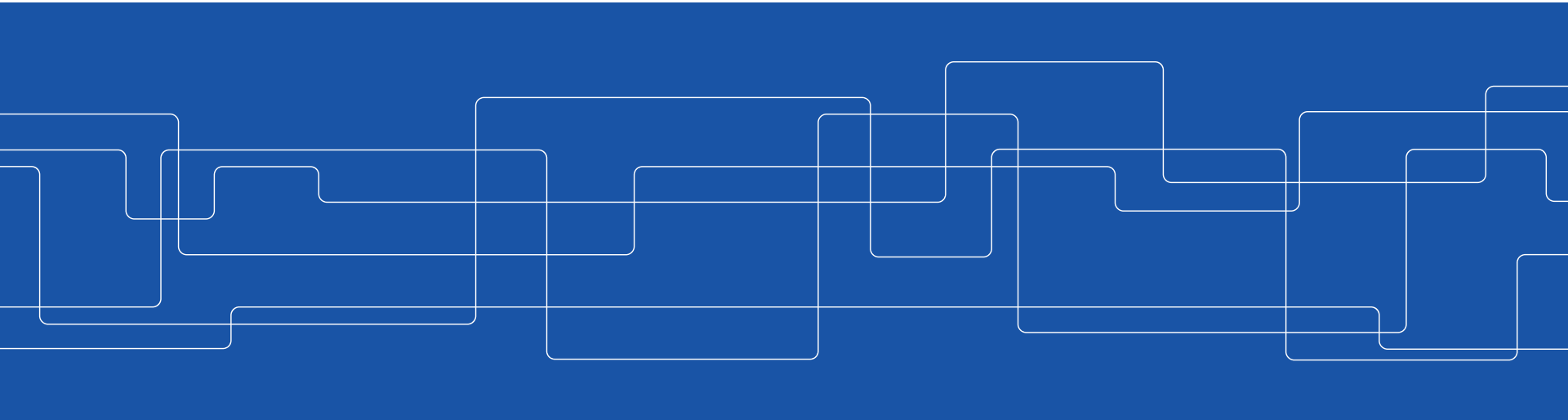




Introduction to Royal Chaos

Long Zhang, KTH Royal Institute of Technology

longz@kth.se



Our ChaosDay 2018





Royal-Chaos @ GitHub

KTH / royal-chaos

Unwatch 7 Unstar 36 Fork 10

Code Issues 1 Pull requests 0 Actions Security Insights

Chaos engineering systems invented at KTH Royal Institute of Technology.

chaos-engineering jvm bytecode exception-handling resilience fault-injection monitoring

329 commits 1 branch 0 releases 4 contributors MIT

Branch: master New pull request Create new file Upload files Find file Clone or download

gluckzhang feature: base image generator pretty prints the test results Latest commit eb61e0d 5 days ago

chaos-ns-3	add our chaos-ns-3 project into the main repo	8 months ago
chaosmachine	refactor: provide an interface for different perturbators	last month
chaosorca	ChaosOrca: Adds unzipped version of experiment data	4 months ago
chore/travis	add travis-ci, test chaosmachine and tripleagent	last month
pobs	feature: base image generator pretty prints the test results	5 days ago
tripleagent	fix an environment variable name for POBS	7 days ago
.gitignore	add tripleagent into the research repo	11 months ago
.travis.yml	add travis-ci, test chaosmachine and tripleagent	last month
LICENSE	update ignore and unit line endings to linux-style	2 years ago
README.md	add a travis build status icon	last month

README.md

Royal Chaos build passing

This repository contains the chaos engineering systems invented at KTH Royal Institute of Technology. Every tool is organized in a separate folder in this repo, with a detailed README file inside.

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

TripleAgent

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

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: $m2 \rightarrow m1 \rightarrow m0$

```
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: $m2 \rightarrow m1 \rightarrow m0$

```
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: $m2 \rightarrow m1 \rightarrow m0$

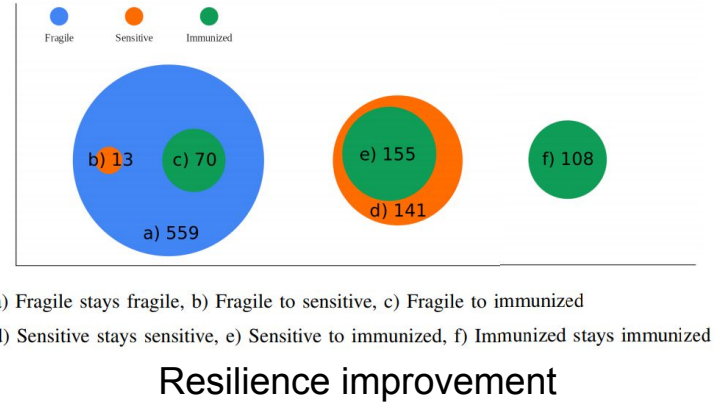
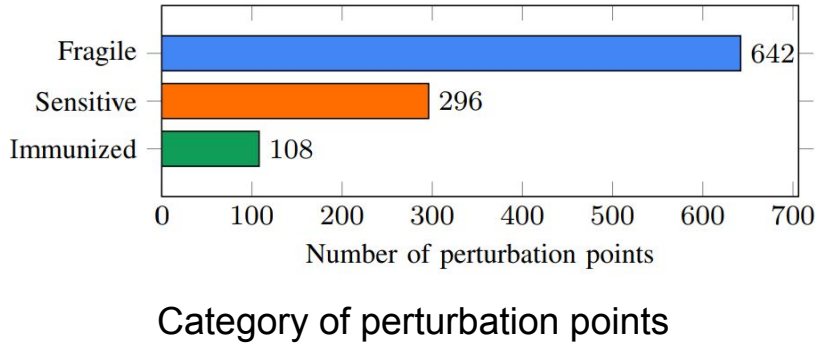
```
Class2 {
    public void m2() {
        try {
            new Class1().m1();
        } catch (EA a) {
            ...
        } catch (EB b) {
            ...
        }
    }
}
```

```
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



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

TripleAgent - Overhead

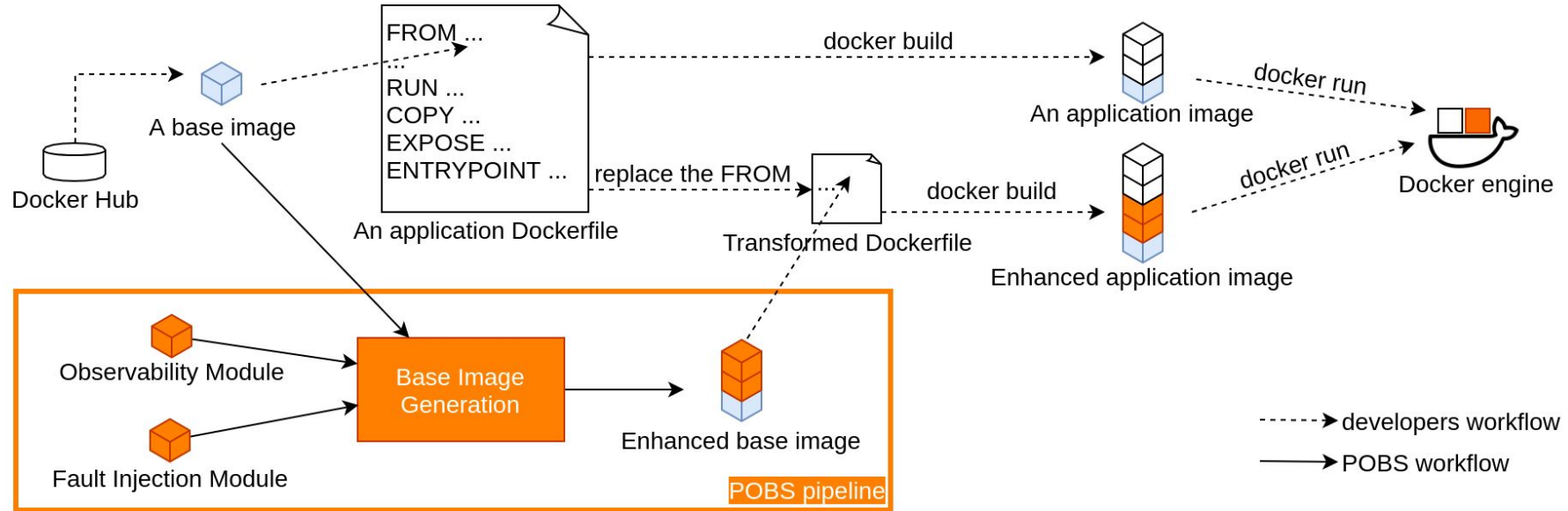
- Application level: the execution time
- Operating system level: CPU usage etc.
- Binary code level: the code bloat

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%

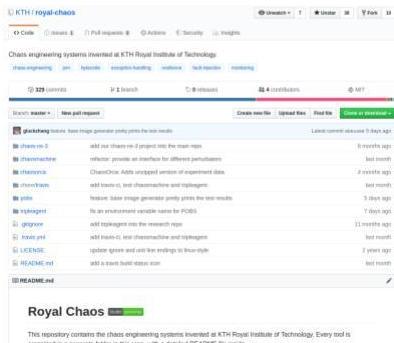
DEMO Time!

POBS - Design



DEMO Time!

Royal-Chaos @ GitHub



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

Long Zhang, longz@kth.se, Stockholm Chaos & Resilience Engineering Day 2019

TripleAgent - Example

- An invocation chain: $m2 \rightarrow m1 \rightarrow m0$

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

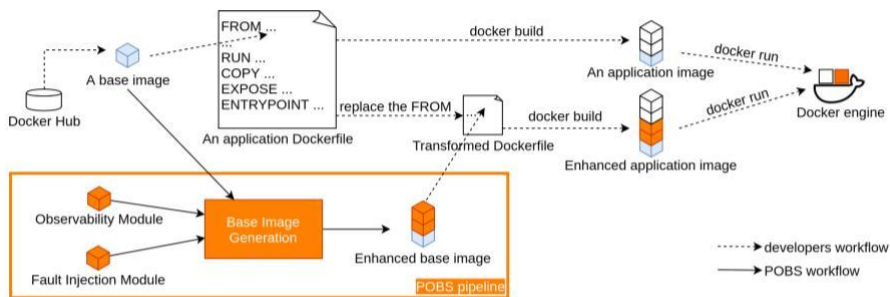
```
Class1 {
  public void m1() throws EA, EB {
    new Class0().m0();
  }
}
Class1 {
  public void foo() throws EA, EB {
    try {
      new Class0().m0();
    } catch (Exception e) {
      if (FOAgent.modelsOn(key)) {
        // the exception is silenced
      } else { throw e; }
    }
  }
}
```

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

3

Long Zhang, longz@kth.se, Stockholm Chaos & Resilience Engineering Day 2019

POBS - Design



DEMO Time!

12

Long Zhang, longz@kth.se, Stockholm Chaos & Resilience Engineering Day 2019

Long Zhang, longz@kth.se, Stockholm Chaos & Resilience Engineering Day 2019

13

Thanks for listening!

Long Zhang longz@kth.se

