



DICE Fault Injection Tool

Craig Sheridan, Darren Whigham

Flexiant Limited

Matej Artač

XLAB d.o.o.

DICE

Horizon 2020 Research & Innovation Action

Grant Agreement no. 644869

<http://www.dice-h2020.eu>



Funded by the Horizon 2020
Framework Programme of the European Union



- Motivation
- Fault Injection Tool
- DICE project and DevOps
- Conclusion



- Infrastructure equipment fails
- Expensive approach:
 - Making hardware completely reliable
 - Making sure that software never fails
- Reasonable approach:
 - Make systems resilient to failures
- Need for on-demand failures

Fault Injection Tool





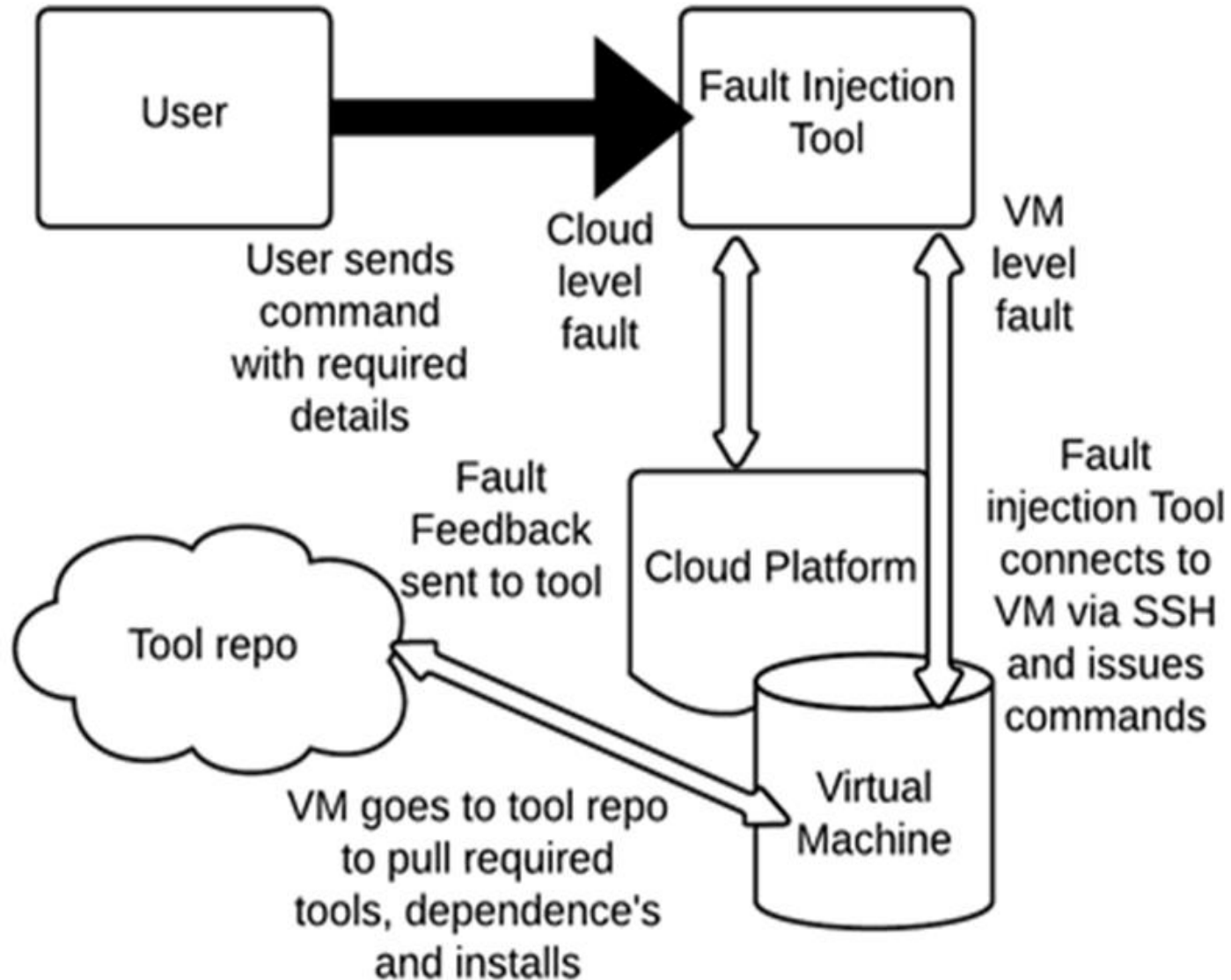
- Problem

- When testing, it is difficult to **reproduce faults** happening in production

- Goal

- Causes environmental problems or **failures on demand**
- Lets users **control** the **levels** and seriousness of induced failures

Design



Supported VM-level Faults



- **Shutdown** random VM (black list or white list)
- High **CPU** for VM
- High **Memory** usage for VM
- Block VM external **network** access
- High **Network Bandwidth** usage
- High **Disk I/O** usage
- Stop a **service** running on VM

Supported Cloud-level Faults



- **Shutdown** a Node
- High **CPU** for Node
- High **Memory** usage for Node
- High **Bandwidth** usage

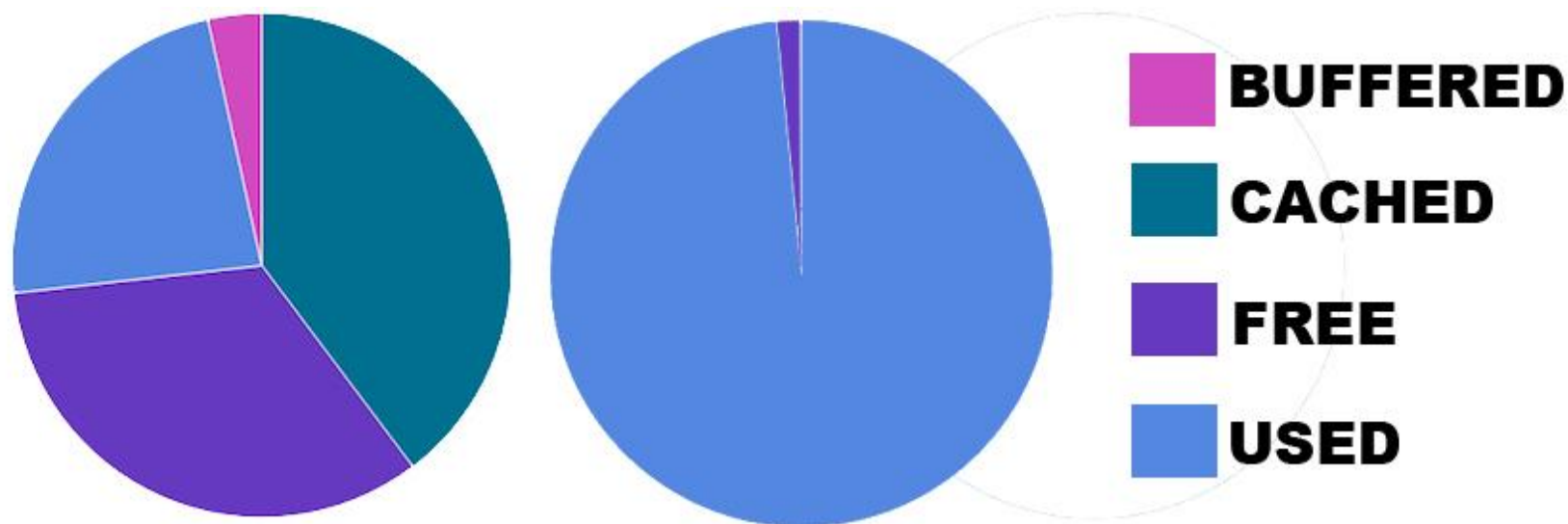
Usage example command



```
$ fleximonkey --stressmem \  
  2 \  
 2048m \  
 ubuntu@109.231.126.101 \  
 -no \  
 home/ubuntu/SSHKEYS/VMkey.key
```

test Loops
total memory
ssh host
no password

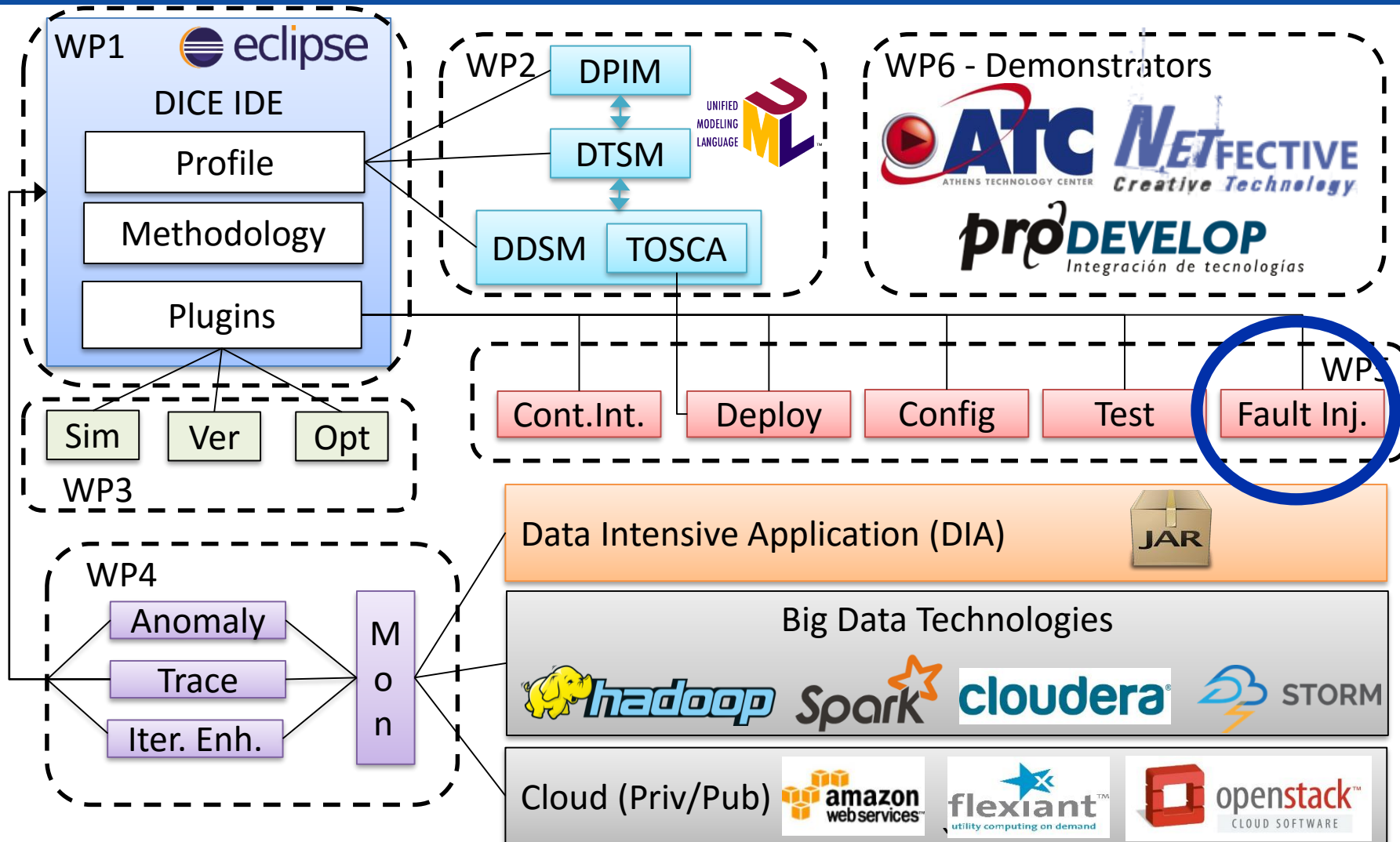
Usage example result





- ChaosMonkey: only accesses AWS & terminates VMs.
- Cocoma: developed but not supported, suffers from high resource usage, complex config & limited extensibility.
- FIT generates various cloud agnostic faults at VM & Cloud.
- Range of functions, greater flexibility to generate multiple faults
- Lightweight and only installs required tools and components on target VMs.
- Future extensibility in mind considering the needs and challenges of cloud service providers such as scalability and resiliency of the cloud consumption marketplace.

DICE Framework





- Continuous Integration/Testing
 - Regular resilience tests
- Anomaly detection
 - Induce faults, check for anomalies
 - Help with quality enhancement

Conclusion and future work



- Aimed at **developers** using **quality driven DevOps** approach
- Helps **cloud operators** perform stress analysis
- Difficulties and possibilities of **extensibility** by external users and investigating limitations
- Consider different topologies, operating systems and vendor agnostic cloud provider infrastructure
- Evaluate the **overhead** of operation
- **Containerised environments** will be a future target to help understand effect on microservices when injecting faults to the underlying host

Release: February 2017



<https://github.com/dice-project/DICE-Fault-Injection-Tool>