LCOO Extreme Testing

Technology & Tools Review

Gautam Divgi, Principal System Architect, AT&T

Extreme Testing Scope

Destructive Testing

Degrade, kill, crash various hardware and software components in the zone.

Benchmarking & KPI

Performance (CP+DP): Benchmark the control and data plane performance

Concurrency: Run large concurrent transactions against single resources (e.g. 1000 modifications on the same subnet)

Scale: Control and data plane performance on large number of resources (100,000 subnets, 50,000 cinder volumes, etc.)

Proposed Direction

Test boundaries cross

- Inducing crashes at load (performance & destructive)
- Large number of transactions on limited resources (performance & concurrent)
- Can have more examples

LCOO Extreme Testing

- Determine focus
- Tackling all tests is a significant effort
- Recommend just fault injection & KPI measurement

Technologies

Rally (openstack project)

- Openstack project
- Control plane benchmarking / performance
- Extensive capabilities
- User defined hooks for fault injection (os-faults), sla measurement

Shaker (openstack project)

- Data plane benchmarking
- *Limited to network – doesn't do storage, compute, underlay, etc.*
- No hooks capability like Rally for extensions

os-faults (openstack project)

- Failure injection library
- Extendable
- *Depends on deployment drivers for discovery*
- Current functionality: restart, kill, network flaps (ifdown), freeze (SIGCONT)

Technologies

Stepler (Mirantis open source)

- Can be used for destructive testing
- Uses os-faults for failure injection
- *No open source scenarios (at least none I can find)*
- *Control plane specific – not clear on advantages over rally/tempest*
- *Similar to rally/os-faults for destructive testing – but scenarios are python code as opposed to yaml/json*

Cloud99 (Cisco open source)

- Mainly s/w based destructive scenarios
- Limited disruptor capabilities
- In beta version
- *Similar to rally/os-faults in functionality*

ops-workload-framework (OSIC open source)

- Similar to rally
- Seems to stress provisioning aspect (creating vm's, volumes, etc.)
- Unclear of why rally could not be used for the same use cases

Technologies

Enos (BeyondTheClouds open source)

- Integrates containerized openstack deployment with rally & shaker
- Benchmarking tool
- Aims to create reproducible experiments

OSProfiler (openstack open source)

- Trace openstack requests across services
- Useful for creating trace and playback functionality
- *Only control plane specific*

Resiliency Studio (AT&T - not yet open source)

- Front end tool
- Share, model & execute (via backends like rally/os-faults) scenarios
- Visualize results

Gaps

In slide deck by Sampath (https://openstack-lcoo.atlassian.net/wiki/display/LCOO/Extreme+Testing+Development+Proposal)

Gaps

- Test Trigger
- Test Manager
- Config Manager
- Fault Generator
- Report Generator

Gaps & Possible Solutions

Test Trigger

How do regular Openstack tests happen today?

Need to integrate this with it

Config Manager

Set up openstack environment

Provide discovery information to test manager

ENoS is a good option here

ENoS Gap: Only s/w and containerized openstack

Need to emulate various data center configurations without actual hardware

Data center emulator

Emulate racks, switches, routers, Load balancers, etc.

Can this be done using LXC/LXD and linux name spaces?

Need research

Test Manager

Automate and orchestrate extreme testing scenarios

Provide tools to visualize and collaborate

Resiliency Studio can play a role here

Use Ansible as a backend to orchestrate across various tools

Gaps & Possible Solutions

Test Executors

- Rally (CP) and Shaker (DP) are the optimal options
- *Shaker needs to be enhanced for other parts of the data plane*
- *Rally's os-faults hook needs to be enhanced for non-deterministic fault injection*
  - *Via randomized models (as opposed to saying at: iteration 200)*
- *Shaker needs better plugin functionality and os-faults hooks*
- **Big Gap:** Traffic models
  - All traffic models are "made up"
  - Not realistic

Fault Injection

- Recommend os-faults
- **Expand os-faults**
  - Degradations (tc, stress-ng, fio/bonnie++, RabbitMQ simulator.py, etc.)
  - More underlay & hardware API
  - Work with data center emulation & ENoS

Gaps & Possible Solutions

Reporting and Metrics

- Weak area for extreme testing
- Metrics collection needs to follow a different path
- *Should not share components with "stressed" components (e.g. can't use RabbitMQ notifications)*

Report Generator

- Rally has SLA measurement plugin
- Need serious SLA/KPI discussions and hard mathematics
  - E.g. How do you automatically determine success/failure without eyeballing graphs?

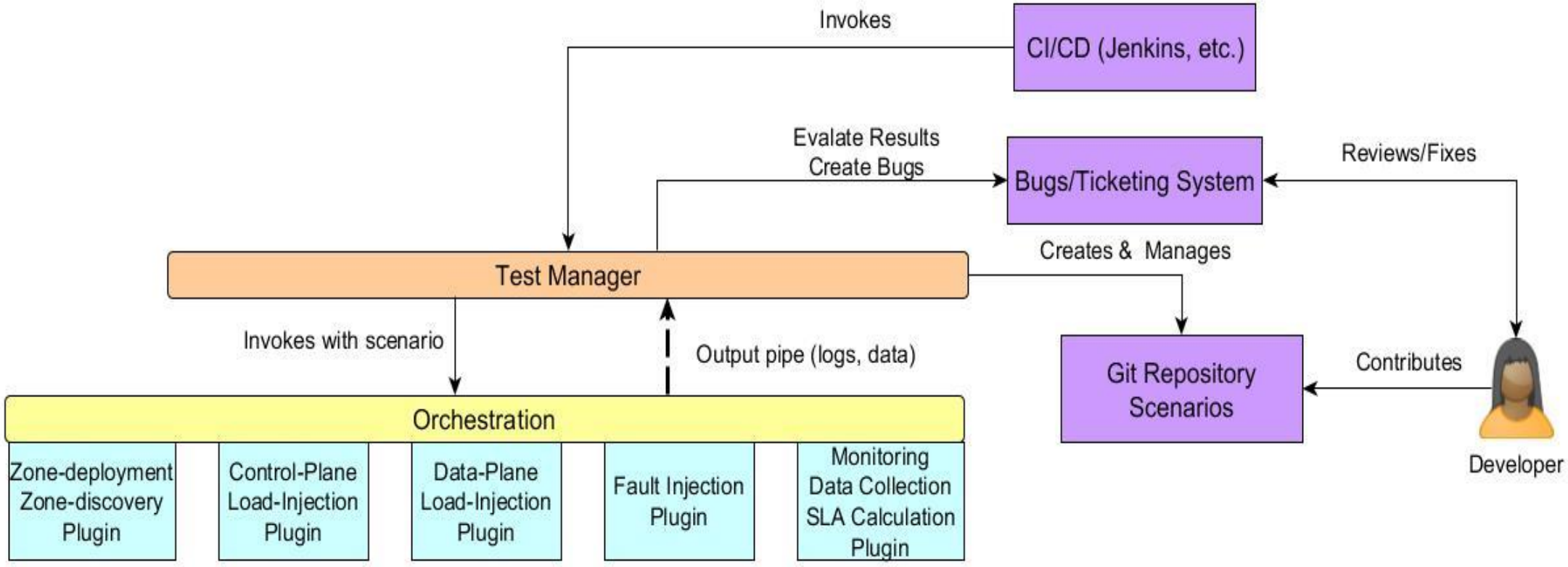
Record/Playback

- Record and play back production load / transactions
- Data plane load
- Control plane load
- Share models

Prioritized Path Forward

1. Tackle Control Plane & Software Faults (rally + os-faults)
    1. Most code already there – need more plugins
    2. os-faults: More fault injection scenarios (degradations, core dumps, etc.)
    3. Rally: Randomized triggers, SLA extensions (e.g. t-test with p-values)
2. Shaker enhancements (rally + shaker + os-faults)
    1. os-faults hooks mechanisms
    2. Storage, CPU/Memory (also cases with sr-iov, dpdk, etc.)
    3. os-faults for data plane software failures (cinder driver, vrouter, kernel stalls, etc.)
    4. Develop SLA measurement hooks
3. os-faults underlay enhancements & data center emulator
    1. os-faults: Underlay crash & degradation code
    2. Data center emulator with ENoS to model underlay & software
4. Traffic models & KPI measurement
    1. Realistic traffic models (CP + DP) and software to emulate the models
    2. Real KPI and scaled KPI to measure in virtual environments

Conceptual Architecture



Proposed Openstack Adaptation