Workload Definition Plan

OSIC Ops

Isaac Gonzalez

## Purpose of this document.

Is to define a plan to generate workloads in our OpenStack deployments that will help us gather meaningful data and close to production scenarios, that will help us to address each of the OSIC KPIs at scale.

## Overview

Understanding how Business-critical workloads (web servers, mail servers, app servers, etc) demand and use resources is key in capacity sizing, in infrastructure operation and testing, and in application performance management [1] ([link](http://www.ds.ewi.tudelft.nl/fileadmin/pds/reports/2014/PDS-2014-001.pdf)).

It is really hard to understand workload information due to its complexity (larger scale, heterogeneous, shared clusters). Based on “Statistical Characterization of Business-Critical Workloads Hosted in Cloud Datacenters” [1]([link](http://www.ds.ewi.tudelft.nl/fileadmin/pds/reports/2014/PDS-2014-001.pdf)) there are four key types of resources that can become bottlenecks for business-critical applications: CPU, disk I/O, memory and Network I/O.

In most cases, for testing, benchmarking and development purposes, the ideal workload to have in the cloud, will be as close as possible to a production environment. But defining and synthetically generating a production-like workload is a complex task, so we need to scope how a workload will look like for us, depending on the specific scenario and KPI we want to test.

## What we currently have

OSIC Ops team has successfully deployed Cloud 8, that currently is being used by OpenStack infra running gate jobs. OpenStack infra is a good workload that will help us to have a transient usage of our resources and services, plus we help the community providing them vms.

On the other hand, this is not a production-like environment (not talking about CI dedicated clouds) since Vms used for gate jobs may be active from seconds to around 1 hour. We need to complement this transient workload with resource specific generated workload, Cassandra DB R/W for memory for example.

We have Rally for exercising the APIs, and one scenario that will deploy a Wordpress stack with a vm that will be making requests to it, helpful for data plane availability testing.

## The plan

We will continue using OpenStack infra as our transient workload and get data in terms of resource usage to address the gaps via synthetic workloads.

This is a high level plan and tooling may change and always open to discussion, this is a starting point. We will address workload plan based on Focus items.

### High Availability of services

In this particular Item, we need to focus on exercising the APIs plus having a mixed resource workload.

* **Proposed workload**
  + Rally (create, delete resources constantly) [2]([link](http://rally.readthedocs.io/en/latest/))
  + OpenStack Infra jobs
  + Mixed resource workload: CPU, Memory, Network, disk.
    - Wordpress stack with Gatling [3]([link](http://gatling.io/#/))
    - Cassandra DB R/W
* **Tooling**
  + Rally
  + PerfKitBenchmarker [4]([link](https://github.com/GoogleCloudPlatform/PerfKitBenchmarker))
* **Test case example**
  + Metric
    - Control plane services availability in a given period of time while generating disruption on services.
  + Expected Result
    - No API downtime after error injection

### High Availability on VMs

On this Item we want to make sure that we are consuming resources intensively (depending on the metric, for live migration metrics we might want to focus on memory and disk usage). We don’t want to live migrate or check ssh on empty VMs.

* **Proposed Workload**
  + Mixed resource workload: CPU, Memory, Network, disk.
    - Wordpress stack with Gatling
    - Cassandra DB R/W
    - Network I/O
    - Online store
* **Tooling**
  + Rally
  + PerfKitBenchmarker
  + YCSB [5]([link](https://github.com/brianfrankcooper/YCSB/wiki))
* **Test case example**
  + Metric
    - Required time to recover vm after compute node failure.
  + Expected result
    - All failed vms on error injected compute nodes are living on a working compute node.

### Enterprise Deploy at Scale & Incremental Scale

Here, we want to have mixed workload from all four key resources and API activity, in case of incremental scale we would check availability in control and data plane while scaling, check that resources persists, then scale workload to meet the new resource capacity of the cloud.

* **Proposed workload**
  + Rally (create, delete resources constantly
  + Mixed resource workload: CPU, Memory, Network, disk.
    - Wordpress stack with Gatling
    - Cassandra DB R/W
* **Tooling**
  + Rally
  + PerfKitBenchmarker
* **Test case example**
  + Metric
    - Time needed to scale existing infrastructure (Compute, storage)
  + Expected Result
    - No downtime for data and control plane
    - Total time needed to scale infrastructure and steps in procedure.

### Zero-Downtime Upgrades.

The business scenario here will be, upgrade Openstack without downtime for services and tenant workloads, plus minimizing the time needed to perform upgrade.

We want a mixed and constant workload in the tenant and API requests to get downtime if any during the process.

* **Proposed workload**
  + Rally (create, delete resources constantly)
  + Mixed resource workload: CPU, Memory, Network, disk.
    - Wordpress stack with Gatling
    - Cassandra DB R/W
* **Tooling**
  + Rally
  + PerfKitBenchmarker
* **Test case example**
  + Metric
    - Time and HA of control and data plane while upgrading
  + Expected Result
    - No downtime for data and control plane
      * Average time of instance downtime if any
      * Number of failures on creating resources
    - Total time needed to upgrade.

## Assumptions/Risks

* This workloads are not intended to benchmark hardware or any vendor specific feature.
* Developers are needed
* This should be a cross-project effort since workloads will be used to test every single KPI

## Dependencies

* Project Engineering support to provide inputs on business critical workloads that would facilitate exercising the respective services at scale
* Project engineering support to respond to issues identified during workload execution
* Engineering development resources for workload generator framework

## Work estimation

Assuming 2 resources.

* Create Rally scenarios for exercising the APIs (Project Specific) - 1 sprint
* Create Rally scenarios for wordpress workload - 2 sprints
* Cassandra DB workload on PerfKitBenchmarker - 1 sprint
* YCSB executing operations in Cassandra - 1 sprint
* Project specific resource need? - 1 sprint per workload.
* IBM Interop challenge - 2 sprint after Barcelona
* Nice to have: Workload Generator Framework - 5 sprints if developed in OSIC. (Resources needed)
* Limit the resources for servers to perform like commodity servers (cgroups) - 1 sprint
* Create monitoring VMs - 1 sprint

## Summary and next steps

## Resources

[1] <http://www.ds.ewi.tudelft.nl/fileadmin/pds/reports/2014/PDS-2014-001.pdf>

[2 ]<http://rally.readthedocs.io/en/latest/>

[3] <http://gatling.io/#/>

[4] <https://github.com/GoogleCloudPlatform/PerfKitBenchmarker>

[5] <https://github.com/brianfrankcooper/YCSB/wiki>