
Chasing 1000 nodes scale

— Dina Belova (Mirantis)
Aleksandr Shaposhnikov (Mirantis)
Matthieu Simonin (Inria) —

Who's here?



Dina Belova

Aleksandr Shaposhnikov



Matthieu Simonin

Agenda

- OpenStack Performance Team - who are we?
- What is 1000 nodes experiment about?
- Test environments
- Observations
- Lessons learnt
- Q&A

Performance Team

- Performance team: since Mitaka summit
- Part of Large Deployment Team
- Defining the performance testing and benchmarking methodologies on various scale
- Most common tools used:
 - Control plane, density, dataplane and reliability OpenStack testing: **Rally, Shaker, os-faults**
 - Other tests: **OSprofiler, sysbench, oslo.messaging simulator**, other tools
- Helping drive found solutions within OpenStack libraries and projects
- Focused on sharing knowledge community-wide

Performance Team

- Posting all data to Performance Docs
 - <http://docs.openstack.org/developer/performance-docs/>
- Sharing all tests we've run and all results for these experiments
- This data is used to improve OpenStack and underlying technologies as well as to choose best cloud topologies

The screenshot shows the OpenStack Performance Documentation page. At the top is a navigation bar with links: Home, Projects, User Stories, Community, Blog, Wiki, and Documentation (which is underlined in red). Below the navigation bar is a sidebar on the left containing a 'Table Of Contents' with links to 'Performance Documentation' (sub-links: Abstract, Contents), 'Next topic' (1. Introduction), 'Project Source', 'This Page', 'Show Source', and a 'Quick search' box with a 'Go' button. The main content area on the right is titled 'Performance Documentation' and contains an 'Abstract' section followed by a paragraph about the documentation's purpose. Below the abstract is a 'Contents' section with a bulleted list of links to various documents, including '1. Introduction', '2. Methodologies', '3. Labs', and '4. Test Plans' with their respective sub-topics.

Home Projects User Stories Community Blog Wiki Documentation

Performance Documentation

Abstract

This documentation section aims to introduce ideas and best practices on how OpenStack cloud may be exercised to evaluate its performance, what can be called performance at all and how it can be measured. Here you will find both testing/probing methodologies and test plans, proposed by OpenStack community members (as well as results of their execution against various cloud topologies).

Contents

- [1. Introduction](#)
 - [1.1. What is the reason for writing this?](#)
 - [1.2. Who will be interested in this?](#)
 - [1.3. Who is contributing to this guide?](#)
- [2. Methodologies](#)
 - [2.1. Tools](#)
 - [2.2. Methodology for testing Hyper-Scale](#)
- [3. Labs](#)
 - [3.1. Intel-Mirantis Performance-Team Lab](#)
- [4. Test Plans](#)
 - [4.1. 1000 Compute nodes resource consumption/scalability testing](#)
 - [4.2. Measuring performance of container repositories](#)
 - [4.3. OpenStack control plane performance test plan](#)
 - [4.4. OpenStack control plane density testing](#)
 - [4.5. SQL Database Test Plan](#)

1000 nodes experiment : what is it ?

- 1000 nodes = 1000 compute nodes
- Control plane speed/latencies/limits evaluation on scale
- Core underlying services evaluation (mysql,rabbitmq) for scale
- Study of
 - the services resource consumption
 - potential bottlenecks
 - key configuration parameters
 - the influence of services topologies

1000 nodes: experiment methodology

Deployment and Benchmark/Monitoring and Analysis tools

- Containers
 - Simplifies CI/CD
 - Granularize services/dependencies
 - Flexible placement
 - Simplifies orchestration
- cadvisor + collectd / influxdb / grafana
- Rally Benchmarks (boot-and-list instance scenario)
- Heka + ElasticSearch + Kibana

1000 nodes experiment : environments



- Mesos + Docker + Marathon as a platform for Openstack (15 nodes with 2x , 256GB RAM, 960GB SSD)
- Containerized OpenStack services (Liberty release)
- Modified Nova-Compute libvirt driver to skip run of qemu-kvm



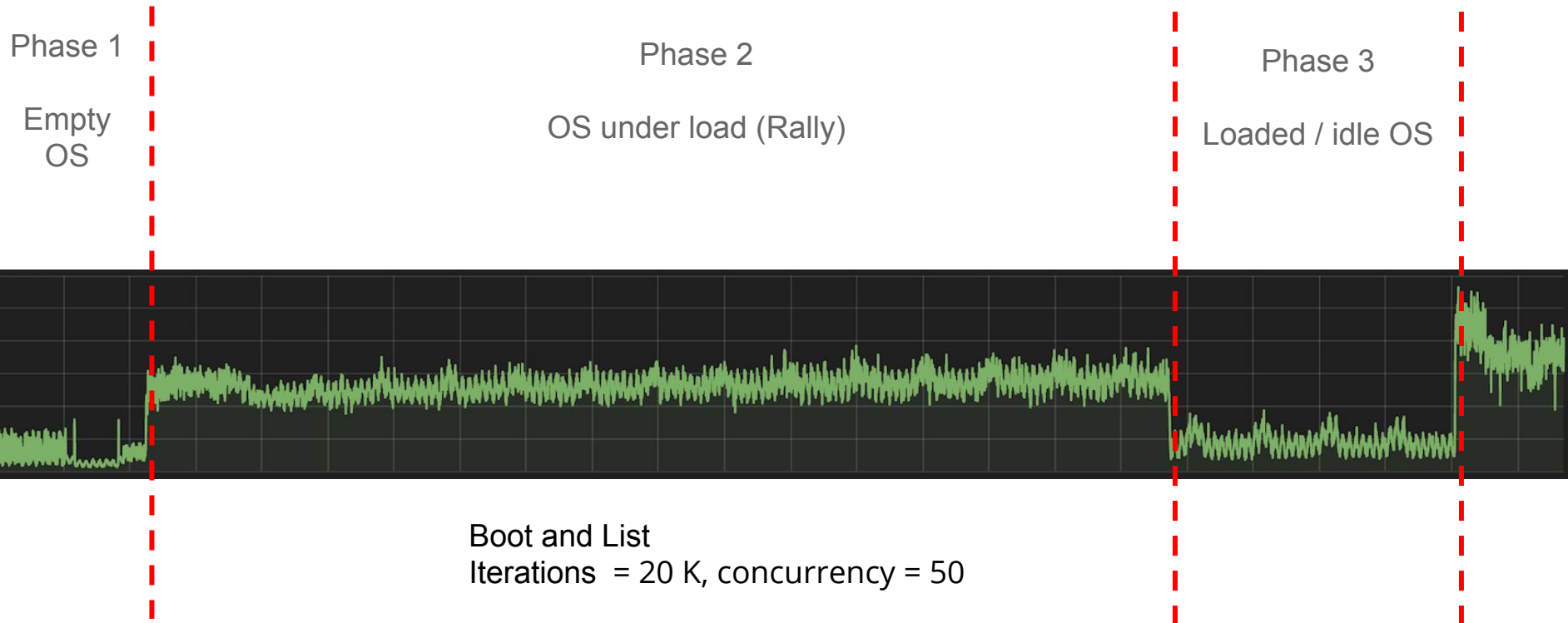
- ~ 30 nodes with powerededge 2xE5-2630, 128GB RAM, 200GB SSD + 3TB HDD (Grid'5000)
- Containerized OpenStack services (Mitaka release)
- Augmented Kolla tool
- Use of fake drivers



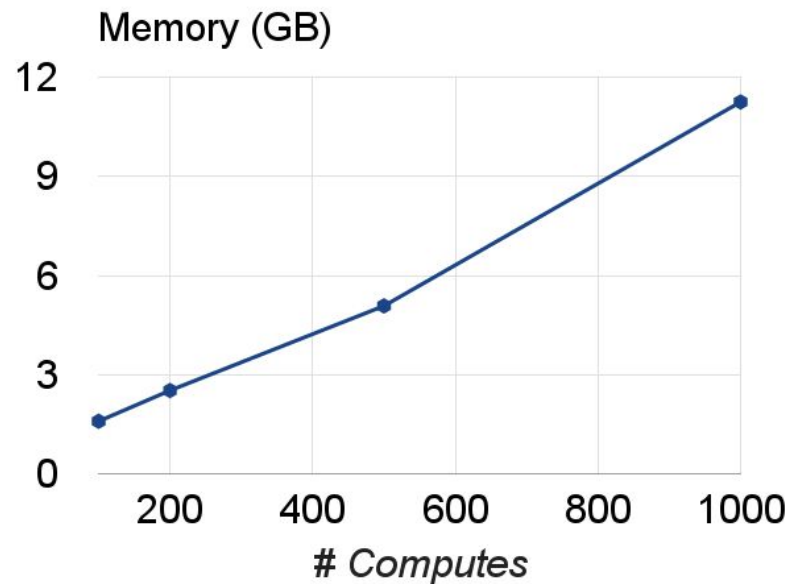
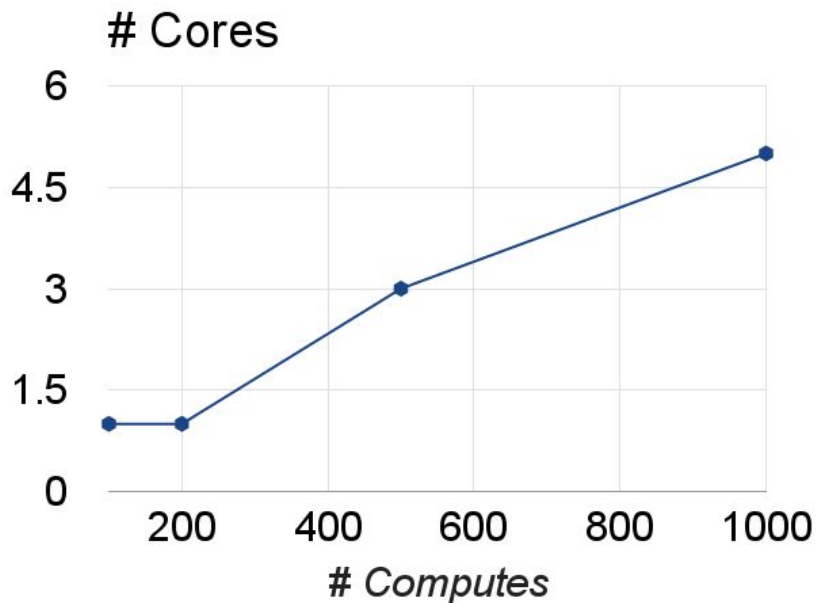
Code available :

<https://github.com/BeyondTheClouds/kolla-g5k>

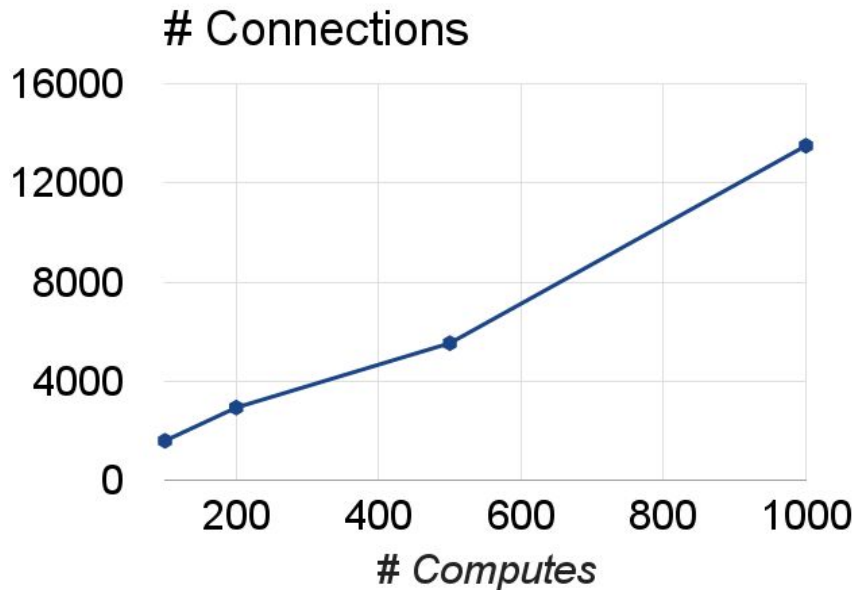
1000 nodes : experiment process



1000 nodes : RabbitMQ (Empty OS)



1000 nodes : RabbitMQ (Empty OS)



- CPU / RAM / Connections
Increase linearly with # Computes
- Connections : 15K with 1000 computes
- RAM : 12 GB with 1000 computes

1000 nodes: RabbitMQ (OS under load)

- (Phase 2) RabbitMQ load is big enough but tolerable, 20 Cores, 17 GB RAM
- (Phase 3) Idle load/Periodic tasks, 3-4 Cores, 16GB RAM.



1000 nodes : database (Empty OS)

Database footprints are small even for 1000 computes

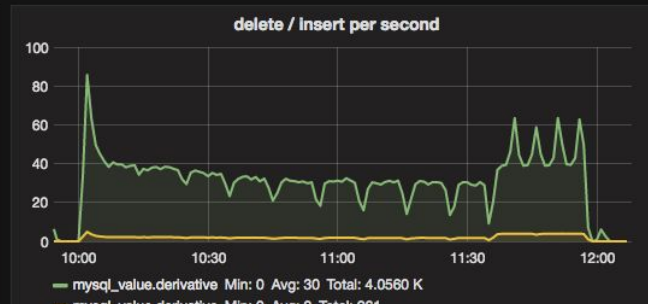
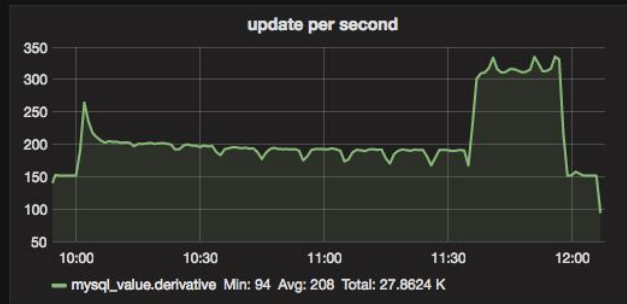
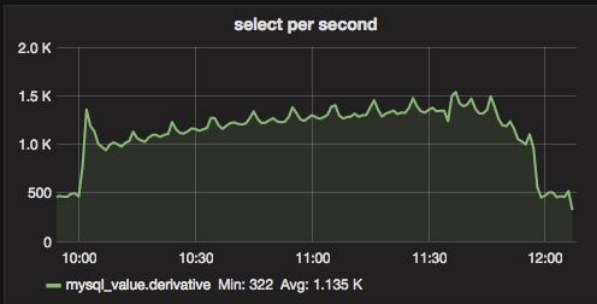
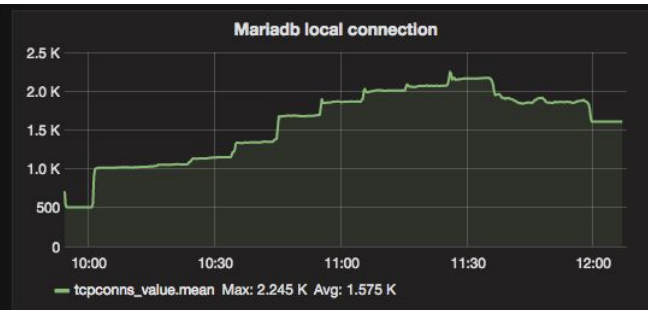
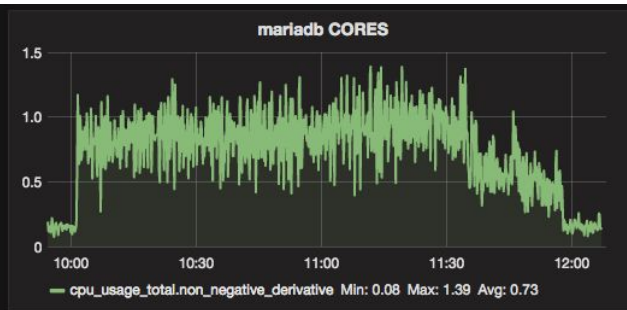
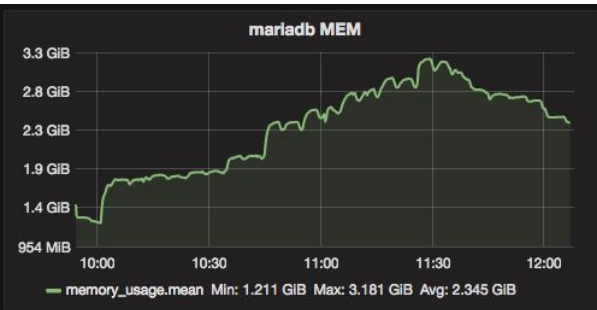
- 0.2 cores
- 600 MB RAM
- 170 opened connections

Effect of periodic tasks for 1000 computes

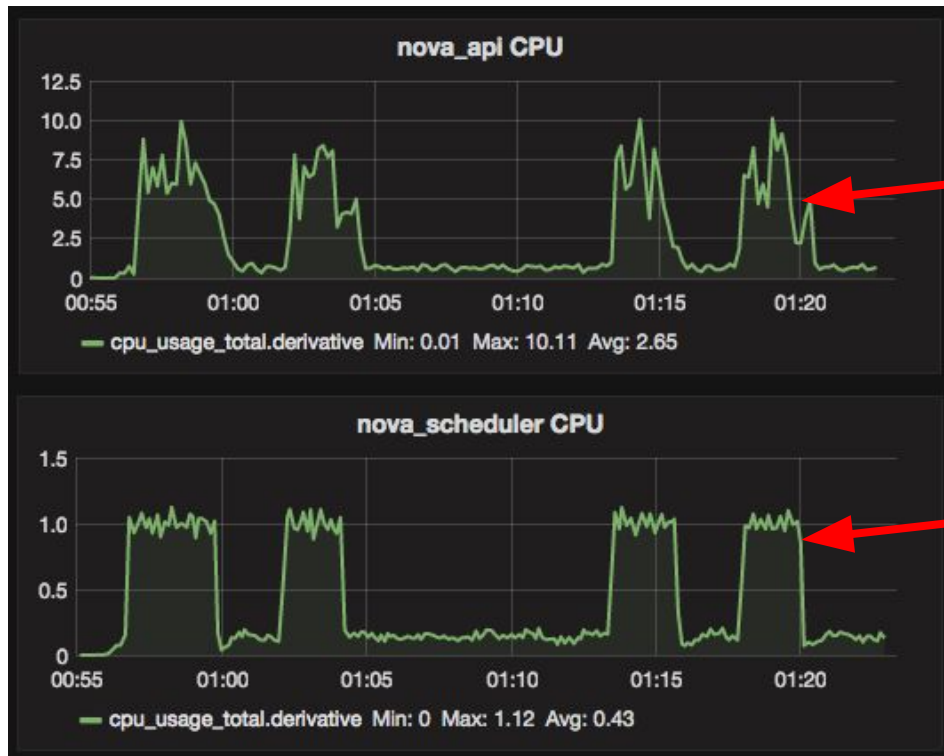
- 500 select / second
- 150 update / second

1000 nodes : database (OS under load)

- Database (single node) behaves correctly under load



1000 nodes: nova-scheduler (OS under load)



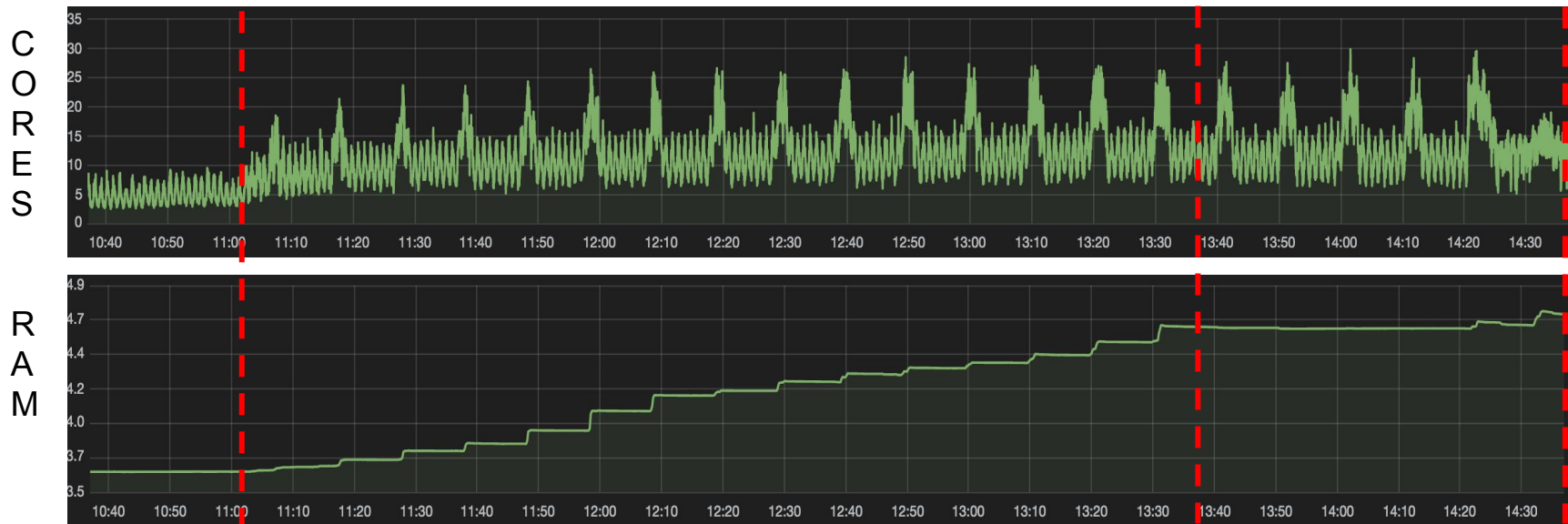
Rally benchmarks

Nova API : n workers

Scheduler : 1 worker only

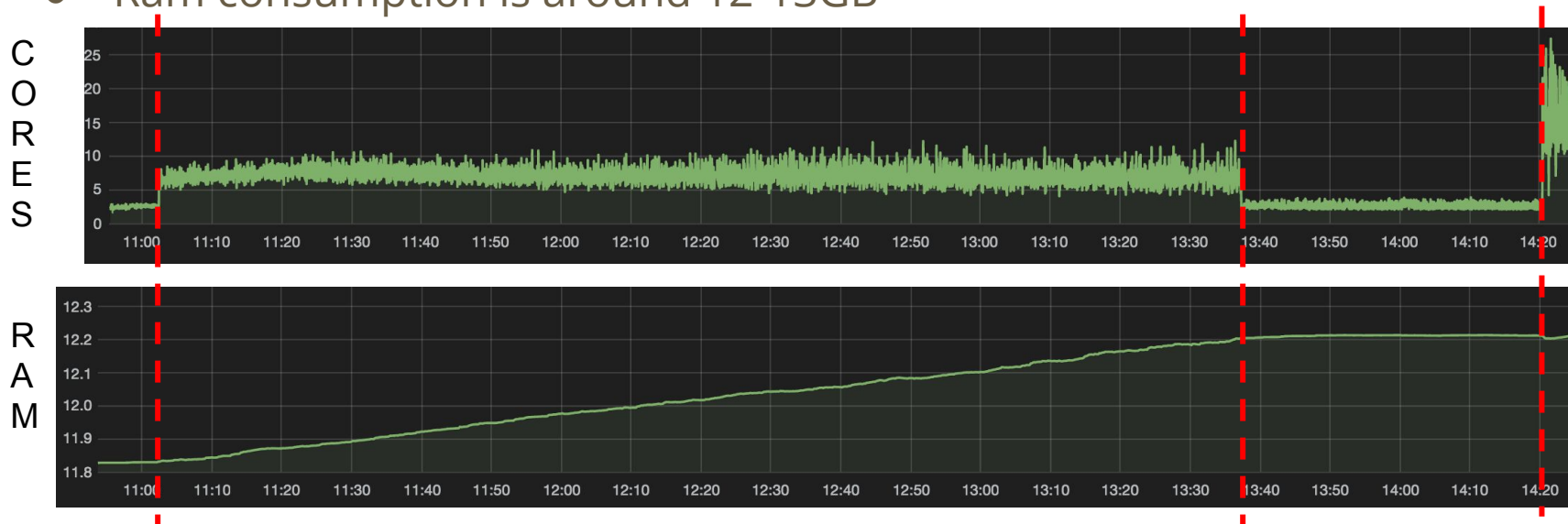
1000 nodes: nova-conductor (OS under load)

- One of the most loaded service
- Periodic tasks could be pretty hungry for CPU resources (up to 30 cores)
- There is no idle time for conductor unless cloud is empty



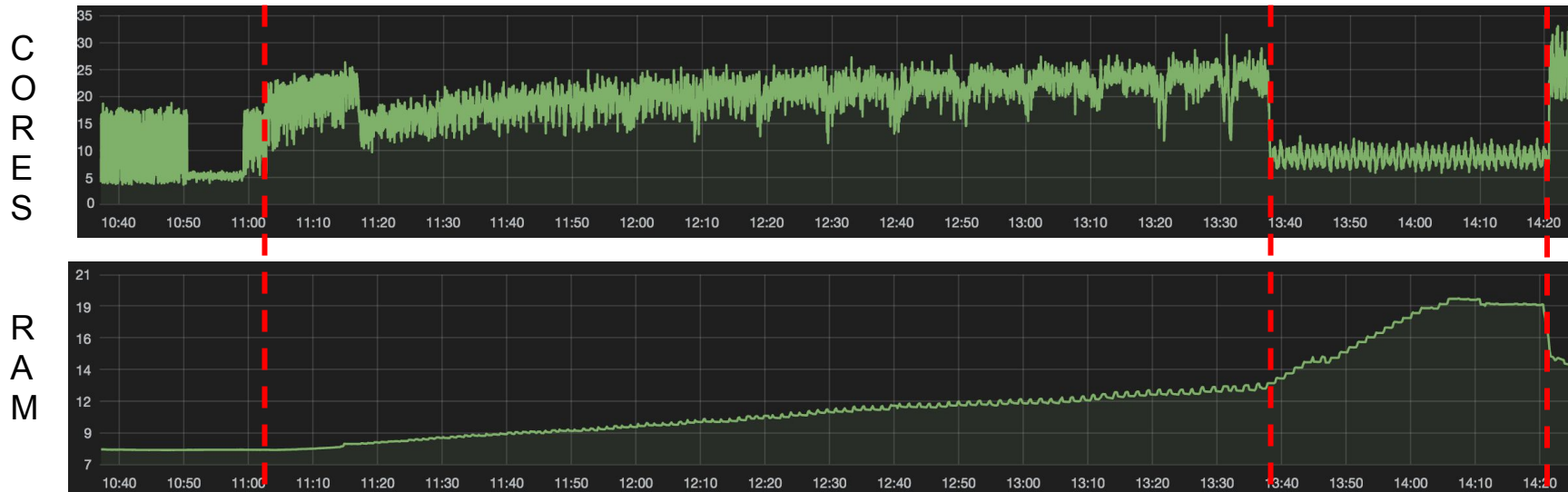
1000 nodes: nova-api

- Under test load it consumes ~ 10 Cores; under critical load ~25 Cores
- Without load/Periodic tasks ~3-4 Cores
- Ram consumption is around 12-13GB



1000 nodes: neutron-server(api/rpc)

- Under test load consumption is ~ 30 Cores, under critical ~ 35 Cores
- Just adding new nodes ~ 20 Cores, Periodic tasks ~ 10-12 Cores



Conclusion

1. Default number of API/RPC workers in OpenStack services wouldn't work for us if it tightened up to number of cores.
2. MySQL and RabbitMQ isn't a bottleneck at all. At least in terms of CPU/RAM usage. Clustered one's is an additional topic.
3. Scheduler performance/scalability issues.

Useful links

- 1000 nodes testing:
 - http://docs.openstack.org/developer/performance-docs/test_plans/1000_nodes/plan.html#reports
- Performance Working group
 - Team info: https://wiki.openstack.org/wiki/Performance_Team
 - Performance docs: <http://docs.openstack.org/developer/performance-docs/>
- Weekly meetings at 15:30 UTC, Tuesdays, **#openstack-performance IRC channel**: <https://wiki.openstack.org/wiki/Meetings/Performance>
- Sessions this week:
 - Today: **OpenStack Scale and Performance Testing with Browbeat**
(<https://www.openstack.org/summit/barcelona-2016/summit-schedule/events/15279>)
 - Wednesday: **Is OpenStack Neutron Production Ready for Large Scale Deployments?**
(<https://www.openstack.org/summit/barcelona-2016/summit-schedule/events/16046>)
 - Thursday: **OpenStack Performance Team: What Has Been Done During Newton Cycle and Ocata Planning**
(<https://www.openstack.org/summit/barcelona-2016/summit-schedule/events/15504>)

Q&A

Backup slides

OpenStack/Core services settings for 1000 scale

Nova-api: `database.max_pool_size = 50`

Nova-conductor: `conductor.workers` by default is a number of cores so be careful if it's too low

Nova-scheduler: you have to run ~ 1 scheduler per 100 compute nodes

Neutron-server: `default.api_workers=100`, `default.rpc_workers=20`

mysql/mariadb: `max_connections = 10240`

Linux: probably will have to tune `ulimits`, `net.core.somaxconn`, `tx/rx` queue on nics

Haproxy : increase `maxconns`, `timeouts`

Grid'5000

Grid'5000

- 1000 physical nodes (8000 cores)
- 10 sites geographically distributed
- 10GB ethernet between sites
- <http://www.grid5000.fr>

