Reproducible performance evaluations of different OpenStack deployments with EnOS

Fog/Edge/Massively Distributed Clouds (FEMDC) SIG

Beyond the Clouds: The DISCOVERY initiative

Ronan-Alexandre Cherrueau, Dimitri Pertin, Anthony Simonet, Matthieu Simonin, Adrien Lebre

Who am I?

Dimitri Pertin

Researcher post-doc at Inria, involved in the Discovery initiative:

Build a fully decentralized Cloud manager

Towards the Deployment and Reconfiguration of Cloud Applications
on top of Massively Distributed infrastructures

Fog/Edge/Massively Distributed Clouds FEMDC SIG enthusiast

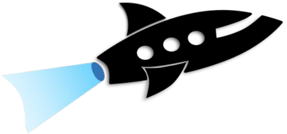
EnOS contributor (GitHub, documentation)

Agenda

1. Introduction (Discovery, FEMDC SIG)
2. OpenStack Performance Evaluations
3. EnOS: Experimental eNvironment for OpenStack
4. Overview of the Works Done with EnOS
5. Future Works and Conclusions

1. Introduction

Discovery initiative



Study Fog/Edge infrastructure:

- New form of Cloud infrastructure
- Many micro to nano data-centers (dozen of compute nodes)
- Micro/Nano data-centers must cooperate to provide Cloud Computing features

Motivations:

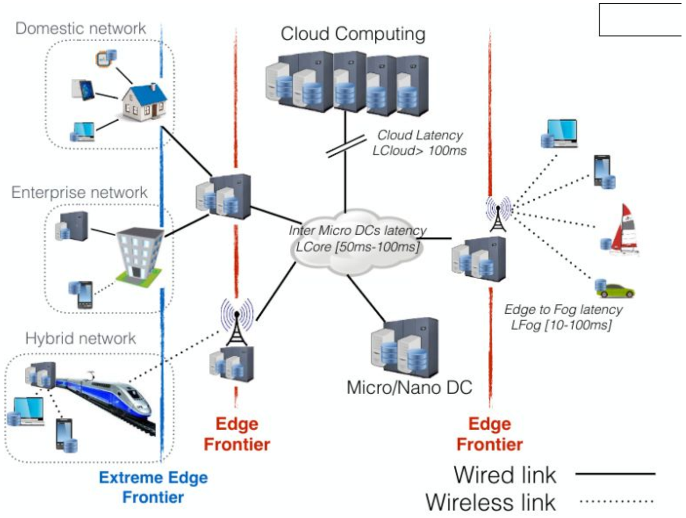
- Points of Presence (PoP) hold servers and routers at the edge of the network
- Deploy micro/nano DCs at PoPs

⇒ **Massively distributed cloud at the edge of the network**

Discovery initiative (cont'd)

Such infra offers a new paradigm: *Fog/Edge computing*

- Reliable – No single point of failure
- Governance – Capability to request an compute node in an australian PoP
- Reduces end-user to compute node latency – For latency-sensitive apps:
  - Internet of Things
  - Smart cars
  - Tactile Internet
  - NFV (telco)
  - …

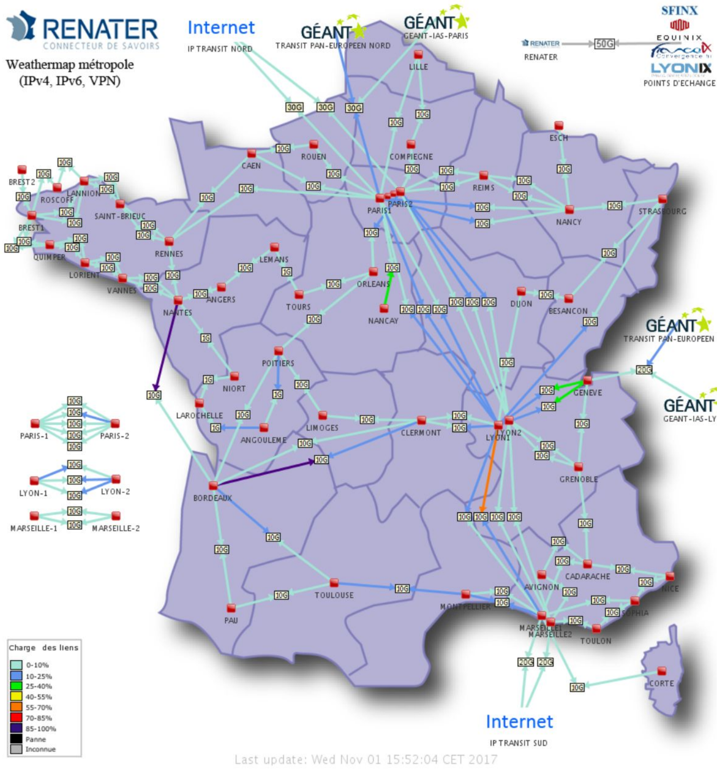


Discovery initiative (cont'd)

Infrastructure (Renater backbone):

- A red point is a Point of Presence (PoP)
- A micro data center in each PoP
- PoPs collaborate to offer Cloud Computing functionalities

Question: How to Operate such a Massively Distributed Cloud Infrastructure?



Last update: Wed Nov 01 15:52:04 CET 2017

Fog/Edge/Massively Distributed Clouds (FEMDC) SIG

- Investigate how OpenStack can address Fog/Edge Computing use-cases
- On-going actions:
  - Executive summary of Fog/Edge computing
  - Use-cases: talk given by Paul-André yesterday
  - **Evaluate** critical services for massively distributed OpenStack deployments
  - **Evaluate** architectural choices
  - …
- We had our F2F meeting on Monday (check our pad or wiki)

Investigate OpenStack in Fog/Edge

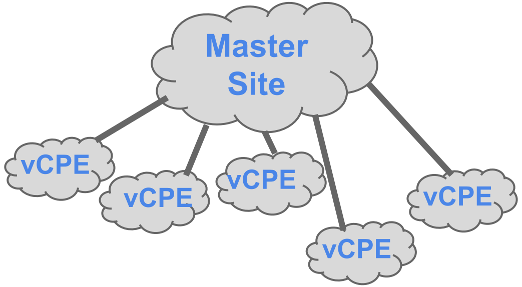
Centralized OpenStack services (e.g. database, message bus) can be potential bottlenecks:

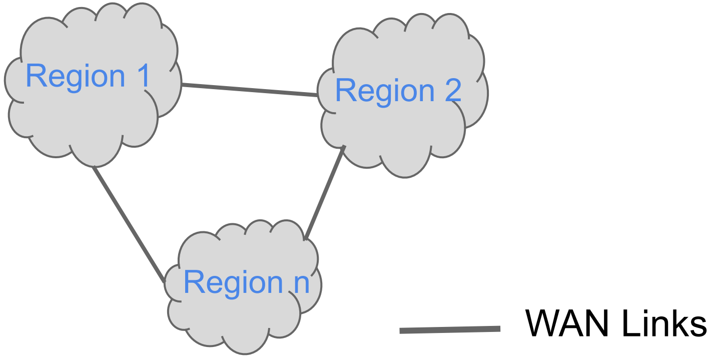
Several deployment possibilities:

1. Control services deployed at one site,

   many compute nodes on remote sites

2. Segregation technics
   - Cells (nova related)
   - Multi-regions (shared keystone)
3. Federated/brokering approaches





* vCPE: virtual Customer Premises Equipment

Investigate OpenStack in Fog/Edge

How to evaluate critical services in Fog/Edge context?

Which type of deployment is the most interesting/appropriate?

⇒ **We need a sandbox to help us conducting various performance analyses**

2. OpenStack performance evaluations

What is performance evaluation?

- Performance evaluations of a system
  - Set the system in a controlled and well-defined environment
  - Collect metrics and logs of the system in real-time and analyze them
    - Hardware metrics (e.g. CPU, RAM, network, disk consumption)
    - Service metrics (e.g. number of services requests, number of messages in queues)
  - When a specific and appropriate workload is applied on the system
    - “Don’t expect to measure significant network activity from writing bytes on local disk”
- Control-plane vs Data-plane evaluations
  - Control-plane: evaluation at the level of OpenStack controllers
    - e.g. database, message queue, API services, …
  - Data-plane: evaluation at the level of the components managed by OpenStack
    - e.g. compute instances, network, storage performances

Why perf evaluations for OpenStack?

- Configuration validation wrt some performance metrics
  - Find system bottlenecks in a specific context
    - e.g. investigating OpenStack in Fog/Edge infrastructures
  - Help to make the good design choices (e.g. how to deploy OpenStack services)
  - Find out appropriate settings for deployed services

- Continuous benchmarking (non-regression in the context of performance)
  - Help developers to validate new features

⇒ Validation needs performance experimentations to be **reproducible** by others

Performance WG

- For OpenStack, the Performance WG is in charge of defining and sharing the performance evaluation methodologies

- Share data about realized performance tests and results:
  - Methodologies
  - Labs/testbeds
  - Test plans
  - Test results

- Develop performance analysis tools
  - osprofiler: OpenStack profiling library
  - osfault: OpenStack fault-injection library

Challenges

1. Hard to compare different deployments
   a. Express different OpenStack configurations (e.g. release, enabled services, settings)
   b. Manage different configurations, and switch between them
      - *“Compare vanilla OpenStack vs OpenStack with customized version of Nova”*
      - *“Compare vanilla Neutron vs Neutron when a specific plugin is activated”*
2. Hard to reproduce or re-use experimental environments
   a. Express a specific environment (e.g. set network constraints between nodes)
   b. Switch from a testbed to another one
      - “*Reproduce automatically an experimental environment made in* testbedA *on* testbedB”
3. Hard to deploy an operable OpenStack
   a. Everybody is not devops (e.g. most academics are not)
      - “*Deploy automatically an OpenStack, even if I don’t know what it is*”

Challenges (cont'd)

4. Hard to run benchmarks:

   a. Different types of benchmark (control/data-plane)

   b. Express complex benchmarking scenarios

   c. Run automatically benchmarks

5. Hard to collect and explore the results:

   d. Many types of generated data: logs, config files, metrics

   e. Many services: monitoring, data stores and data visualization

   f. Collect metrics from many nodes

⇒ We look for **software-defined** benchmarks and experimentation **automation**

3. Experimental eNvironment for OpenStack (EnOS)

EnOS: Experimental Env. for OpenStack

- Motivation: Conducting performance analyses
  - **In a scientific and reproducible manner (automation)**
  - For different testbeds (small, large-scale)
  - Under different network topologies (traffic shaping)
  - Between different OpenStack releases and configurations
  - With any kind of benchmarks

  ⇒ We talk about ephemeral perf-oriented deployments: **not for production**

- Built on **Kolla**: OpenStack deployment tool leveraging Docker and Ansible
  - Amazing work to build and deploy Openstack services as Docker containers
  - Ability to highly customize OpenStack deployment and service settings

EnOS Workflow

```
1. $ enos deploy
```

- Read a deployment description file (topology + openstack configuration)
- Set an experimental environment (**solve challenges 1 and 2**)
- Deploy OpenStack (**solve challenge 3**)

```
2. $ enos bench
```

- Read a benchmark description file (test plan)
- Run benchmarks (**solve challenge 4**)

```
3. $ enos backup
```

- Backup confs, metrics and logs (**solve challenge 5**) for post-mortem analysis

1. EnOS deploy: Topology Description

```
$ cat ./basic.yml
resources:
  clusterA:
    control: 1
    network: 1
  clusterB:
    compute: 50

$ enos deploy -f basic.yml
```

```
$ cat ./advanced.yml
resources:
  clusterA:
    control: 1
    network: 1
    nova-conductor: 5
  clusterB:
    compute: 50

$ enos deploy -f advanced.yml
```

```
$ cat ./network-topo.yml
resources:
  grp1:
    clusterA:
      control: 1
      network: 1
      nova-conductor: 5
  grp2:
    clusterB:
      compute: 50

network_constraints:
  - src: grp1
    dst: grp2
    delay: 100ms
    rate: 10Gbit
    loss: 0%
    symetric: true

$ enos deploy -f network-topo.yml
```

1. EnOS deploy: Under the hood

```
resources:
  grp1:
    clusterA:
      control: 1
      network: 1
  grp2:
    clusterB:
      compute: 50

network_constraints:
  delay: 100ms
  rate: 10Gbit
  loss: 0%
```

```
$ enos deploy
```

1. Provider gets 2 nodes on `clusterA`, 50 nodes on `clusterB` and returns node's IP addresses
2. EnOS provisions nodes with Docker daemon (Kolla dependencies)
3. EnOS installs OpenStack using Kolla
4. EnOS sets up bare necessities (flavors, cirros image, router, ...)
5. EnOS applies network constraints between `grp1` and `grp2` using tc

- Provider to get testbed resources
  - Resources: anything running a Docker daemon and EnOS can SSH to + some IPs
  - Existing providers: Vagrant (VBox/Libvirt), Grid’5000, Chameleon, OpenStack, Static
  - ~500 LoC each
- Kolla to deploy OpenStack over testbed resources
- TC to apply network constraints

2. EnOS bench

- Benchmarks description

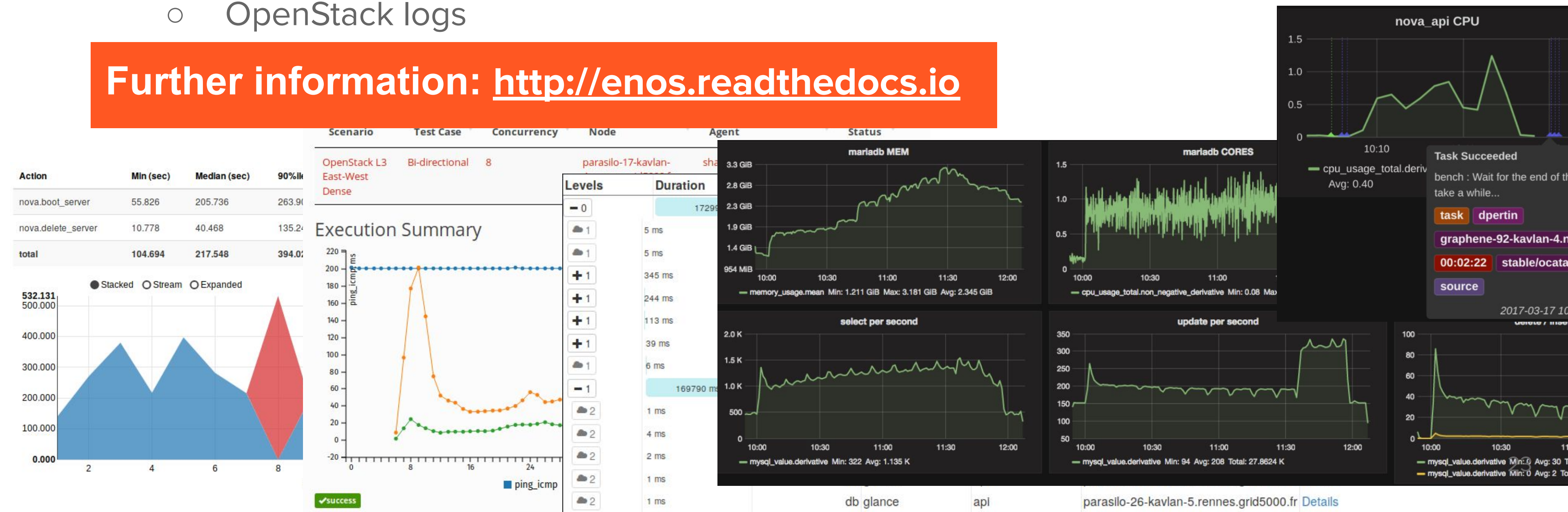
```
$ cat ./run.yml

rally:
  args:
    concurrency: 5
    times: 100
  scenarios:
    - name: boot and list servers
      file: nova-boot-list-cc.yml
      osprofiler: true
    - …
shaker: ...

$ enos bench --workload=run.yml
```

- Under the hood
  - Rally: control plane benchmark
  - Shaker: data plane benchmark
  - OSProfiler: code profiling
  - Monitoring stack: cAdvisor/Collectd to collect CPU/RAM/Network consumption per service/node/cluster

3. EnOS backup

- `enos backup` produces a tarball with:
  - Rally/Shaker reports
  - OSProfiler traces
  - InfluxDB database with cAdvisor/Collectd measures
  - OpenStack logs

Further information: http://enos.readthedocs.io



Action	Min (sec)	Median (sec)	90%ile
nova.boot_server	55.826	205.736	263.9
nova.delete_server	10.778	40.468	135.2
total	**104.694**	**217.548**	**394.0**

Scenario	Test Case	Concurrency	Node	Agent	Status
OpenStack L3 East-West Dense	Bi-directional	8	parasilo-17-kavlan-	sha	

Execution Summary

Task Succeeded

bench : Wait for the end of th take a while...

task dpertin

graphene-92-kavlan-4.n

00:02:22 stable/ocata

source

2017-03-17 10

4. Overview of the jobs done with EnOS

“Chasing 1000 nodes scale”

Presented at the OpenStack Summit in Barcelona (Newton, October 2016)

Goals: Find out how OpenStack behaves when deploying 1000 compute nodes

- Analysis of the control plane at scale
- Identify potential bottlenecks
- Identify:
  - Key service settings (e.g. number of service workers)
  - The influence of services topologies (e.g. service scalability)

“Chasing 1000 nodes scale”

Results:

- Some key services settings were identified
  - e.g. increase nova-conductor and neutron-server workers, max connections for database
- Identified some architectural requirements
  - e.g. one nova-scheduler for 100 compute nodes

How EnOS nailed it:

⇒ Leverage EnOS control-plane analysis to find out bottlenecks and impact of settings

⇒ Leverage EnOS flexibility to find the correct topology

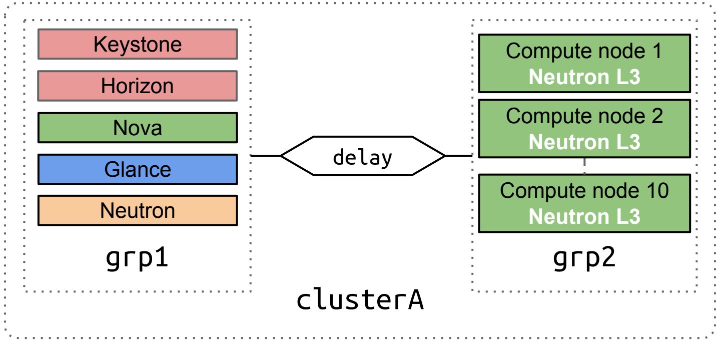
“Evaluating OpenStack WAN-wide”

Presented at the OpenStack Summit in Boston (Ocata, April 2017)

Goals:

- Investigate the latency impact on both control plane and data plane
- Validate results on two reconfigurable testbeds for experimentations:

Grid’5000 Chameleon



- 100+ bare-metal nodes

“Evaluating OpenStack WAN-wide”

Results:

- Control plane: Latency impacts completion times
- Data plane: Latency highly impact inter-VM response time
  - Because VMs have to fetch routing information at Neutron server
  - Enabling Neutron Distributed Virtual Routing (DVR) feature solves it (one line to add in EnOS config file)

How EnOS nailed it:

⇒ Leverage EnOS control plane (rally), data plane (shaker) and profiling (osprofiler) benchmark tools

⇒ Leverage EnOS providers to validate the behavior of 250 benchmarks on two different testbeds

⇒ Leverage EnOS network constraints to simulate WAN-wide infrastructure

FBK CREATE-NET (Trento, Italy)

- FEMDC active members
- OS and K8S for workload migration for Fog/Edge infra
- EnOS contributors

Fed4Fire+

- European project: federation of testbeds for research experimentations
- Benchmark comparison between OpenNebula and OpenStack
- EnOS contributors

5. Future works and conclusions

Future works

Evaluation of critical OpenStack services in massively distributed context:

1. Message bus service (RabbitMQ, ZeroMQ, QPID):

- Ongoing work: Orange, RedHat and Inria
- Massively distributed RPC test plans available from Performance WG

2. Database alternatives (MariaDB, newSQL):

- Ongoing work: Cockroach Labs (CockroachDB) and Inria
- Proof of concept with Keystone for now

⇒ Results expected to be presented for Vancouver Summit

Take-away message

Experimental environments for conducting OpenStack performance analyses:

- **Automate** each step of the experimentation workflow:
  - set experimental environment
  - deploy highly customizable OpenStack configuration
  - run benchmarks
  - collect generated data for real-time and post-mortem analysis
- **Software-defined approach:** describe everything description files
  - topology
  - OpenStack tuning
  - benchmark scenarios
- **Reproducibility:** help to validate your evaluations
  - many providers available

Take-away message (cont'd)

EnOS has been and will be used for conducting rigorous experiments:

- Large-scale OpenStack
- WAN-wide OpenStack
- Critical services in massively distributed infrastructures (message bus, database)

You can easily reproduce and tune them on your own infrastructure:

- Use existing providers (Vagrant, OpenStack, ...)
    - or create your own (~500 LoC)
- OpenStack tuning (reproduce on new releases, new service features)

Reproducible performance evaluations of different OpenStack deployments with EnOS

Fog/Edge/Massively Distributed Clouds Working Group

Beyond the Clouds - The Discovery initiative

Ronan-Alexandre Cherrueau, Dimitri Pertin, Anthony Simonet, Matthieu Simonin, Adrien Lebre

Inria
INVENTORS FOR THE DIGITAL WORLD