Toward Fog, Edge and NFV Deployments Evaluating OpenStack WANwide

Fog/Edge/Massively Distributed Clouds Working Group

Beyond the clouds - The Discovery initiative

Inria INVENTORS FOR THE DIGITAL WORLD

THE UNIVERSITY OF CHICAGO

Who are We?

Adrien Lebre

Fog/Edge/Massively Distributed WG Chair

https://wiki.openstack.org/wiki/Fog_Edge_Massively_Distributed_Clouds

Discovery Initiative Chair

http://beyondtheclouds.github.io

Ronan-Alexandre Cherrueau

Fog/Edge/Massively Distributed WG and Performance team Contributor

Discovery Initiative Researcher Engineer

EnOS main developer

http://enos.readthedocs.io

Pierre Riteau

Scientific WG contributor

Blazar project Core reviewer

Chameleon Lead DevOps Engineer

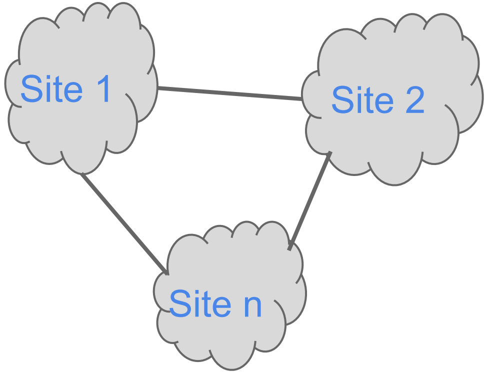
University of Chicago

https://www.chameleoncloud.org

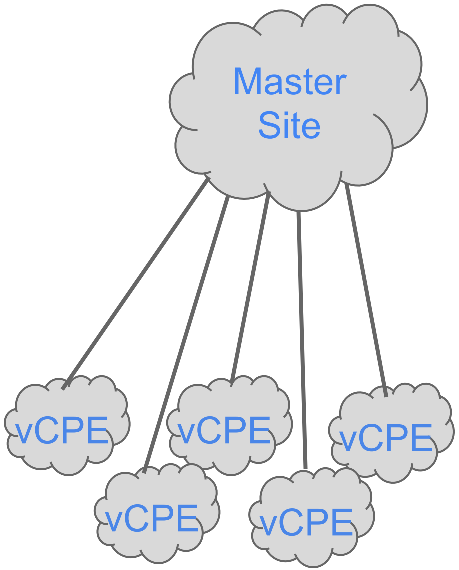
Agenda

1. OpenStack WANWide, Why ?
2. EnOS: Experimental Environment for OpenStack
3. Evaluation of OpenStack WANWide (using EnOS)

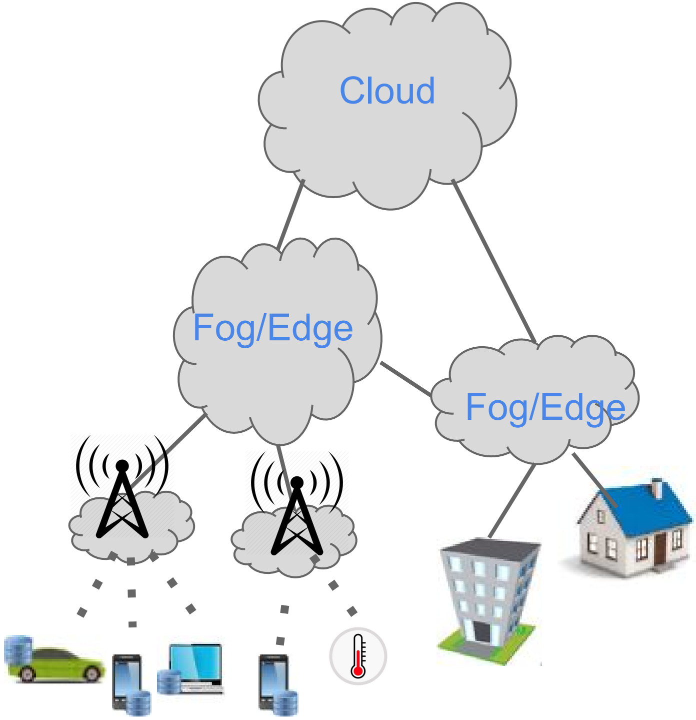
OpenStack WANWide



Academic cloud federation
(NeCTAR /EGI Federated Cloud)



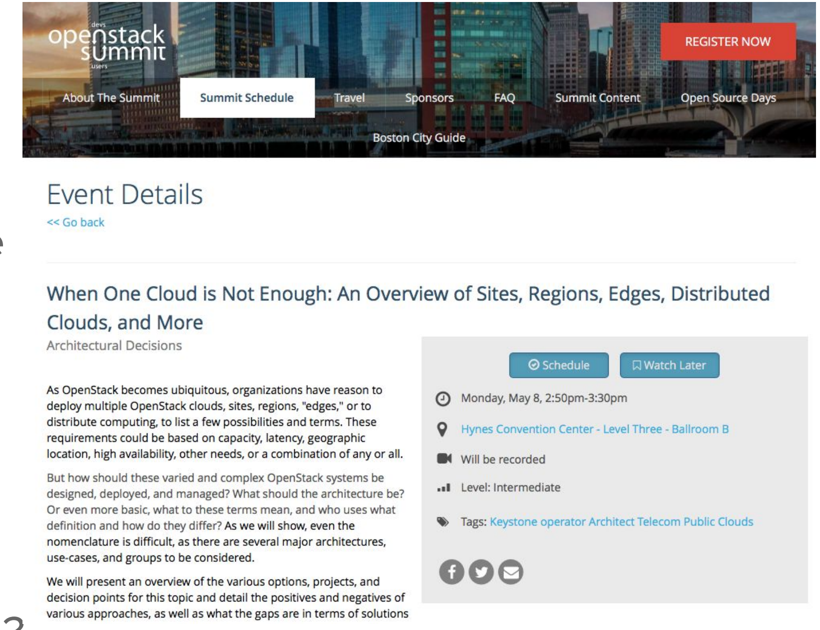
NFV/SDN
(large scale but provisioning frequency is rather low)



Fog/Edge Computing
(Latency aware / smart* / IoT Apps)

WAN Link
Wireless Link

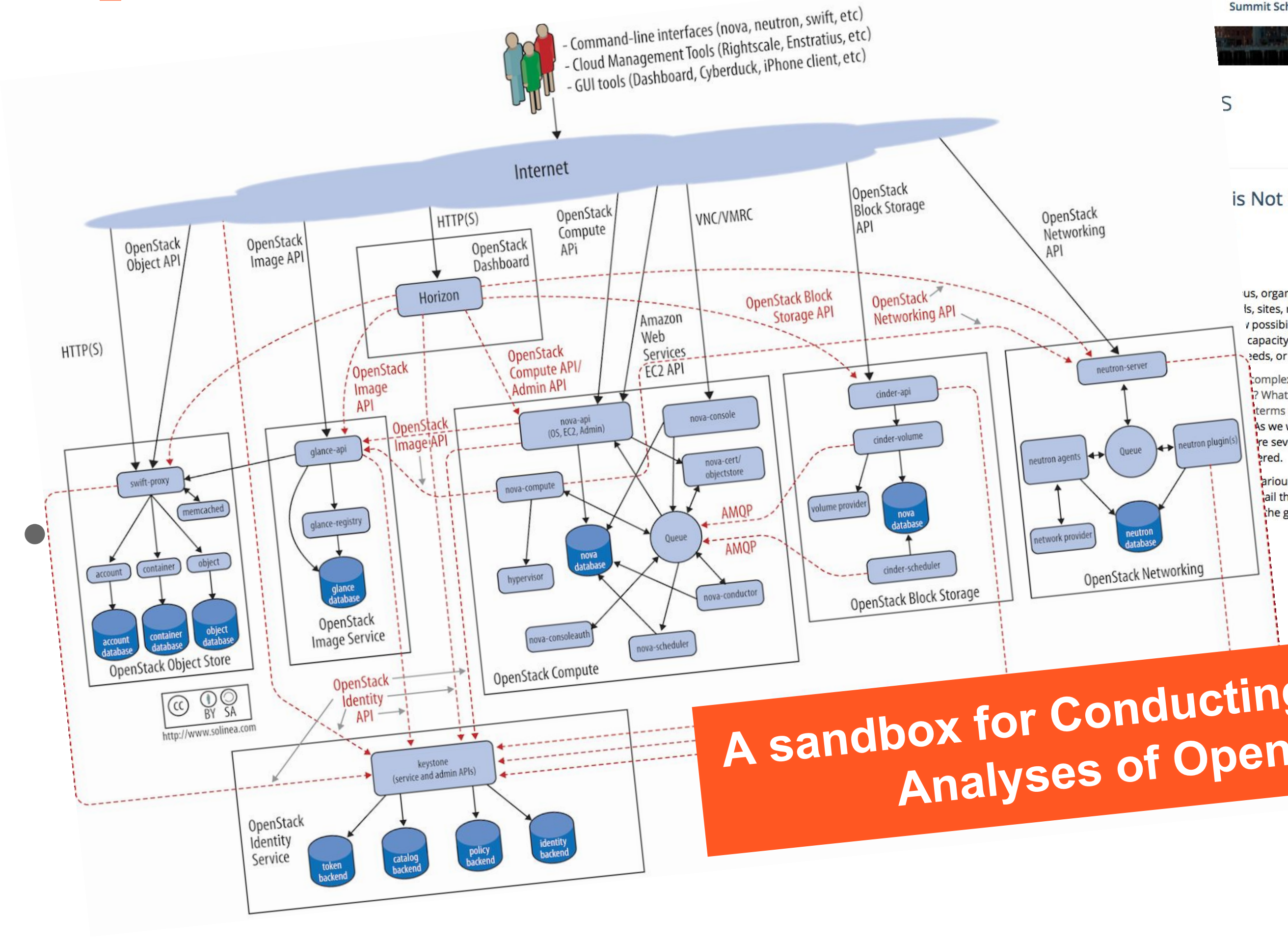
OpenStack WANWide (cont)

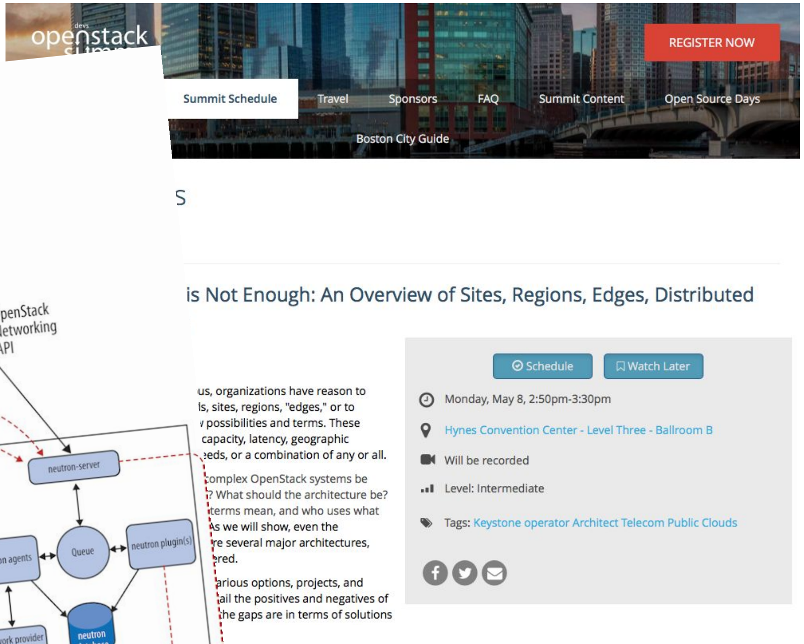


- Several deployment possibilities
  - control services deployed at one site compute nodes remotely
  - Segregation technics
    - cells (nova related)
    - regions (shared keystone)
  - Federated/brokering approaches

- Which one is the most interesting/appropriate?
  - No real functional/performance evaluations
    - Latency/throughput impact?
    - Message characterization: distinction between LAN and WAN traffic?
    - Changes between OpenStack Releases
    - Deployment complexity

OpenStack WANWide (cont)





A sandbox for Conducting Performance Analyses of OpenStack?

EnOS: Experimental Env. for OpenStack

- Motivation: Conducting performance analysis
  - **In a scientific and reproducible manner (automation)**
  - At small and large-scale
    Under different network topologies (traffic shaping)
  - Between different releases
  - With any kind of benchmarks

- Built on **Kolla** and leverage OSProfiler, Rally and Shaker

- Workflow
  - `$ enos deploy`
  - `$ enos bench`
  - `$ enos backup`

EnOS deploy – Resource/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
```

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
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 Provider: Vagrant (VBox/Libvirt), Grid’5000, Chameleon, OpenStack
  - ~500 LoC each
- Kolla to deploy OpenStack over testbed resources
- TC to apply network constraints

EnOS bench

- Benchmarks description

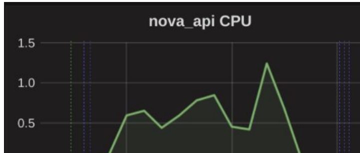
```
$ cat ./run.yml

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
```

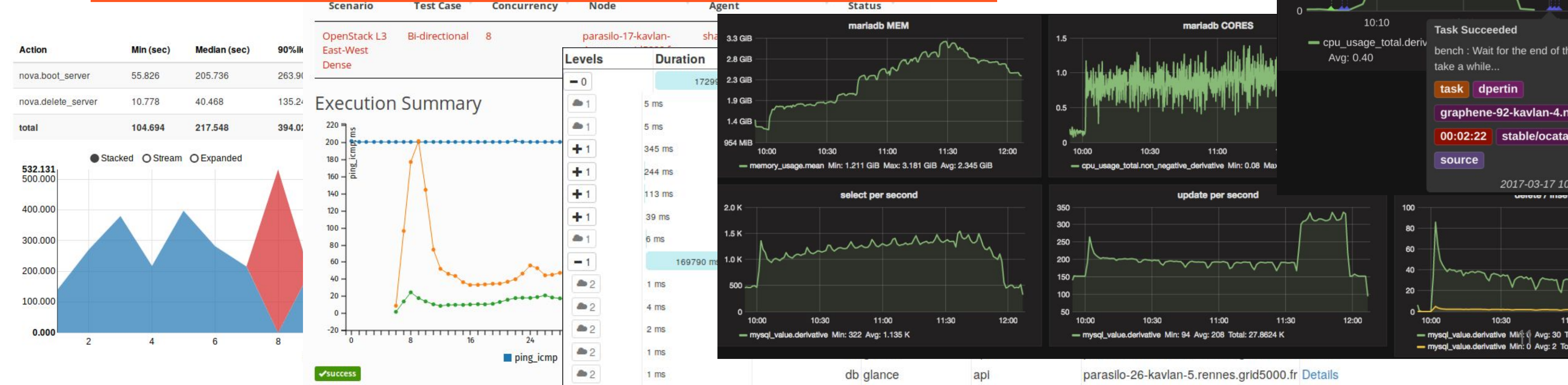
- Under the hood
  - Rally: control plane benchmark
  - Shaker: data plane benchmark
  - OSProfiler: code profiling
  - Monitoring stack: cAdvisor/Collectd to collect CPU/RAM/Network consumption per service/node/cluster

EnOS backup

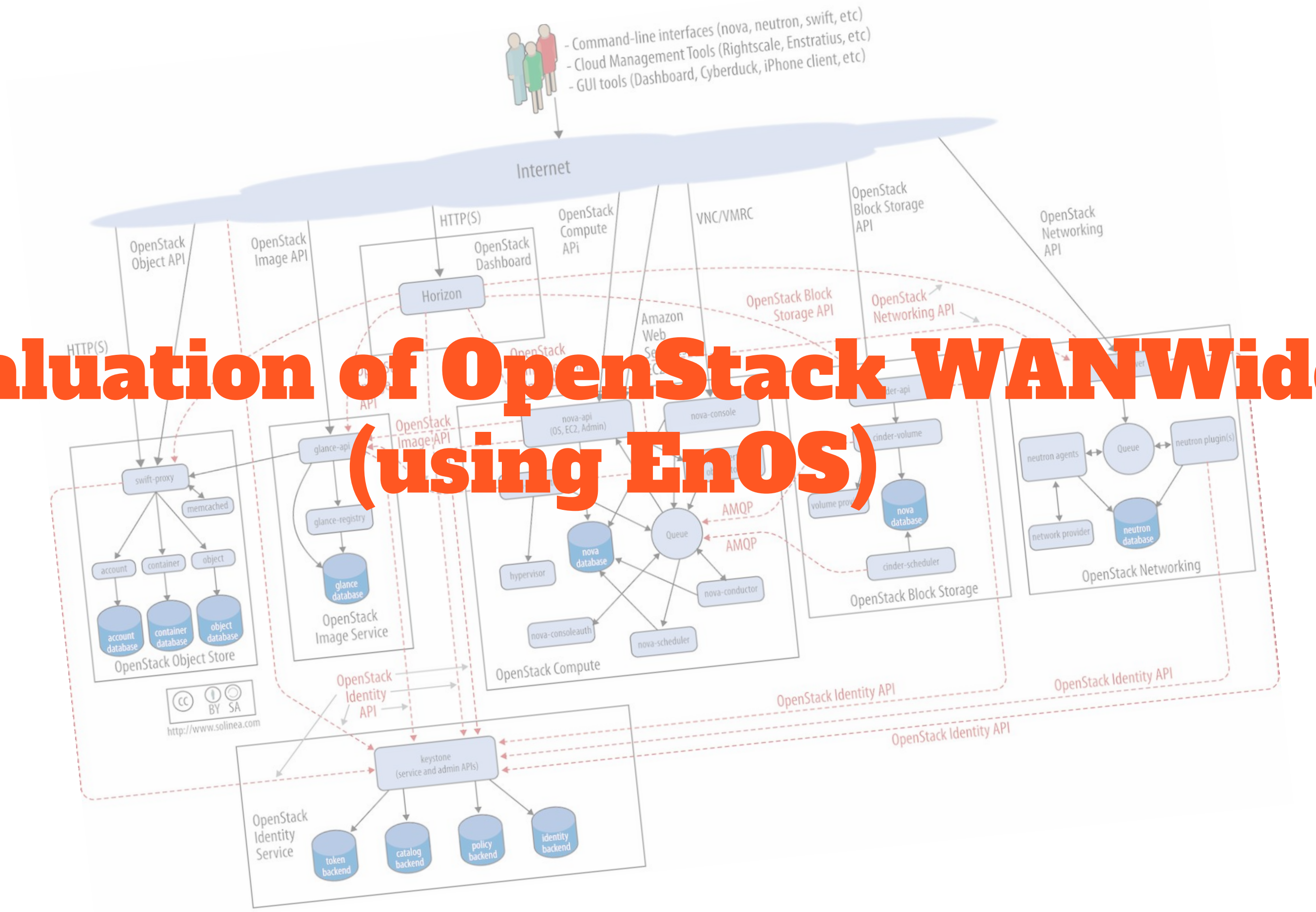
- `enos backup` produces a tarball with
  - Rally/Shaker reports
  - OSProfiler traces
  - InfluxDB database with cAdvisor/Collectd measures
  - OpenStack logs

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

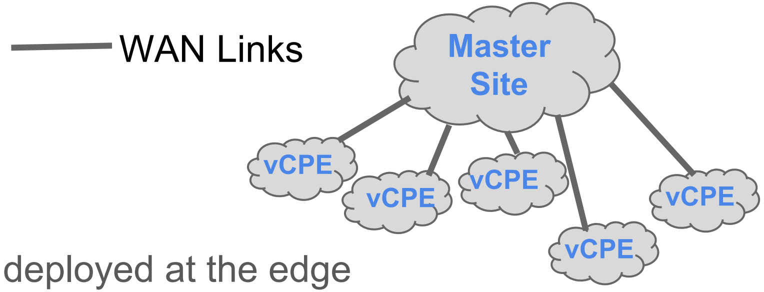




Evaluation of OpenStack WANWide (using EnOS)

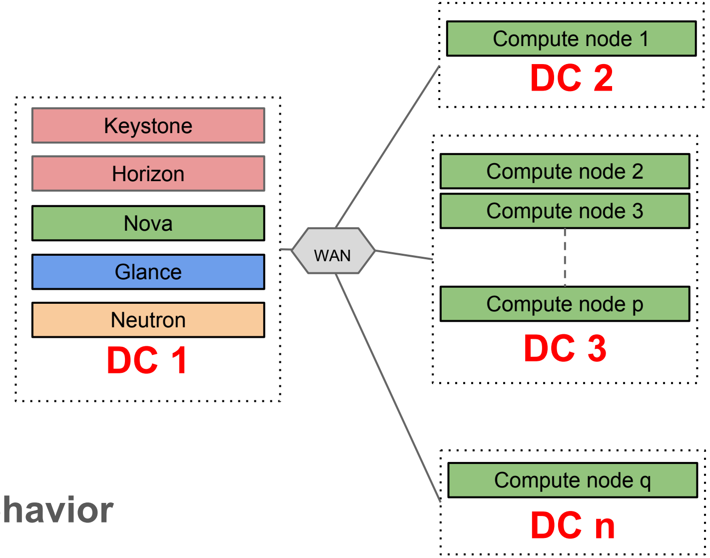


OpenStack WANWide



- A Single OpenStack to operate remote compute resources deployed at the edge
  - All control services are deployed into the master.
  - The RabbitMQ bus is deployed across all locations (i.e., through each server composing the infrastructure)

- Pros: simple
- Cons:
  - security management for RPC message and port, Single Point of Failure…
  - Scalability (not addressed in this presentation, see "Chasing 1000 Nodes Scale", Barcelona Summit 2016)
  - **Network latency/throughput impacts on functional behavior and performance degradations**.



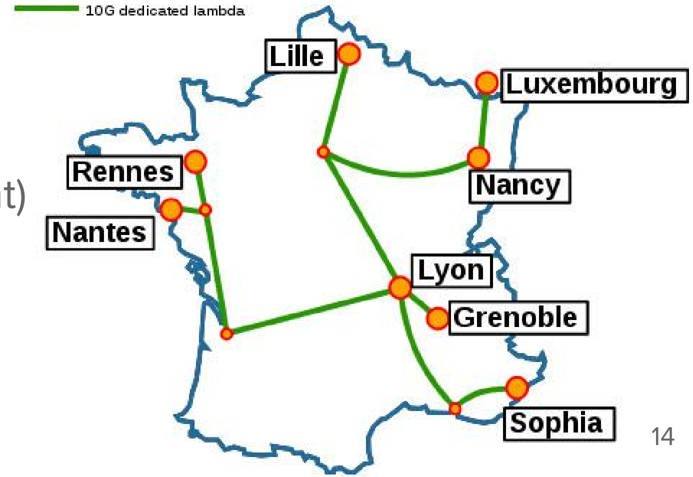
TestBed – Grid'5000

Grid'5000

- One of the world-leading **testbeds for Distributed Computing**
  - 8 sites, 30 clusters, 840 nodes, 8490 cores
  - Dedicated 10Gbps backbone network
  - 550 users and 100 publications per year

- **Used by CS researchers in HPC / Clouds / Big Data / Networking**

- Design goal
  - Support **high-quality, reproducible experiments**
    (i.e. in a fully controllable and observable environment)



TestBed – Chameleon

- NSF-funded testbed for computer science experimentation
- Built with OpenStack software (and some from Grid'5000)
- **Reconfigurable:** node reservation (Blazar), bare-metal deployment (Ironic)
- **Large-scale:** 504 compute nodes & 48 storage nodes, 3.6 PB global storage
- **Heterogeneous hardware:** Infiniband, NVMe, SSDs, GPUs, FPGAs, ARM & Atom
- Hardware distributed over two sites: TACC (Austin, TX) and UC (Chicago, IL)
- Serving ~1,400 users in 200+ projects

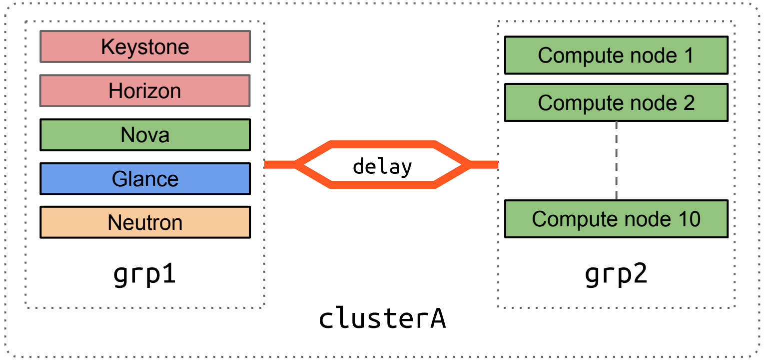
Just after this talk in MR 208: "We Need Clouds to Build Clouds: Developing an Open Cloud Testbed Using OpenStack"

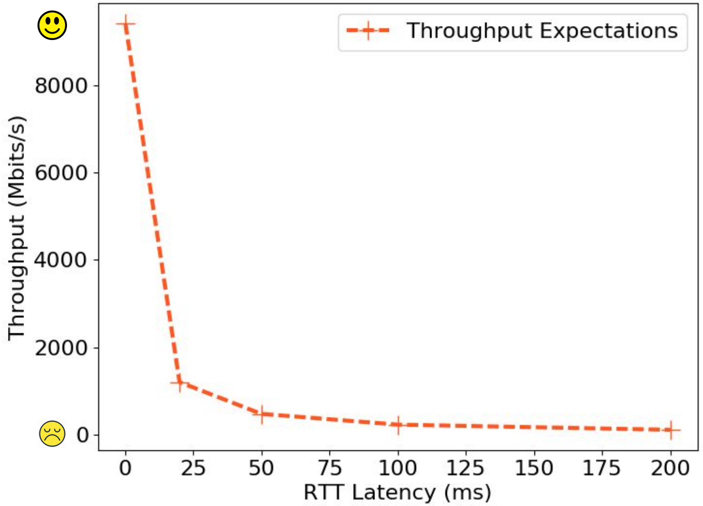
TestBeds Grid'5000 +

- Experiments runs independently on both testbeds in a fully automatized manner (Software defined experiments leveraging EnOS)
- 250 benchmarks (approx. 100 running hours) on each testbed.
- Results lead to the same conclusion whatever the testbed (collected performance are almost identical).
- Experimental setup: https://github.com/BeyondTheClouds/enos-scenarios/
- Results: http://enos.irisa.fr/html/

Latency Impact (Experiment #1)

```
$ cat ./wan-exp1.yml

resources:
  grp1:
    clusterA:
      control: 1
  grp2:
    clusterA:
      compute: 10

network_constraints:
  delay: 0ms # 10ms, 25ms, 50ms, 100ms
  loss: 0%
  rate: 10Gbit
  src: grp1
  dest: grp2
  symetric: true

$ enos deploy -f wan-exp1.yml
```

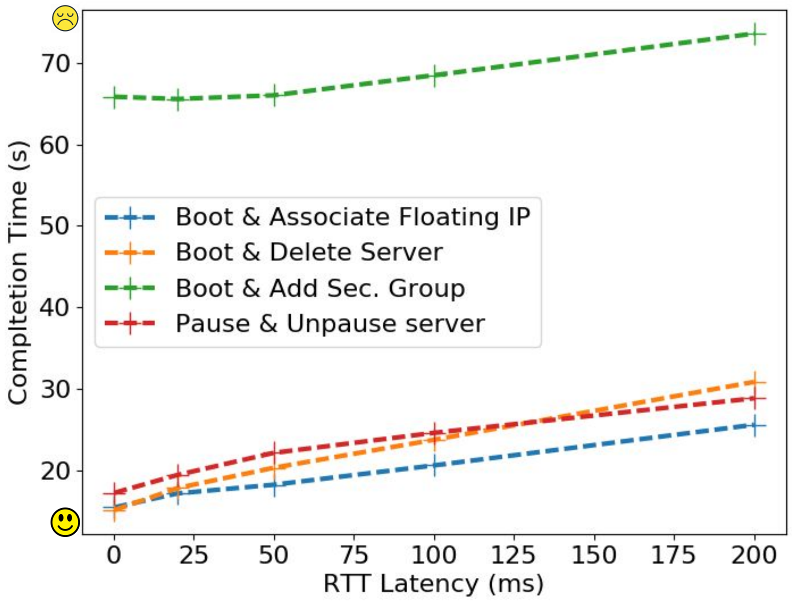




Find the scenario at https://github.com/BeyondTheClouds/enos-scenarios/tree/master/wan/cpt10

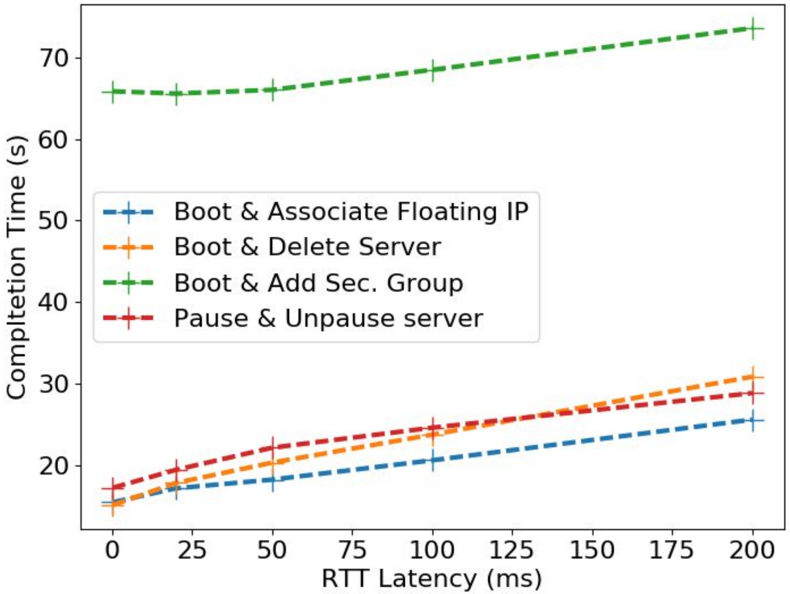
Latency Impact – Control Plane (Rally Metrics)

```
$ cat ./run.yml

rally:
  args:
    concurrency: 1
    times: 20
  scenarios:
    - file: nova-boot-and-associate-floating-ip.yml
    - file: nova-boot-and-delete.yml
    - file: nova-boot-and-add-secgroup.yml
    - file: nova-pause-and-unpause.yml
shaker: ...

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

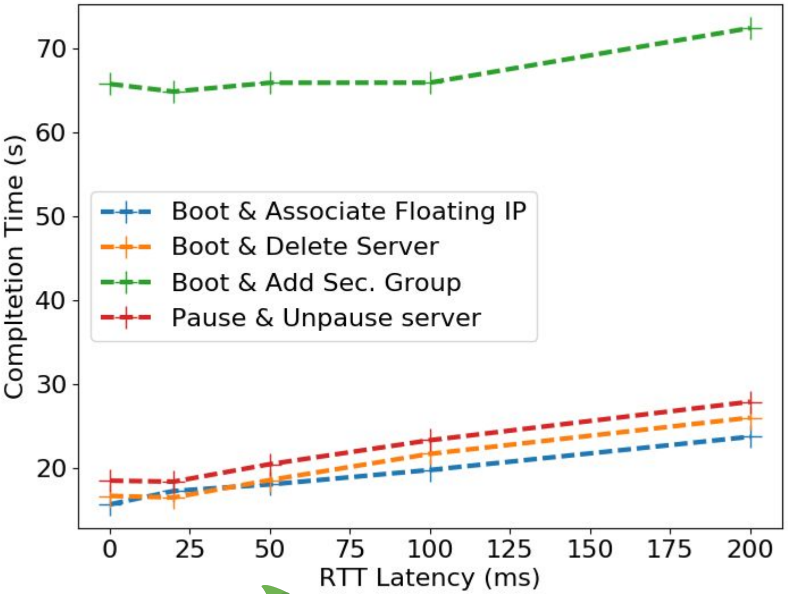


Completion time increases with latency (factor 2 between 0 and 200ms)

Latency Impact – Control Plane (Rally Metrics)



Grid'5000



Chameleon

Trends are similar (which is what we expected !)

Latency Impact – Control Plane (OSProfiler Metrics)

- OSProfiler:
  - OpenStack cross-project **profiling** library
  - Provides execution times and arguments for REST/RPC/Python/DB calls
- OSProfiler outputs:
  - JSON file + HTML view
  - View rather heavy: 10.5K calls for "`nova-boot-and-add-secgroup`" rally scenario, resulting in 47MBytes HTML files
- New PoC project **`osp-utils`**:
  - Provides operators to query an OSProfiler trace: filter, folder, … to reduce trace size
  - Produces sequence diagram to show interactions between services
  - https://github.com/beyondtheclouds/osp-utils

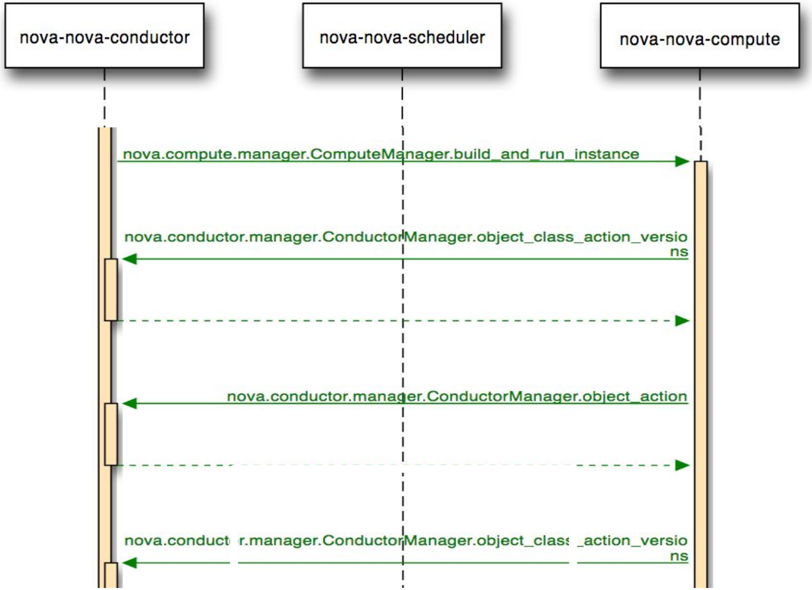
Latency Impact – Control Plane (OSProfiler Metrics)

HTML View

− 4	51 ms	**rpc**	nova	nova-conductor
5	2 ms	db	nova	nova-conductor
5	2 ms	db	nova	nova-conductor
− 5	5 ms	**rpc**	nova	nova-compute
+ 6	52 ms	**rpc**	nova	nova-conductor
+ 6	196 ms	**rpc**	nova	nova-conductor
+ 6	14 ms	**rpc**	nova	nova-conductor
+ 6	208 ms	**rpc**	nova	nova-conductor
+ 6	20 ms	**rpc**	nova	nova-conductor
+ 6	55 ms	**rpc**	nova	nova-conductor
+ 6	145 ms	**rpc**	nova	nova-conductor
+ 6	46 ms	**rpc**	nova	nova-conductor
6	1 ms	neutron_api	nova	nova-compute

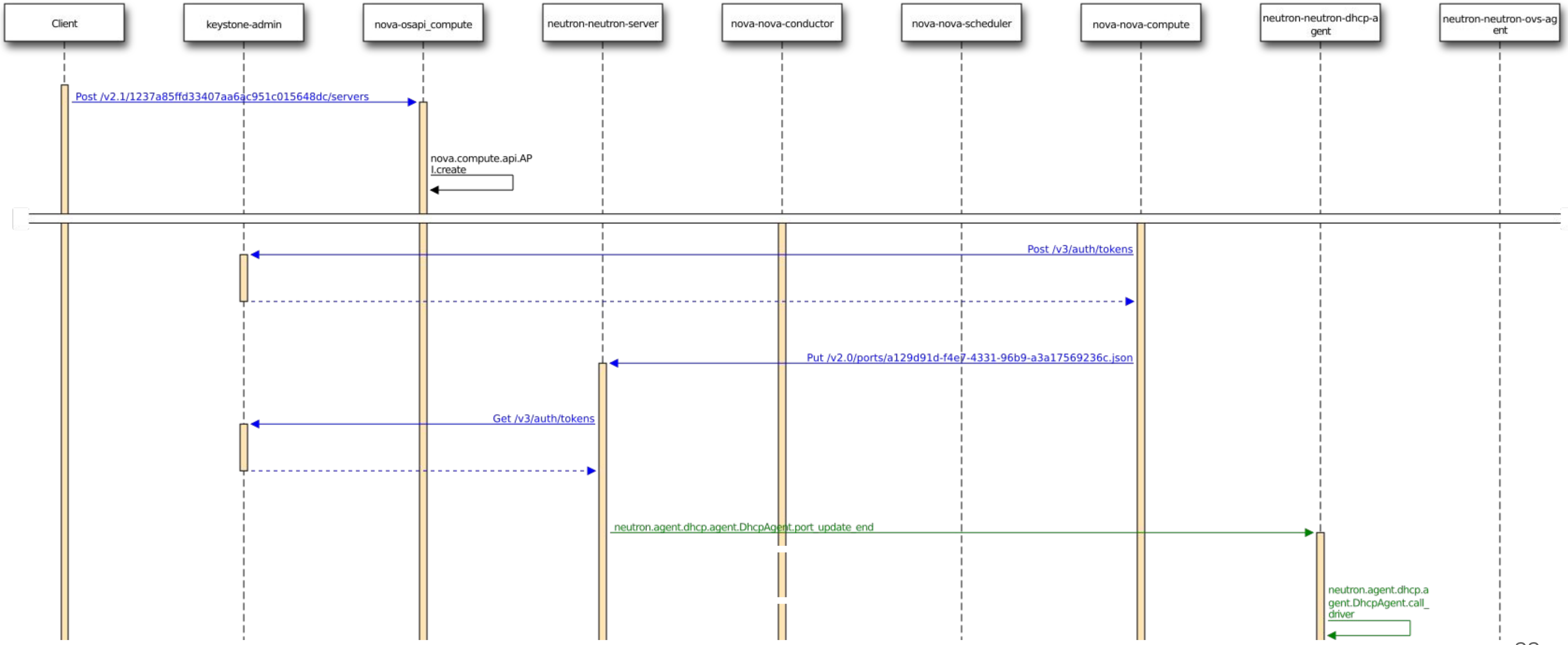
…

Sequence Diagram



…

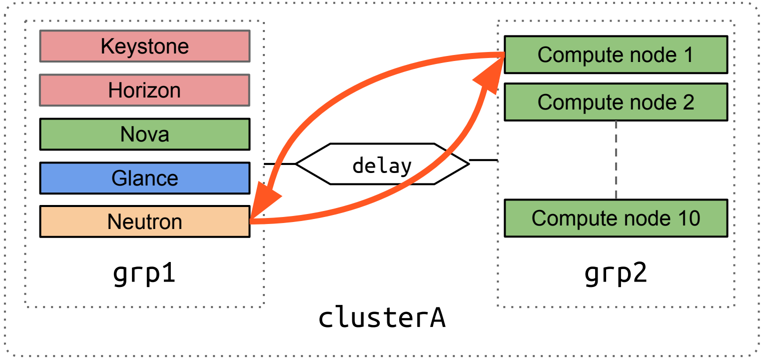
Latency Impact – Control Plane (OSProfiler Metrics)

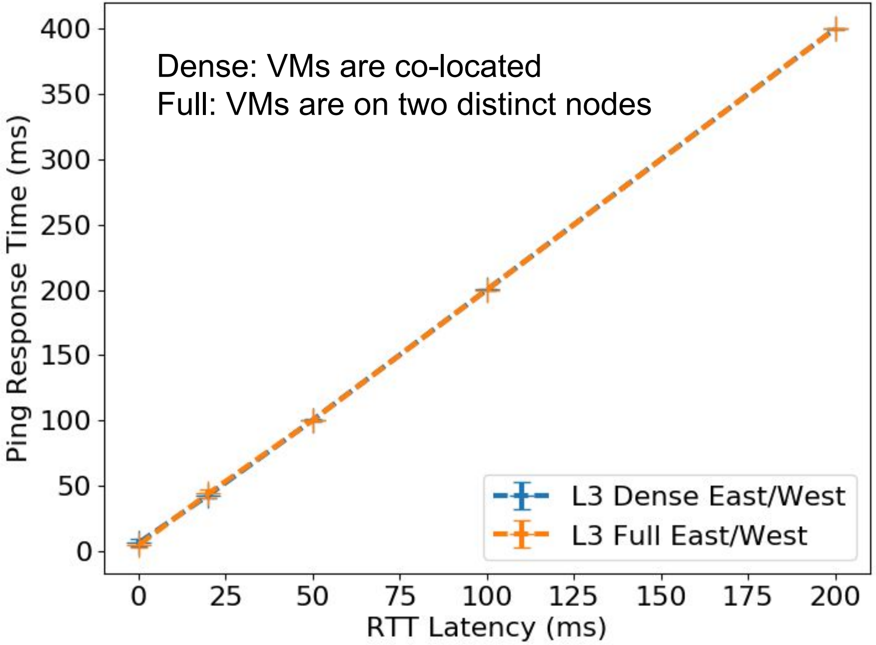


• • •

Latency Impact – Data Plane (Shaker Metrics)

```
$ cat ./run.yml

rally: ...
shaker:
  - file: openstack/dense_l3_east_west.yml
  - file: openstack/full_l3_east_west.yml

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

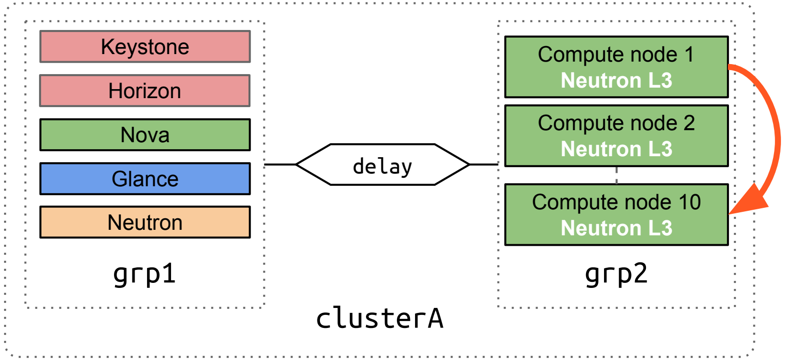




- Ping response time is twice the RTT (which corresponds to the normal workflow)

Latency Impact with DVR (Experiment #2)

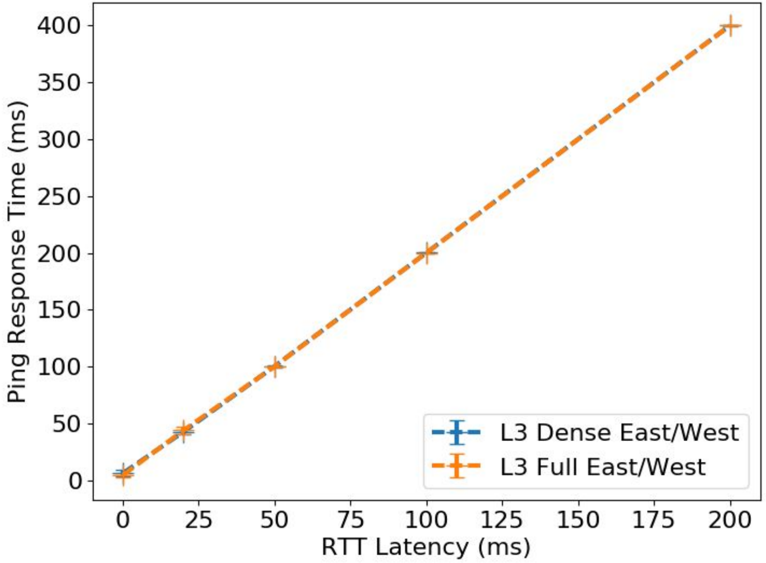
- You say DVR?
  - Distributed Virtual Routing
  - L3 forwarding/NAT distributed to the compute nodes



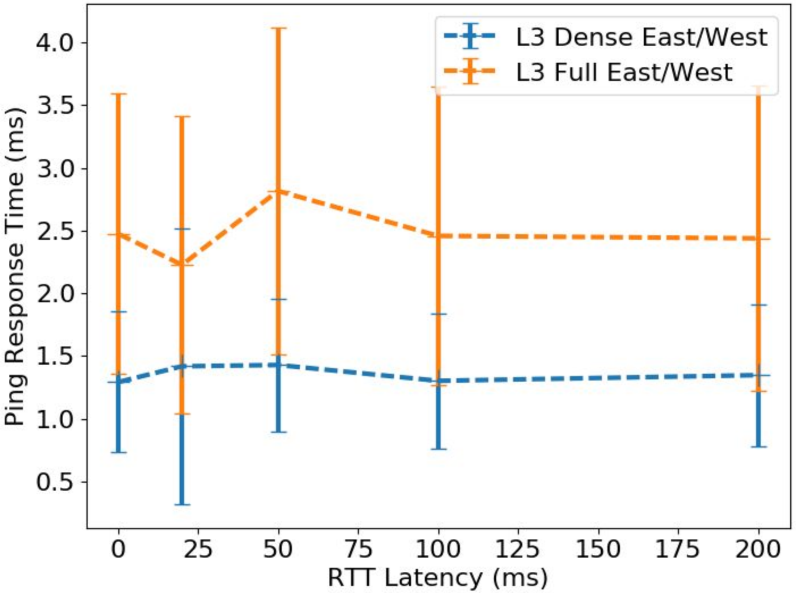

```
$ cat ./wan-exp2.yml

resources: ...

network_constraints: ...

kolla:
  enable_neutron_dvr: true

$ enos deploy -f wan-exp2.yml
```

Find the scenario at https://github.com/BeyondTheClouds/enos-scenarios/tree/master/wan/cpt10-dvr

Latency Impact with DVR – Data Plane



Without DVR (2*RTT)

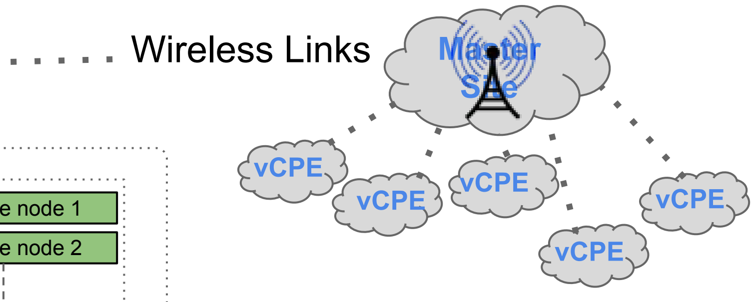


With DVR (LAN RTT)

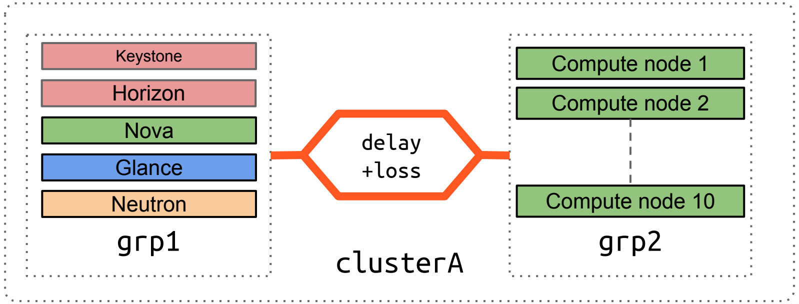
Critical change in WAN context

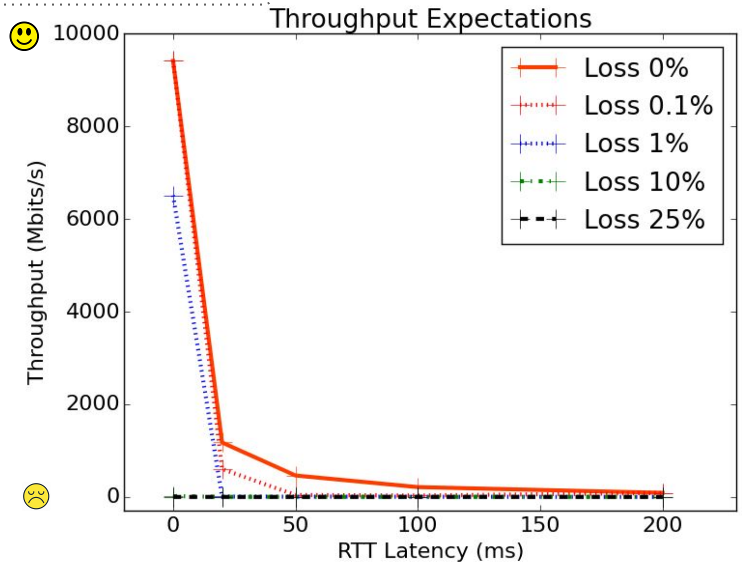
Find the scenario at https://github.com/BeyondTheClouds/enos-scenarios/tree/master/wan/cpt10-dvr

Loss Impact (Experiment #3)



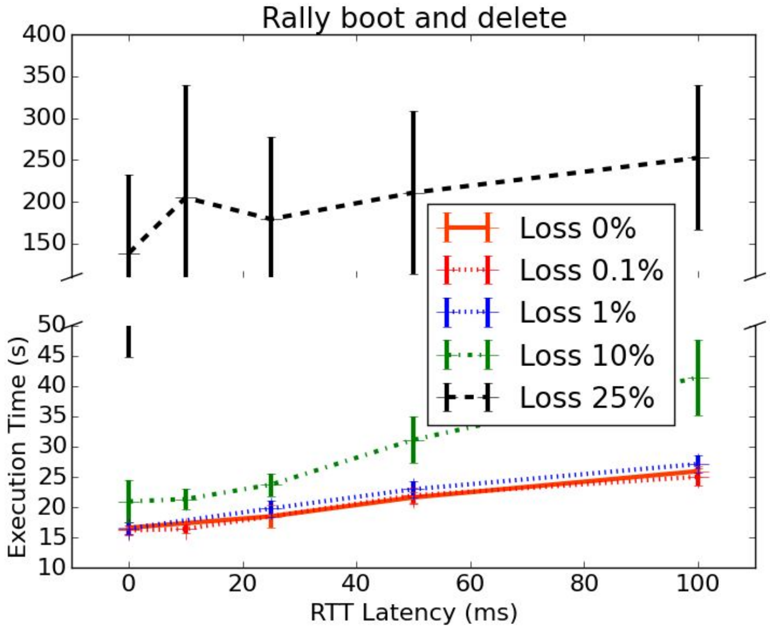

```
$ cat ./wan-exp1.yml

resources:
  grp1:
    clusterA:
      control: 1
  grp2:
    clusterA:
      compute: 10

network_constraints:
  delay: 0ms # 10ms, 25ms, 50ms, 100ms
  loss: 0% # 0.1%, 1%, 10%, 25%
  rate: 1Gbit
  src: grp1
  dest: grp2
  symetric: true

$ enos deploy -f wan-exp1.yml
```





Find the scenario at https://github.com/BeyondTheClouds/enos-scenarios/tree/master/wan/cpt10

Network Packet Loss (Rally Metrics)



- Possible tuning
  - TCP parameters (e.g buffers)
  - oslo.messaging configuration (e.g retries, timeouts)

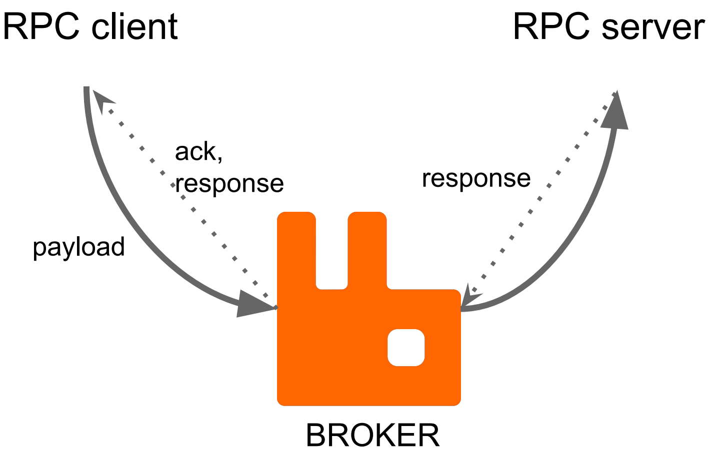
- Enos helps to find the right configuration/ implementation

Higher loss and latency : higher fluctuation and errors

Ongoing Actions: Diving Into The Gathered Results

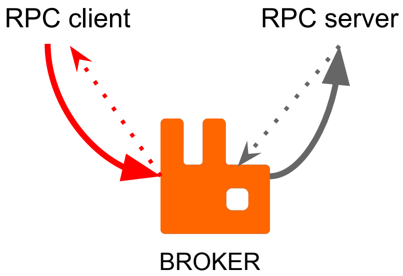
- REST API and RPC
  - inter-service : REST API calls (P2P like)
  - intra-service : RPC calls (RabbitMQ)
    - RPC.call
      (wait for the response, synchrone)
    - RPC.cast (fire and forget)
- RPCs require permanent connections to the broker
- **RPC cast is "half asynchronous"**
  It's more *"fire (wait) and forget"*

Typical flow for RPCs:

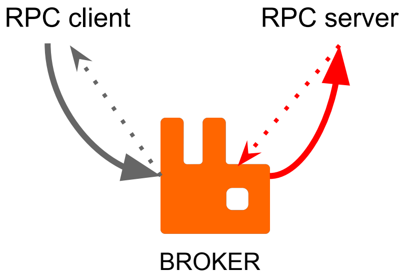


RPC / API calls

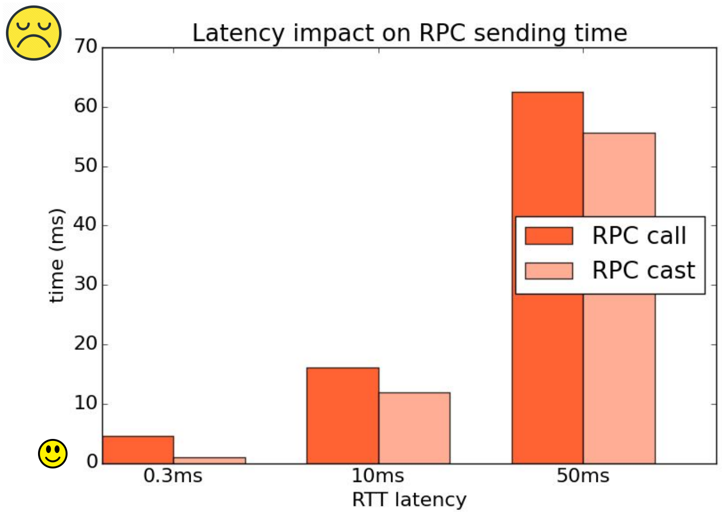
Latency between client and broker

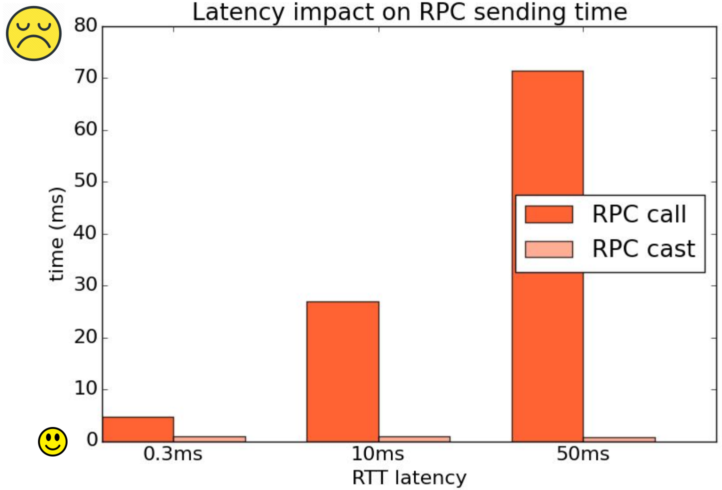


Latency between server and broker



Placement Challenge





Takeaway Message - EnOS

- Experimental **environment for conducting OpenStack Performance Analyses**
    - Consolidation of the initial proposal (‘’Chasing 1000 Nodes Scale’’, Barcelona Summit, Oct 2016, in collaboration with the Performance Team)
    - **Evaluations are conducted on real deployments** (not by leveraging DevStack)
    - OSProfiler: a powerful/mandatory tool to dive into details

- Automation is critical: conducting rigorous experiments take times
    - **EnOS helps you to automatize the process** (software defined experiments)

- **OpenScience**
    - **All scripts+results available** at http://enos.irisa.fr/html/
    - You can **redo everything on your own infrastructure** using available providers (Vagrant, Grid’5000, Chameleon, and any OpenStack compliant infrastructures) or by implementing yours (500 LoC max).

Takeaway Message - WANWide

- Control Plane:
  - Completion times increase but **OpenStack still behaves correctly**
  - **More complex scenarios** involving Glance, Cinder **should be investigated**

- Data Plane: **be aware of key components** and configured them correctly

- What did we miss in the experiment protocol
  - (network split brain, ...)? / Gather new tests (OPNFV)?
  - Please come and tell us how to improve it, adapting "enos bench" phase is straightforward

- **What's next**
  - **The F2F meeting session (5:20pm - Room MR 201)** 
  - Focus on **AMQP alternatives** (Apache Qpid Dispatch Router/ZeroMQ/...)
  - **Placement challenges of central components** such as Glance, Cinder.

Toward Fog, Edge and NFV Deployments Evaluating OpenStack WANwide

Thanks

Slides available by going on the F2F Etherpad WG
https://wiki.openstack.org/wiki/Fog_Edge_Massively_Distributed_Clouds

Timeline

4 min who we are

- 1 min each: Ronan/Adrien/Pierre
- 1 min Discovery initiative / Massively Distributed WG / Performance WG

5 min: OpenStack WANWide.

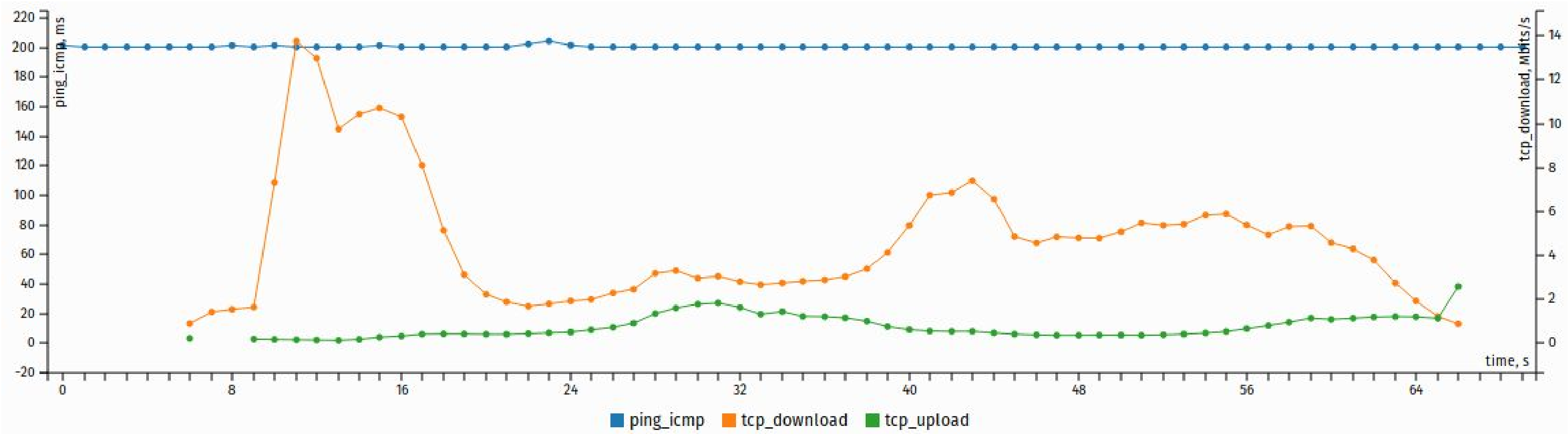
4 min testbeds (Adrien / Pierre)

8 min ENOS (Ronan)

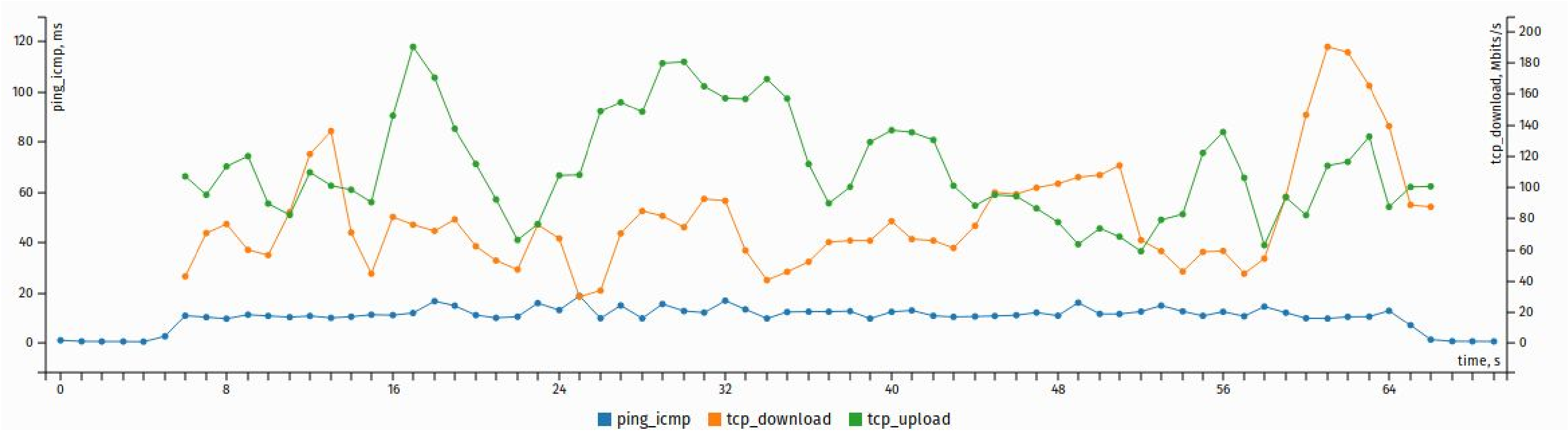
12 min exp (Ronan*2/Pierre the last)

2 min to conclude (Adrien)

Backup / junk slides



ping_icmp, ms
tcp_download, Mbits/s
time, s
ping_icmp
tcp_download
tcp_upload



ping_icmp, ms
tcp_download, Mbits/s
time, s
ping_icmp
tcp_download
tcp_upload

Latency Impact – Control Plane (OSProfiler Vision)

− 5	5 ms	build_and_run_instance (receiver)	**rpc** nova	nova-compute
+ 6	52 ms	update instance (receiver)	**rpc** nova	nova-conductor
+ 6	196 ms	update instance (receiver)	**rpc** nova	nova-conductor
+ 6	14 ms	update instance (receiver)	**rpc** nova	nova-conductor
+ 6	208 ms	update instance (receiver)	**rpc** nova	nova-conductor
+ 6	20 ms	update instance (receiver)	**rpc** nova	nova-conductor
+ 6	55 ms	update instance (receiver)	**rpc** nova	nova-conductor
+ 6	145 ms	update instance (receiver)	**rpc** nova	nova-conductor
+ 6	46 ms	update instance (receiver)	**rpc** nova	nova-conductor

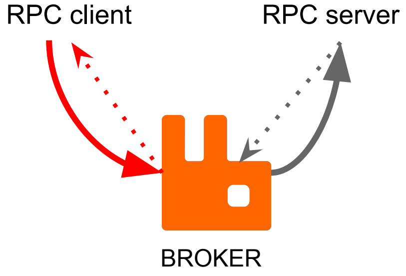
LAN latency

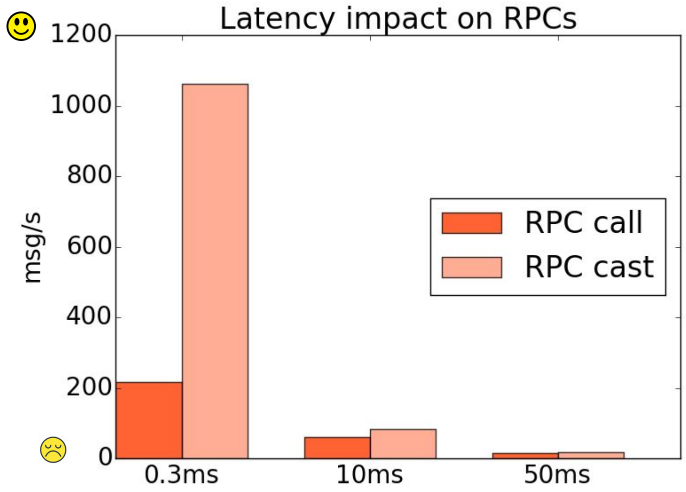
− 5	202 ms	**rpc** nova	nova-compute
+ 6	47 ms	**rpc** nova	nova-conductor
+ 6	168 ms	**rpc** nova	nova-conductor
+ 6	13 ms	**rpc** nova	nova-conductor
+ 6	170 ms	**rpc** nova	nova-conductor
+ 6	14 ms	**rpc** nova	nova-conductor
+ 6	54 ms	**rpc** nova	nova-conductor
+ 6	166 ms	**rpc** nova	nova-conductor
+ 6	63 ms	**rpc** nova	nova-conductor

200ms latency

RPC / API calls

Latency between client and broker





Latency between server and broker