Revising OpenStack to Operate Fog/Edge Computing Infrastructures

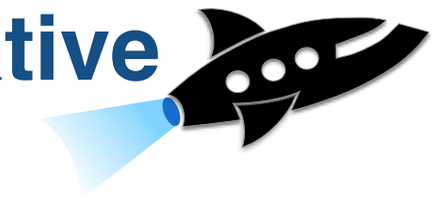
Frédéric Desprez, Adrien Lebre, Jonathan Pastor, Anthony Simonet

International Conference on Cloud Engineering
Vancouver, April 6th, 2017

Inria, IMT Atlantique, LS2N, France

Outline

- Fog/Edge Computing and the **Discovery Initiative**



- Contribution: ROME
- Evaluation
- Conclusion

Fog and Edge Computing

Current Cloud Computing Trend: Large Offshore DCs

To cope with the increasing demand while handling energy concerns but…

credits: datacentertalk.com - Microsoft DC, Quincy, WA state

Current Cloud Computing Trend: Large Offshore DCs

To cope with the increasing demand while handling energy concerns but…

credits: google map - Quincy

Current Cloud Computing Trend: Large Offshore DCs

To cope with the increasing demand while handling energy concerns but…

credits: coloandcloud.com

Reliability
Jurisdiction concerns
Latency (distance to DC)

Reliability
Jurisdiction concerns
Latency (distance to DC)

Image: Fettweis, G.; Alamouti, S., "5G: Personal Mobile Internet beyond What Cellular Did to Telephony," Communications Magazine, IEEE , vol. 52, no. 2, pp. 140-145, February 2014

2016 - xxxx
Big Data - Internet of Things/Everything

The Cloud is Not Enough: Saving IoT from the Cloud
Zhang et al. HotCloud Usenix 2015

Driving Cloudlet into OpenStack
(OpenStack summit Tokyo 2015)

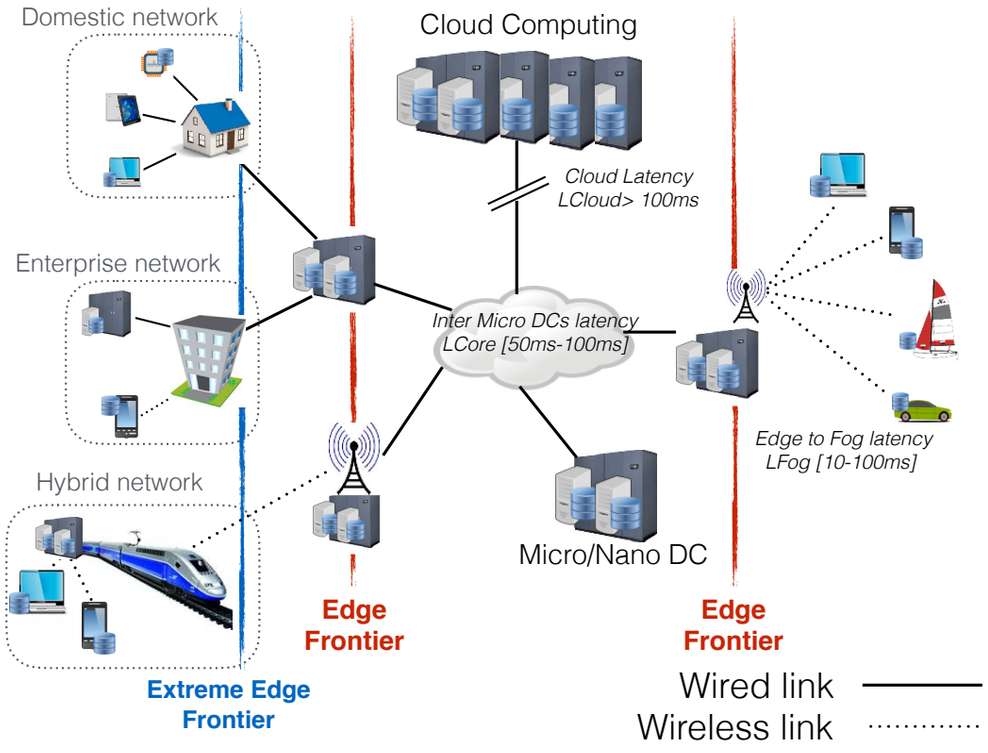
…

The Discovery Initiative

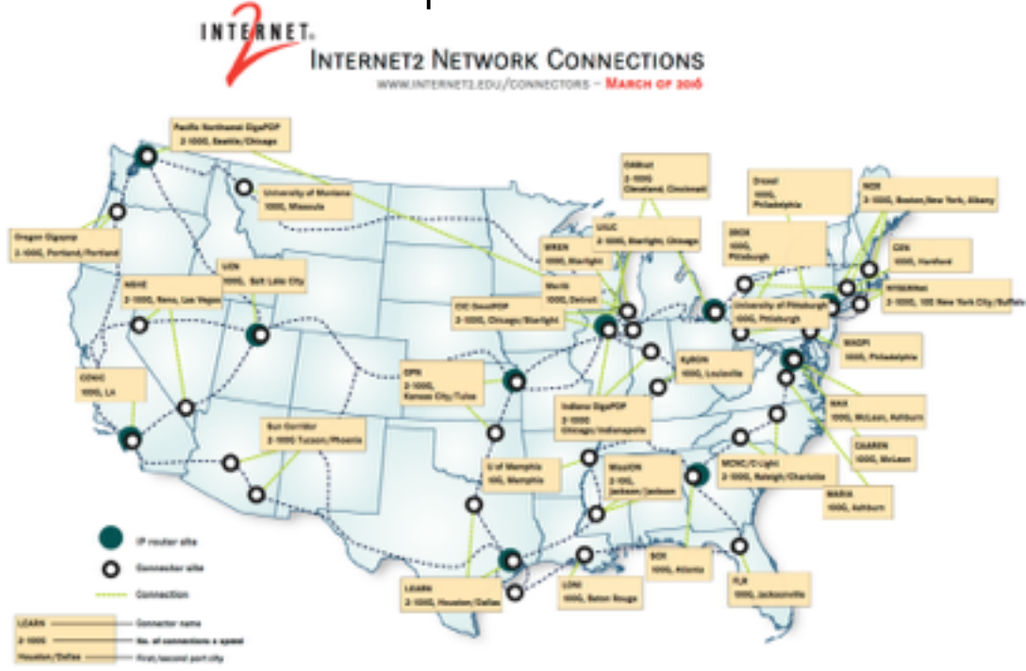
- Leverage network backbones

Extend any Point of Presence (PoP) of network backbones with servers (from network hubs up to major DSLAMs that are operated by telecom companies, network institutions…).

- Extend to the edge by including radio base stations



European NREN



USA NREN

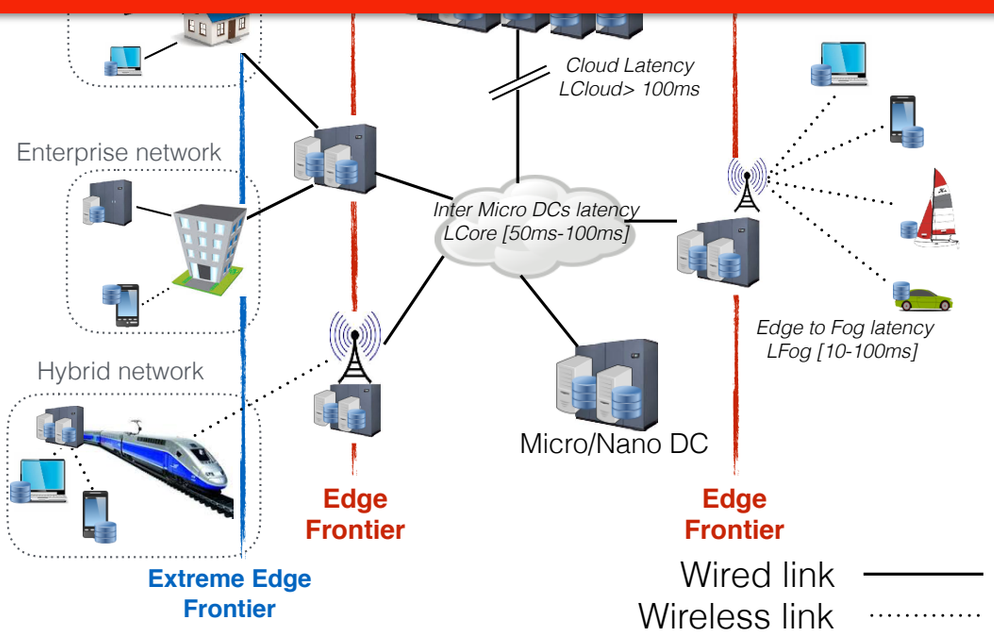
The Discovery Initiative

- Leverage network backbones

Extend any Point of Presence (PoP) of network backbones with servers (from network hubs up to major DSLAMs that are operated by telecom companies, network institutions…).

- Extend to the edge by including radio base stations

Discovery: how to operate such a massively distributed infrastructure

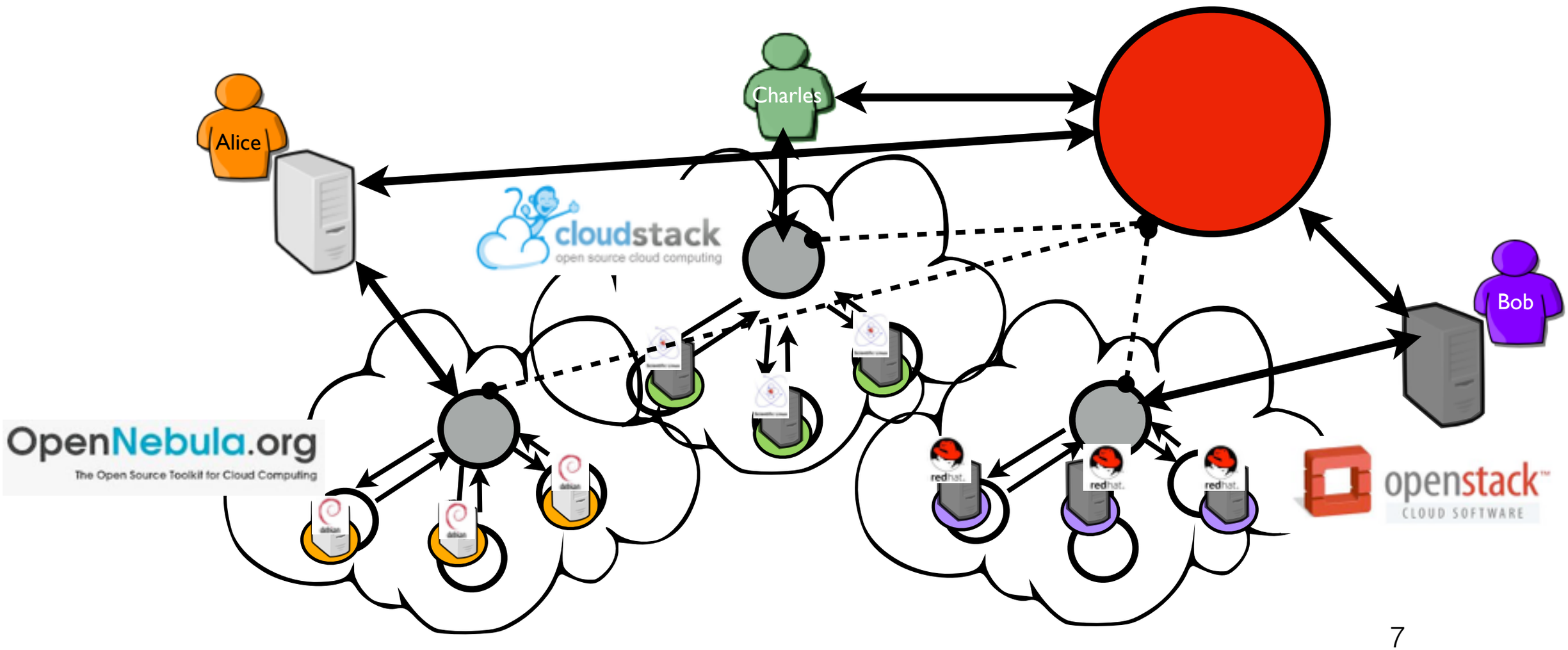


USA NREN

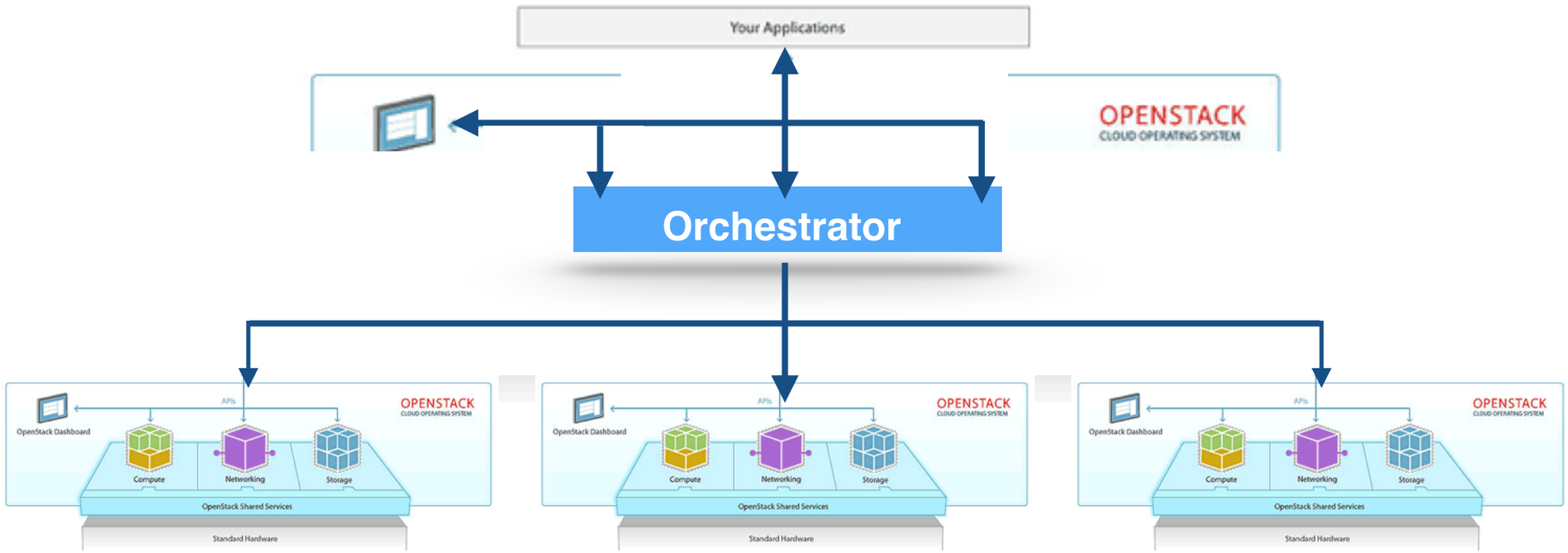
What About Brokering Approaches?

- Sporadic (hybrid computing/cloud bursting) almost ready for production
- While standards are coming (OCCI, ….), current brokers are rather limited

Advanced brokers must reimplement standard IaaS mechanisms while facing the API limitation



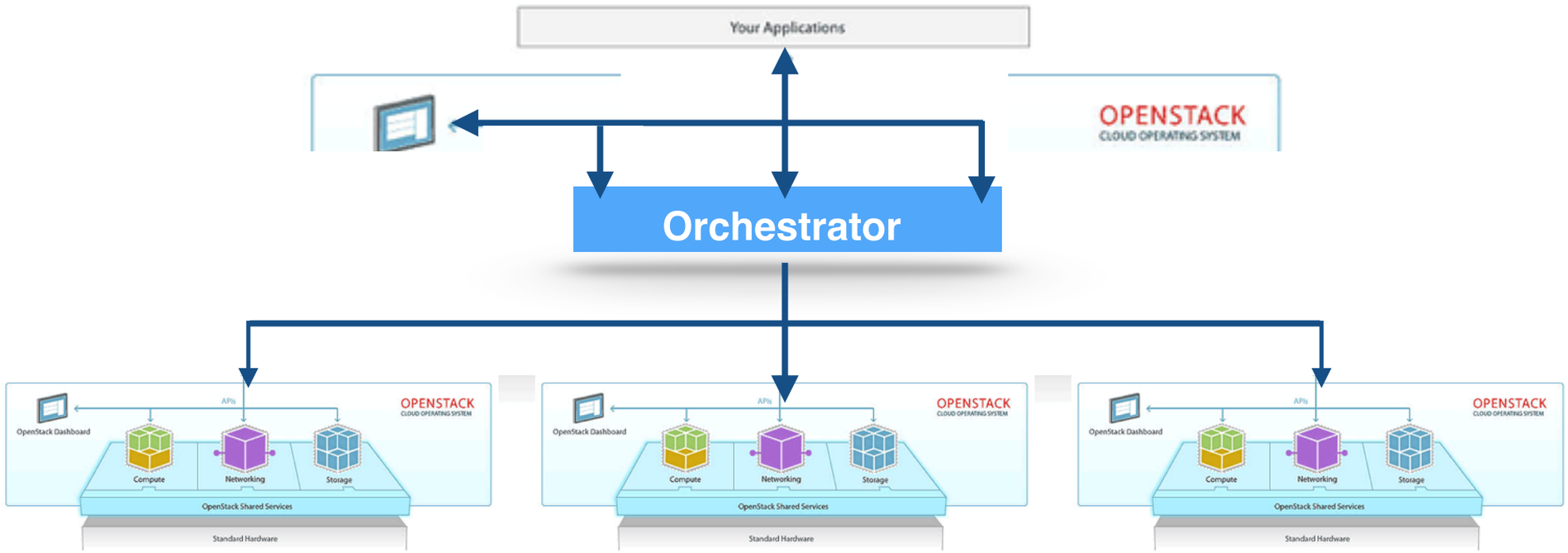
Top-Down



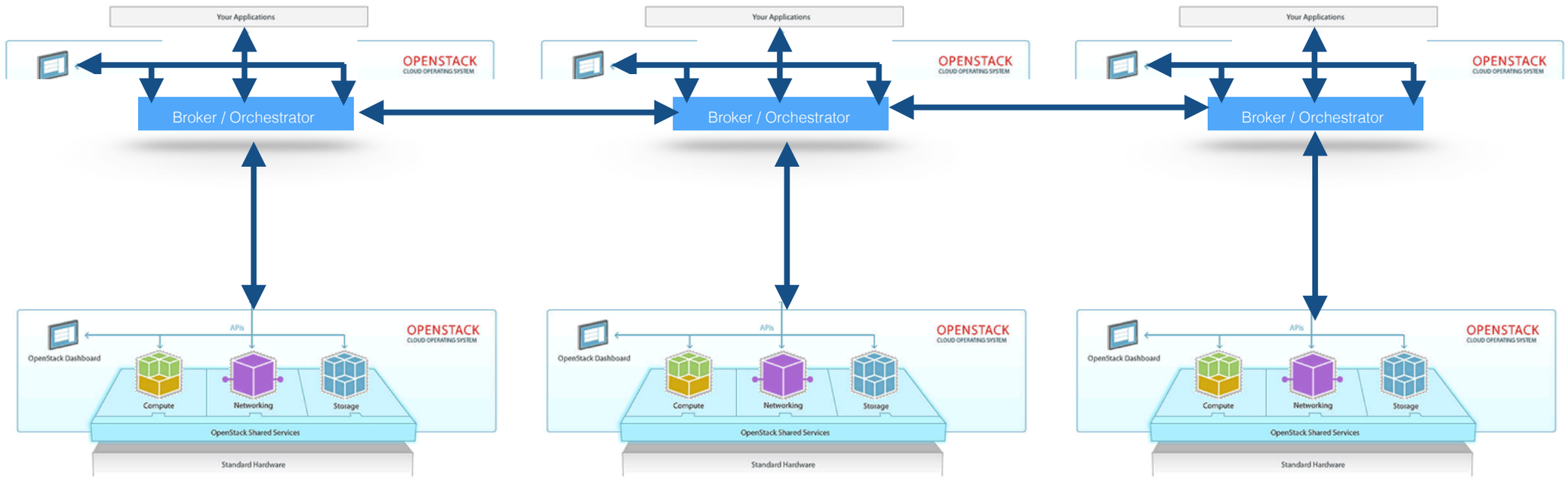
Centralized

Distributed

Top-Down

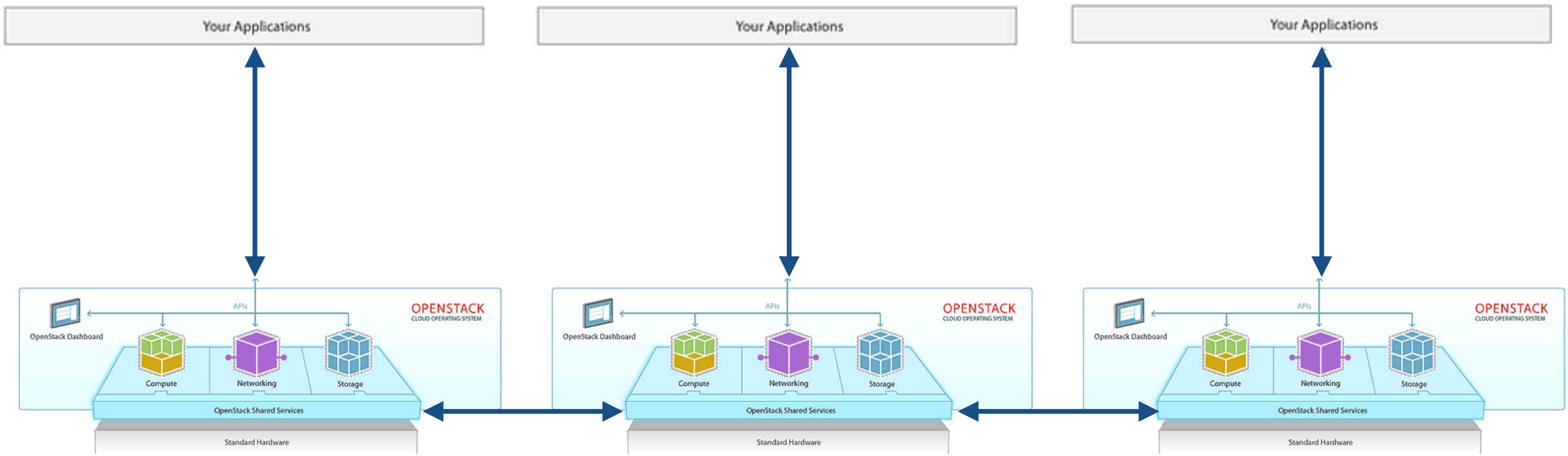


Centralized



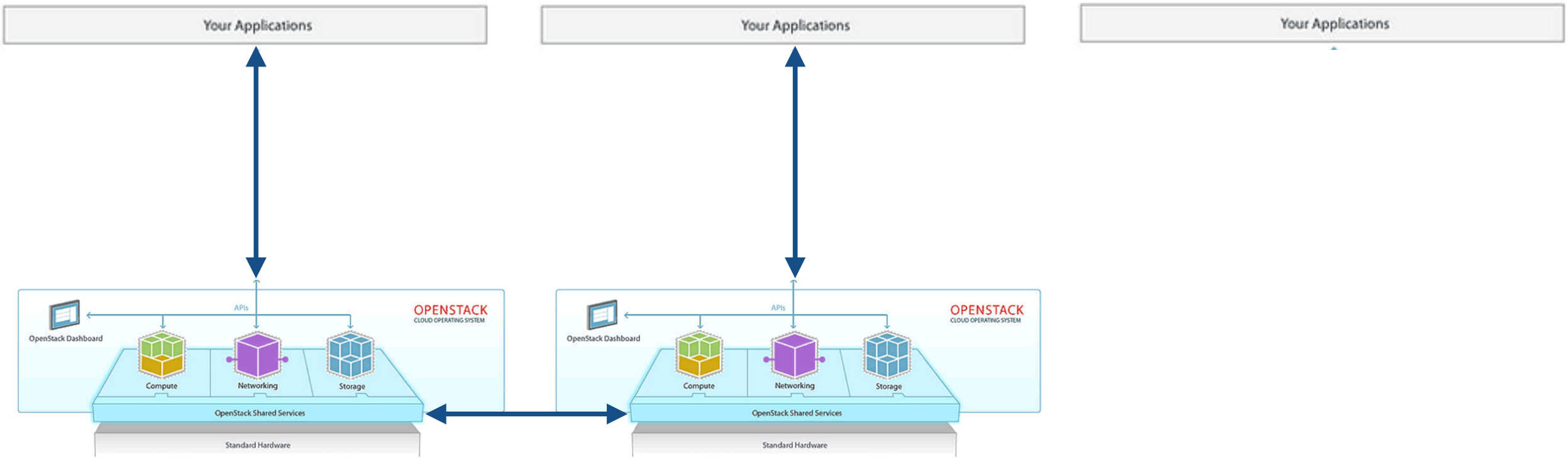
Distributed

Bottom-Up



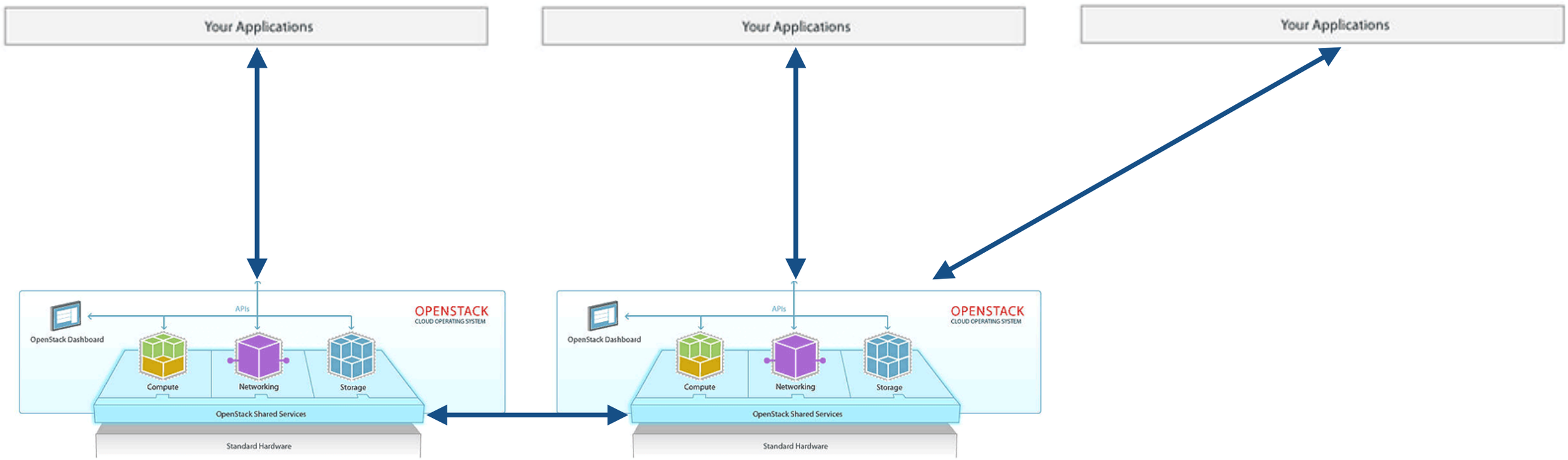
Natively distributed

Bottom-Up



Natively distributed

Bottom-Up

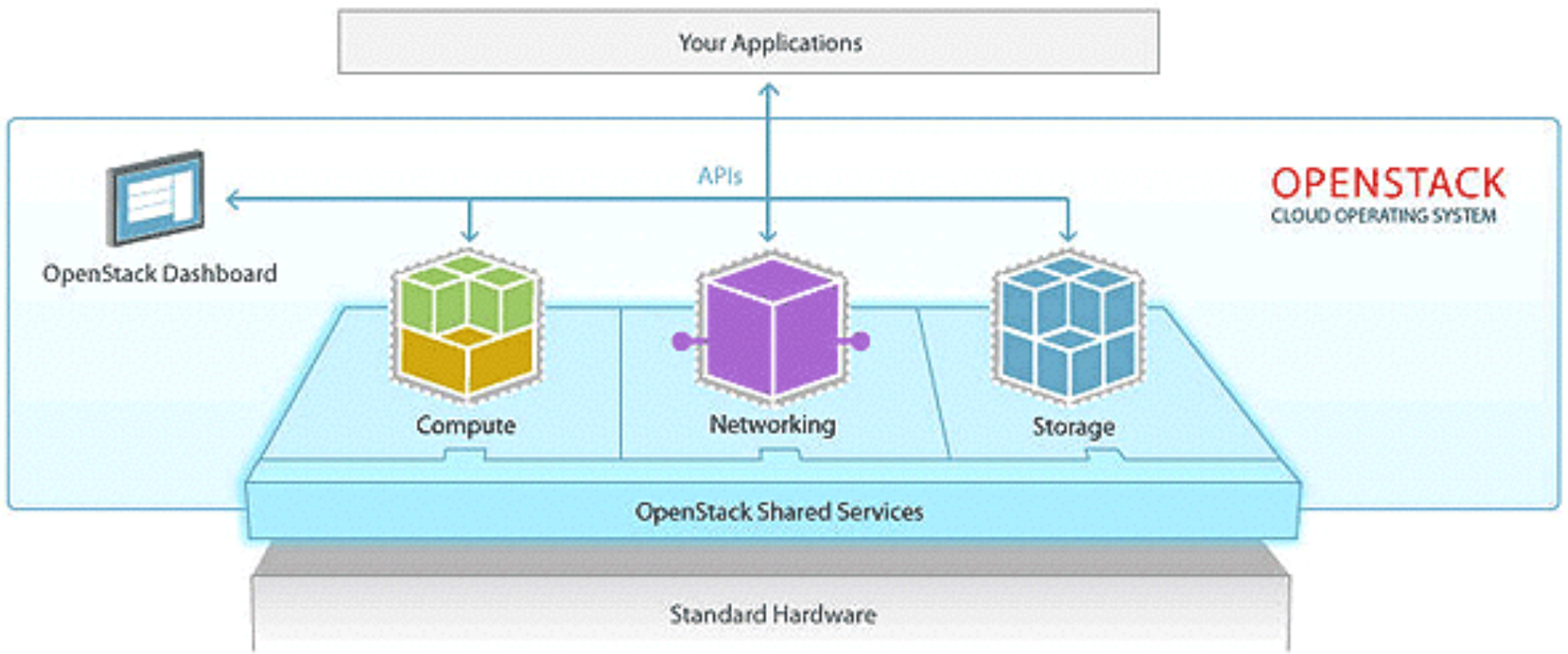


Natively distributed

One PoP == One Access Point

Revise OpenStack to Support Fog/Edge Computing Infrastructures

- Do not reinvent the wheel… it is too late
- Mitigate development efforts
  - By favoring a bottom/up approach
    Investigate whether/how OpenStack core services can become cooperative by default (using P2P and Self-* technics).



Proof of Concept for Nova

Nova PoC

openstack CLOUD SOFTWARE

openstack CLOUD SOFTWARE

Technical considerations

SIDEBAR ◂ PREV | UP | NEXT ▸

CONTENTS SEARCH

OPENSTACK MANUALS › OPENSTACK ARCHITECTURE DESIGN GUIDE - CURRENT

Technical considerations

Repurposing an existing OpenStack environment to be massively scalable is a formidable task. When building a massively scalable environment from the ground up, ensure you build the initial deployment with the same principles and choices that apply as the environment grows. For example, a good approach is to deploy the first site as a multi-site environment. This enables you to use the same deployment and segregation methods as the environment grows to separate locations across dedicated links or wide area networks. In a hyperscale cloud, scale trumps redundancy. Modify applications with this in mind, relying on the scale and homogeneity of the environment to provide reliability rather than redundant infrastructure provided by non-commodity hardware solutions.

Infrastructure segregation

OpenStack services support massive horizontal scale. Be aware that this is not the case for the entire supporting infrastructure. This is particularly a problem for the database management systems and message queues that OpenStack services use for data storage and remote procedure call communications.

Traditional clustering techniques typically provide high availability and some additional scale for these environments. In the quest for massive scale, however, you must take additional steps to relieve the performance pressure on these components in order to prevent them from negatively impacting the overall performance of the environment. Ensure that all the components are in balance so that if the massively scalable environment fails, all the components are near maximum capacity and a single component is not causing the failure.

Nova PoC

Technical considerations

openstack
CLOUD SOFTWARE

SII ... UP | NEXT ▸

OPENSTACK MANUALS ▸ OPENSTACK ARCHITECTURE DESIGN GUIDE - CURRENT

Technical considerations

Repurposing an existing OpenStack environment to be massively scalable is a formidable task. When building a massively scalable environment from the ground up, ensure you build the initial deployment with the same principles and choices that apply as the environment grows. For example, a good approach is to deploy the first site as a multi-site environment. This enables you to use the same deployment and segregation methods as the environment grows to separate locations across dedicated links or wide area networks. In a hyperscale cloud, scale trumps redundancy. Modify applications with this in mind, relying on the scale and homogeneity of the environment to provide reliability rather than redundant infrastructure provided by non-commodity hardware solutions.

Infrastructure segregation

OpenStack services support massive horizontal scale. Be aware that this is not the case for the entire supporting infrastructure. This is particularly a problem for the database management systems and message queues that OpenStack services use for data storage and remote procedure call communications.

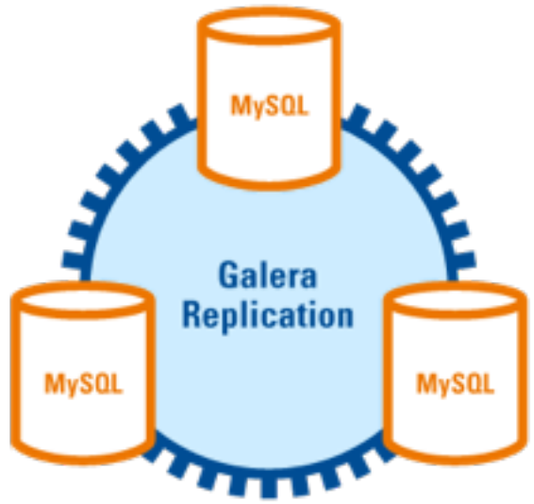
Traditional clustering techniques typically provide high availability and some additional scale for these environments. In the quest for massive scale, however, you must take additional steps to relieve the performance pressure on these components in order to prevent them from negatively impacting the overall performance of the environment. Ensure that all the components are in balance so that if the massively scalable environment fails, all the components are near maximum capacity and a single component is not causing the failure.

Distributing OpenStack Through a Bottom/Up Approach

- Step 1: OpenStack shared services

  A SQL database 

  A messaging queue 



- Active/Active replication

  Production ready but does not scale to our target.

- Key/Value Store systems

  Alternate solutions for storing states over a highly distributed infrastructure

Distributing OpenStack Through a Bottom/Up Approach

- Step 1: OpenStack shared services

  A SQL database

  A messaging queue

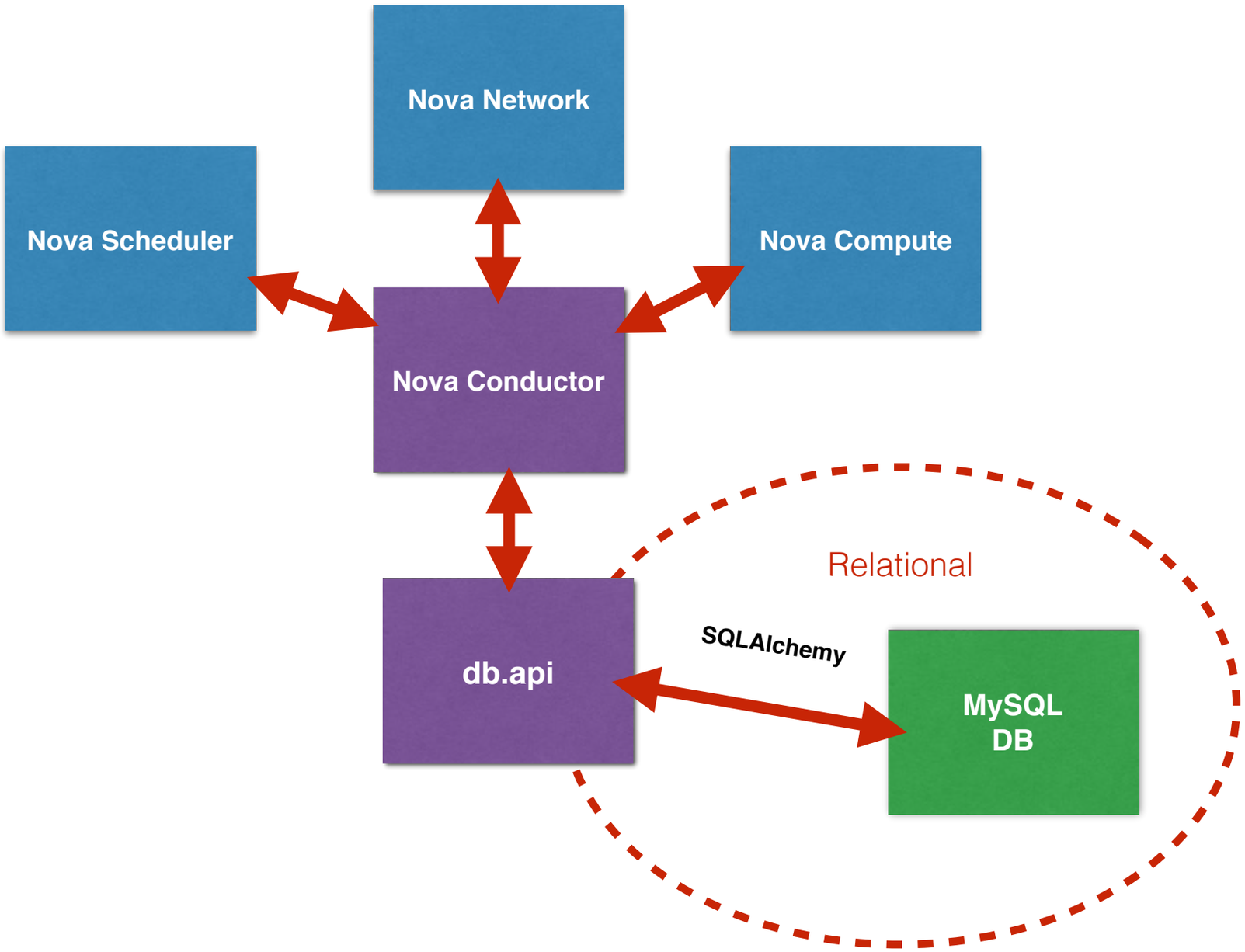
- Active/A… but does not scale to our target.

- …y/Value Store systems

  Alternate solutions for storing states over a highly distributed infrastructure

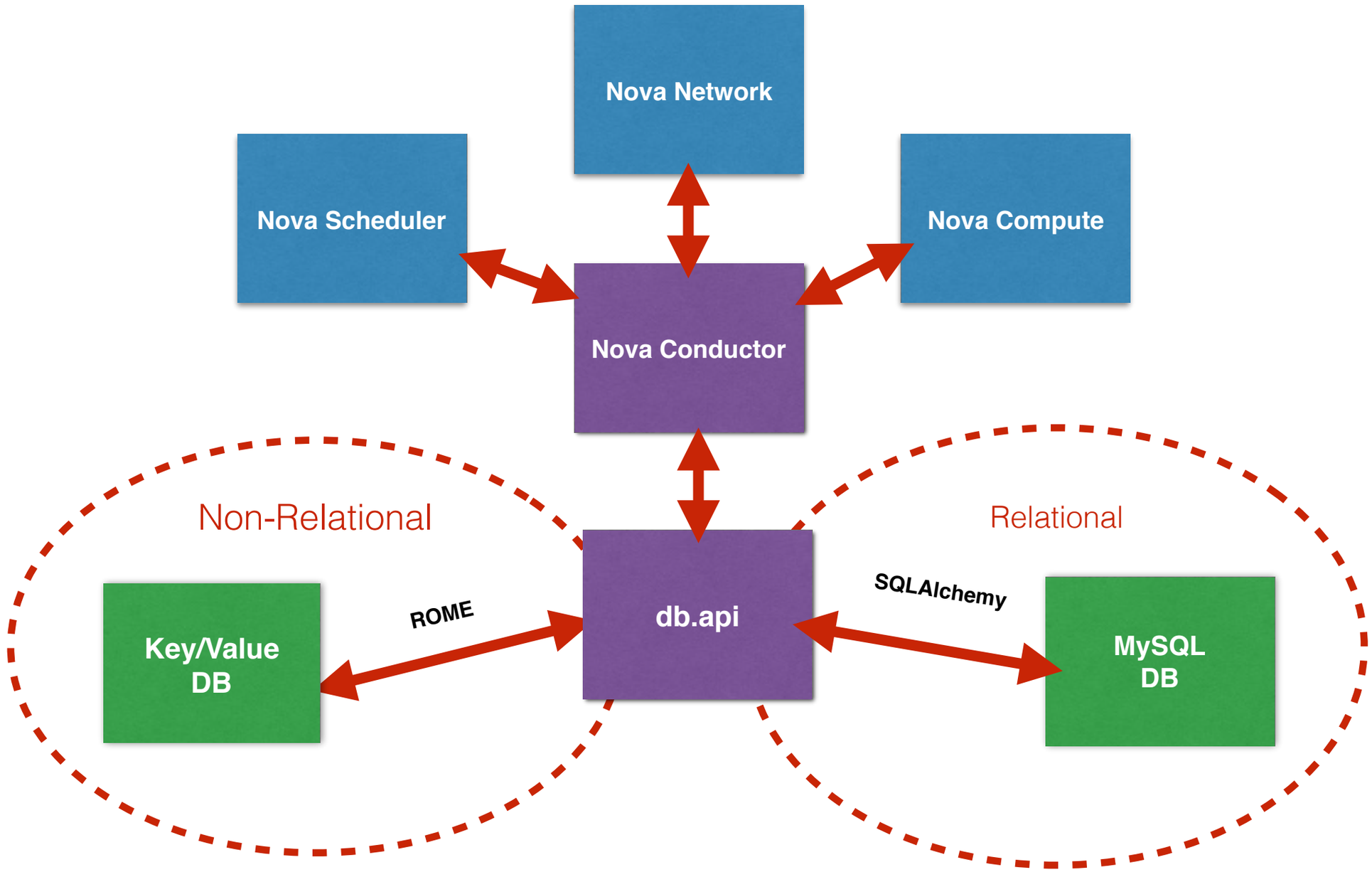
Problem: going from a SQL to a NoSQL backend for the inner states of OpenStack (with minimal change)?

Leveraging a Key/Value Store DB



Nova (compute service) – software architecture

Leveraging a Key/Value Store DB

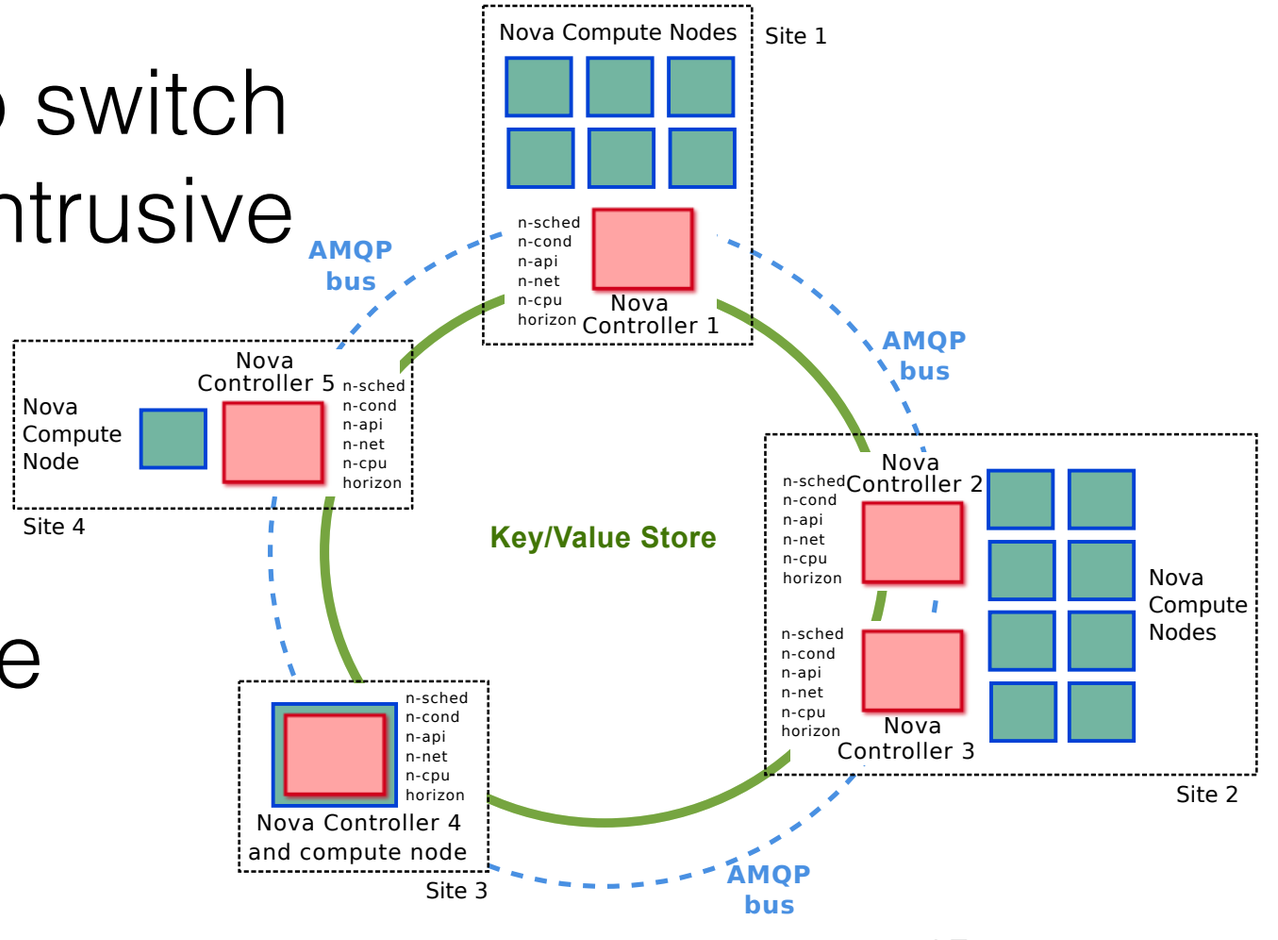


Nova (compute service) – software architecture

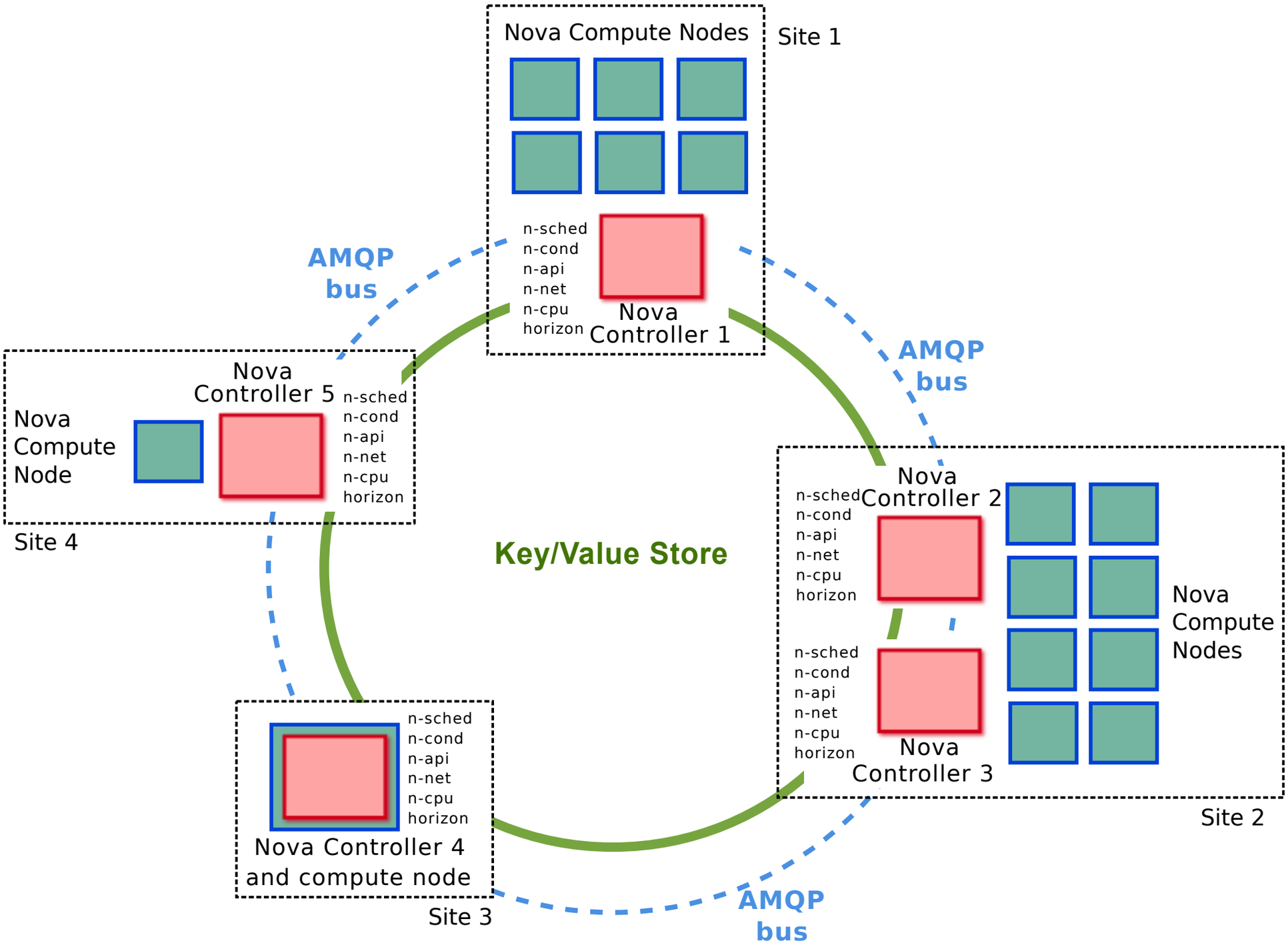
ROME

- **Relational Object Mapping Extension** for key/value stores (Jonathan Pastor's Phd)
  https://github.com/BeyondTheClouds/rome
- Enables the query of Key/Value Store DB with the same interface as SQLAlchemy

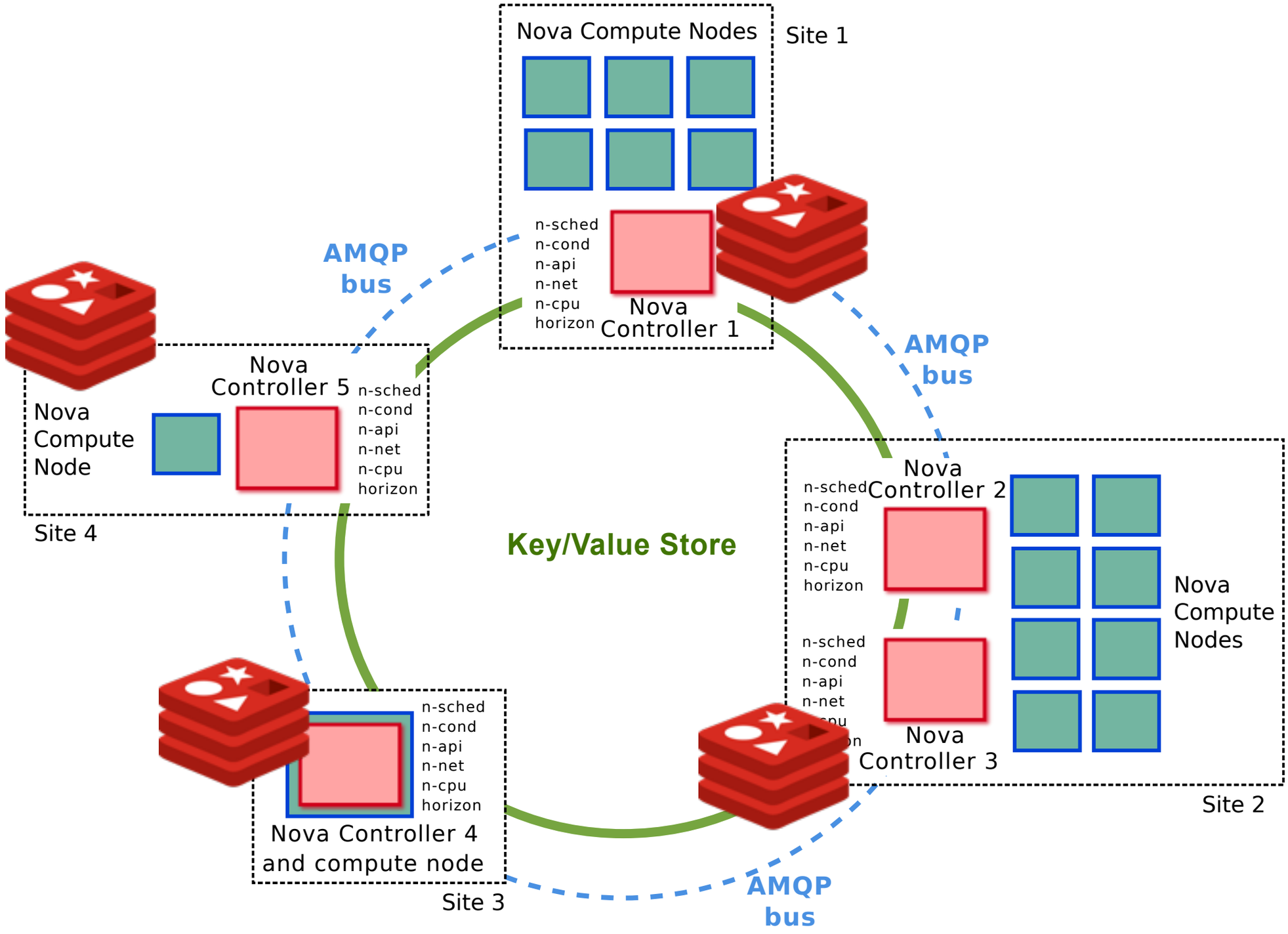
- Enables Nova OpenStack to switch to a KVS without being too intrusive
- The KVS is distributed over (dedicated) nodes
- Nova services connect to the Key/value store cluster



Nova Proof-of-Concept



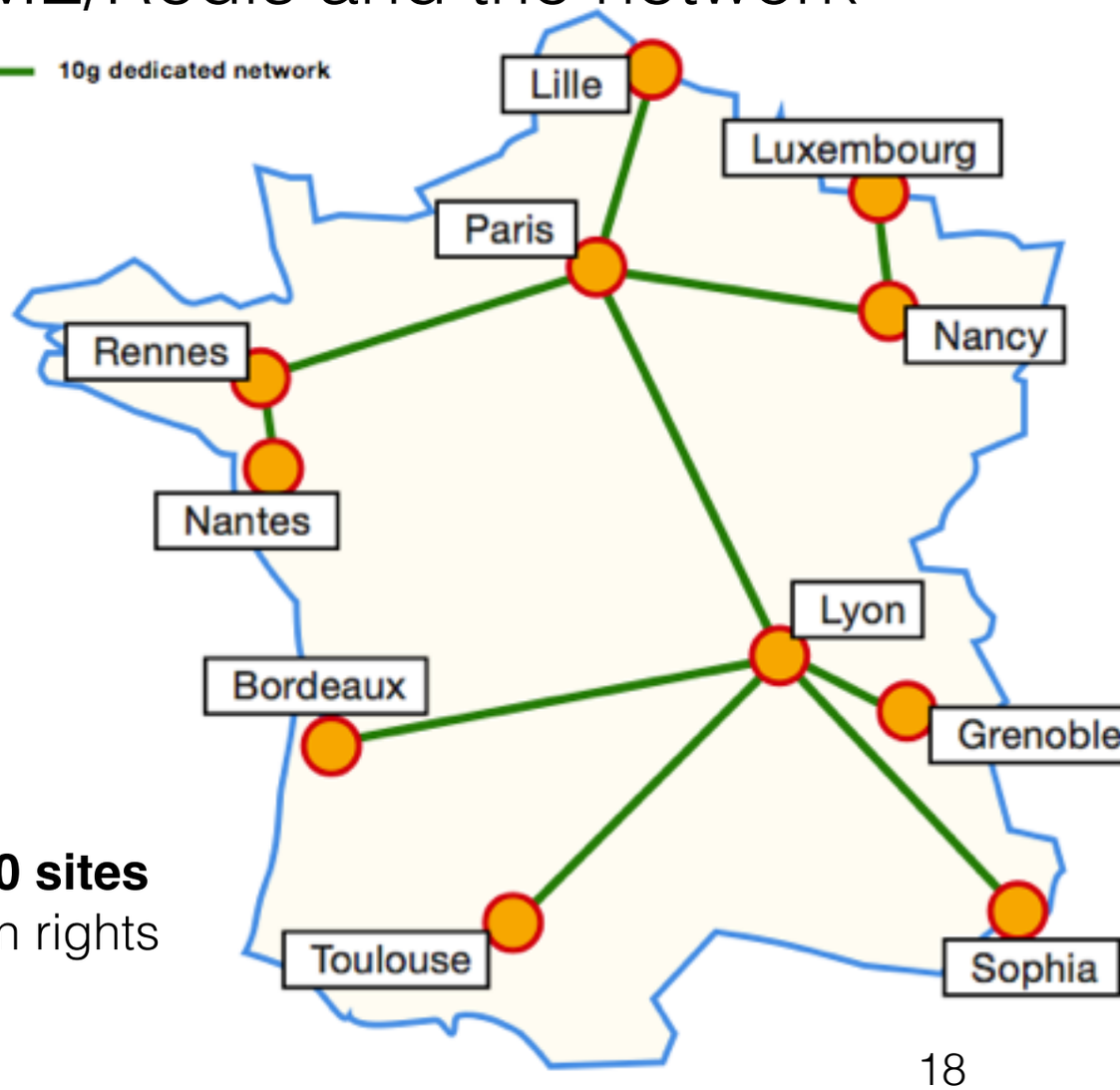
Nova Proof-of-Concept



Evaluation

Evaluation

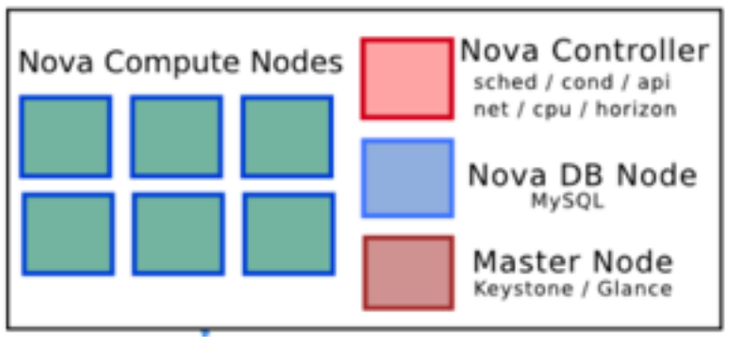
- Experiments have been conducted on **Grid'5000**
- On OpenStack Kilo
- Single-site experiments
  ⟹ Evaluate the overhead of using ROME/Redis and the network impact.
- Multi-site experiments
  ⟹ Determine the impact of latency.
  ⟹ Validate compatibility with higher level mechanisms validation



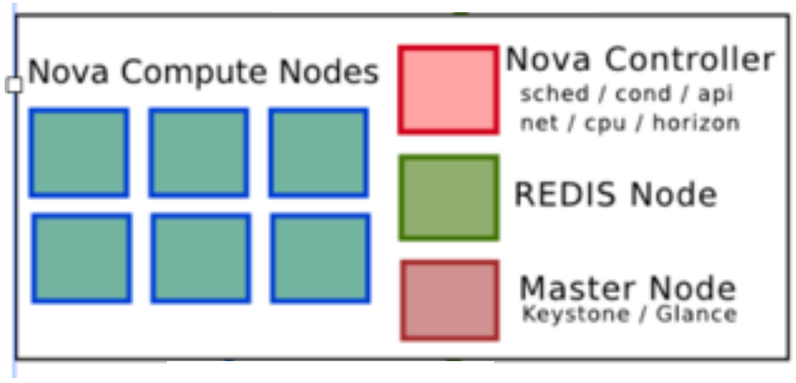
https://www.grid5000.fr
1,500 servers, spread across **10 sites**
Bare metal resources, full admin rights

Single-Site Experiments

- Creation of 500 VMs
- Comparison MySQL/SQLAlchemy vs ROME/Redis (one dedicated node for the DB server)



MySQL/SQLAlchemy



ROME/Redis

Single-Site Experiments

- Evaluate the overhead of using ROME/Redis
- Redis stores objects in a JSON format: *serialization/deserialization cost*
- ROME reimplements some mechanisms: *join, transaction/session, …*

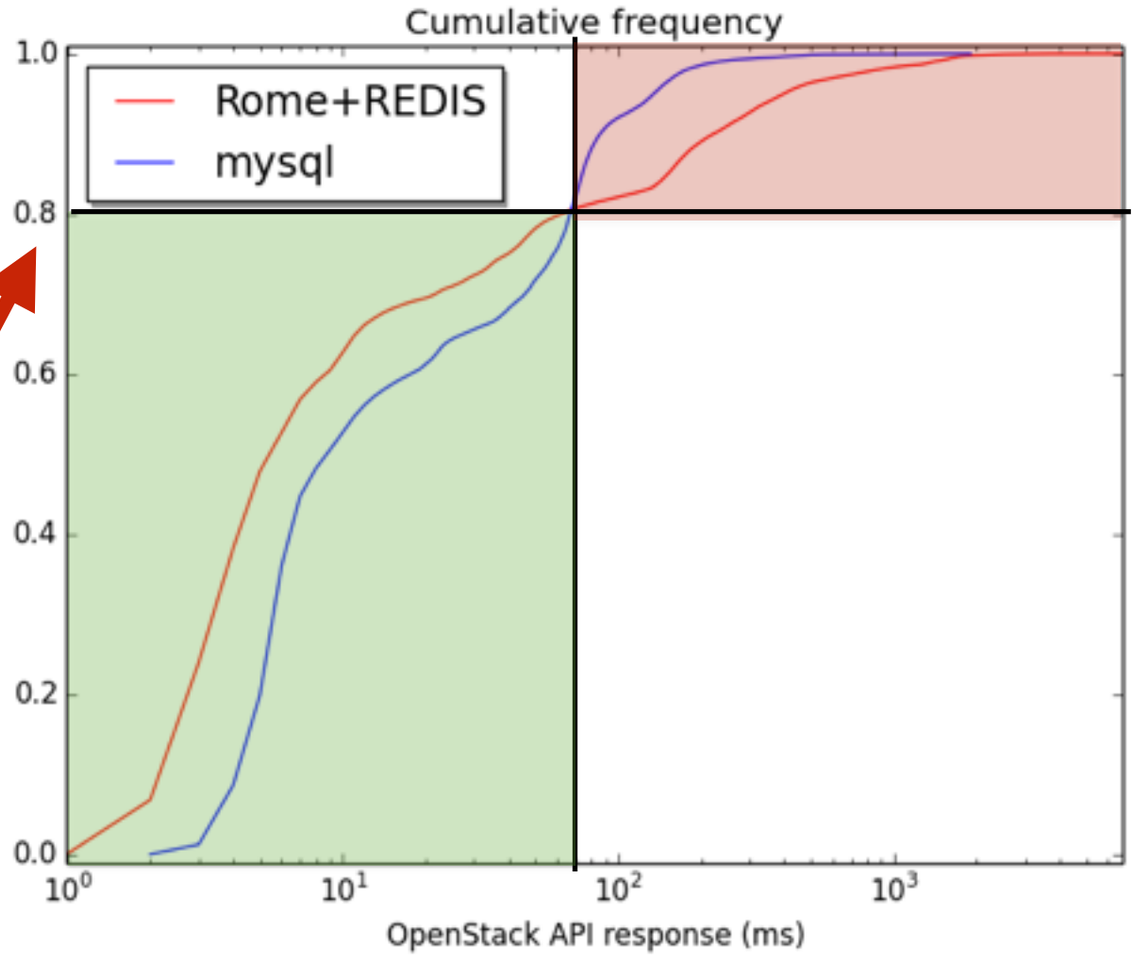
TABLE I
AVERAGE RESPONSE TIME TO API REQUESTS FOR A MONO-SITE DEPLOYMENT (IN MS).

Backend configuration	Redis	MySQL
1 node	83	37
4 nodes	82	-
4 nodes + repl	91	-

Single-Site Experiments

- Evaluate the overhead of using ROME/Redis
- Redis stores objects in a JSON format: *serialization/deserialization cost*
- ROME reimplements some mechanisms: *join, transaction/session, …*

ROME requests are faster for 80% of requests



SQLAlchemy is faster for 20% of requests

Single-Site Experiments

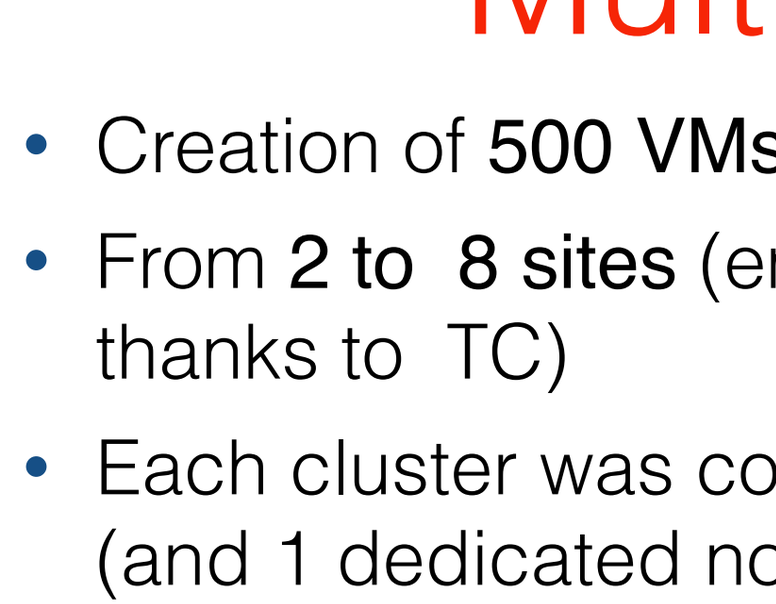
- Evaluate the overhead of using ROME/Redis
- Redis stores objects in a JSON format: *serialization/deserialization cost*
- ROME reimplements some mechanisms: *join, transaction/session, …*

TABLE II
TIME USED TO CREATE 500 VMS ON A SINGLE CLUSTER CONFIGURATION (IN SEC.)

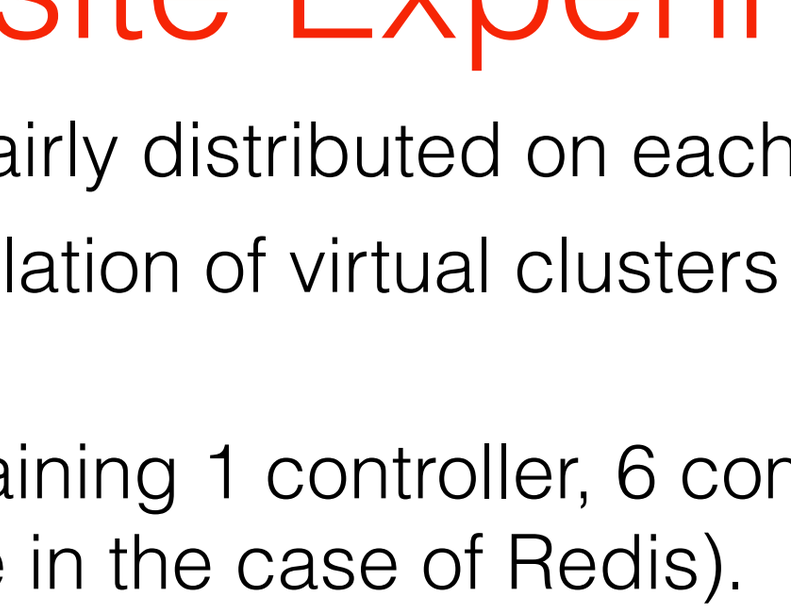
Backend configuration	Redis	MySQL
1 node	322	298
4 nodes	327	-
4 nodes + repl	413	-

Multi-site Experiments

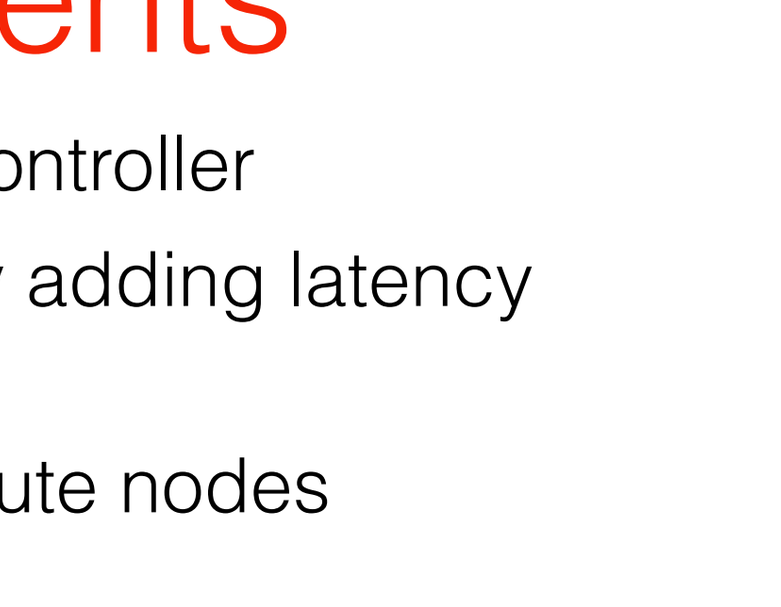
- Creation of **500 VMs**, fairly distributed on each controller
- From **2 to 8 sites** (emulation of virtual clusters by adding latency thanks to TC)
- Each cluster was containing 1 controller, 6 compute nodes (and 1 dedicated node in the case of Redis).
- MySQL and Redis used in the default configuration
- To fairly compare with MySQL, data replication was not activated in Redis



SQLAlchemy+MySQL

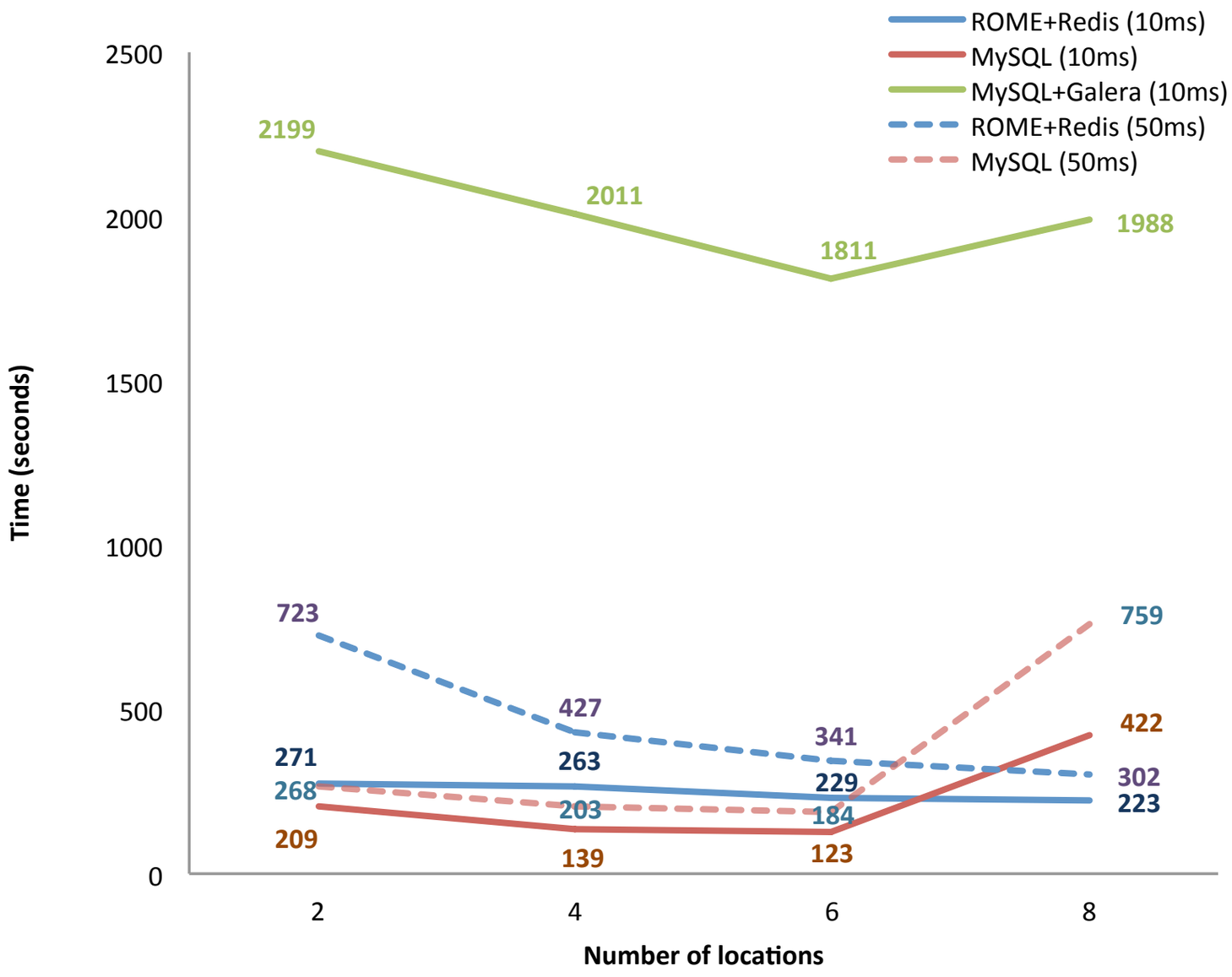


SQLAlchemy+Galera

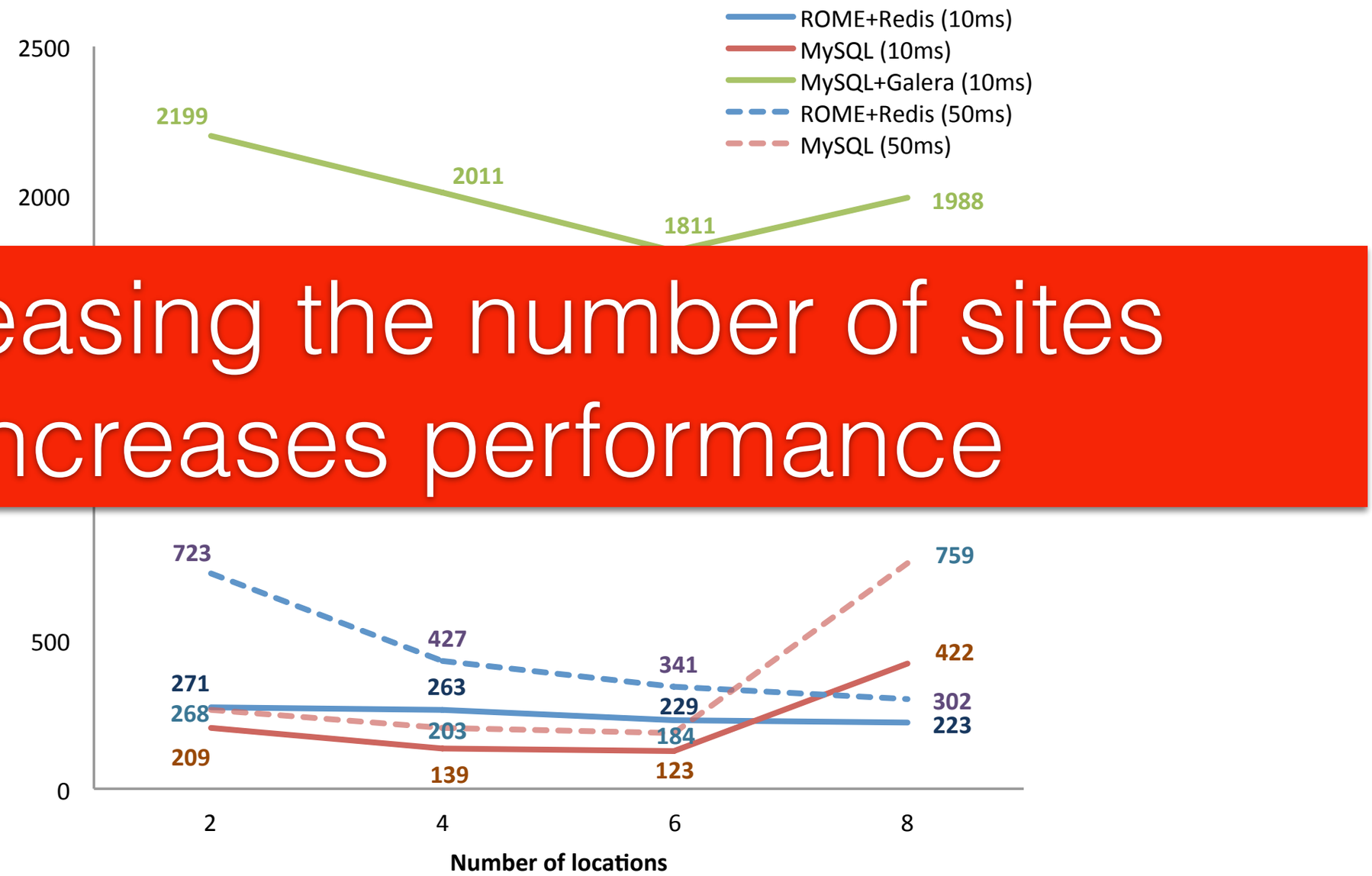


ROME+Redis

Multi-Site Experiments



Multi-Site Experiments



Increasing the number of sites increases performance

Ongoing Work

- Building an Internet-scale IaaS manager…
  - Co-chair of OpenStack Massively Distributed Clouds WG
    https://wiki.openstack.org/wiki/Massively_Distributed_Clouds
  - Involved in several other Working Groups (Performance, NFV)
- …including the most important services
  (Glance, Neutron, Cinder, Keystone, …)
- + Several on-going Working Groups Discovery
  (Architecture, Placement, Data management)
- Considering other backends than Redis
  (What about NewSQL?)
- Enos: an Experimental eNvironment for OpenStack 🐒

  https://github.com/BeyondTheClouds/enos

  (CCGrid 2017)

Thank You!

Questions?

anthony.simonet@inria.fr

http://beyondtheclouds.github.io/