OpenStack on the Edge
BoF Session - Boston 2017

Cockroach Labs

Adrien Lebre
Inria
The Discovery Initiative
Fog Computing / Edge Computing/ Massively Distributed Clouds Working Group

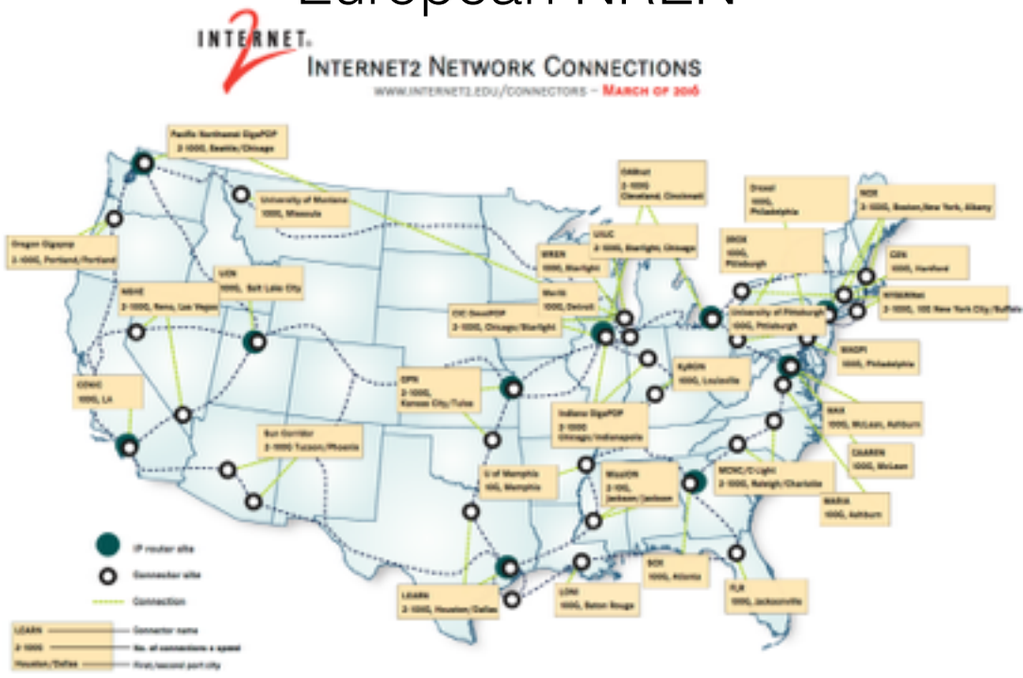
Fog/Edge/MDC Infrastructures

- Leverage network backbones

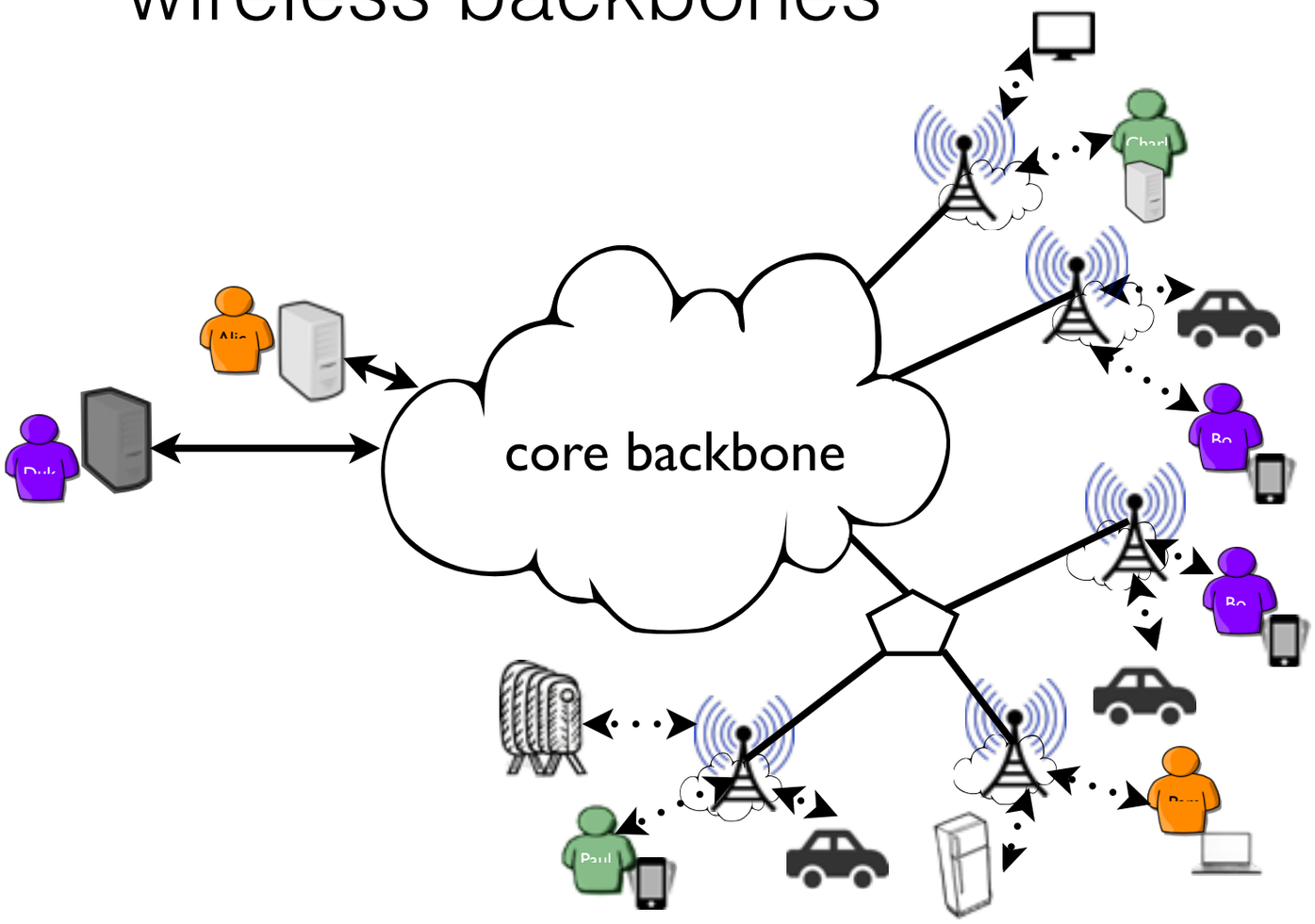
  Extend any point of presence of network backbones (aka PoP) with servers (from network hubs up to major DSLAMs that are operated by telecom companies, network institutions…).

- Extend to the edge by including wireless backbones

European NREN



USA NREN



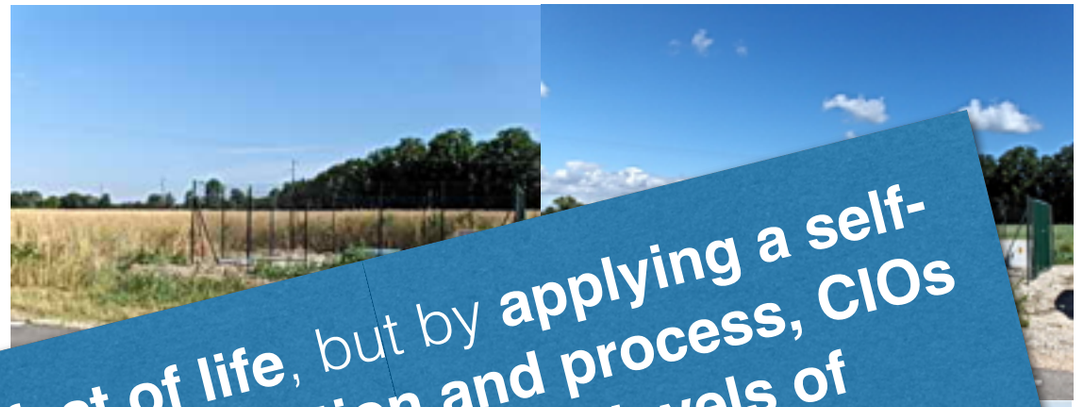
Micro/Nano DCs

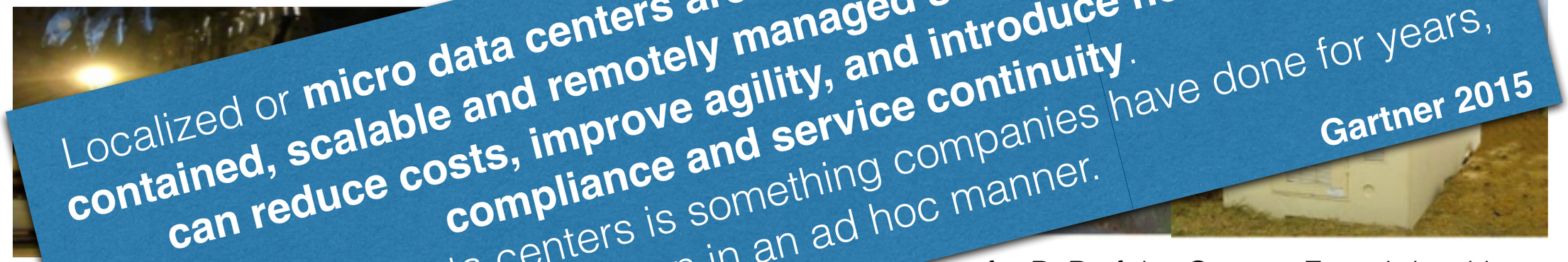
Sagrada Familia microDC
(Barcelona, Spain)

Deployment of a PoP of the Orange French backbone

MDC Industry - Brazil

Micro/Nano DCs





Localized or **micro data centers are a fact of life**, but by **applying a self-contained, scalable and remotely managed solution and process, CIOs can reduce costs, improve agility, and introduce new levels of compliance and service continuity**.

Creating micro data centers is something companies have done for years, but often in an ad hoc manner.

Gartner 2015

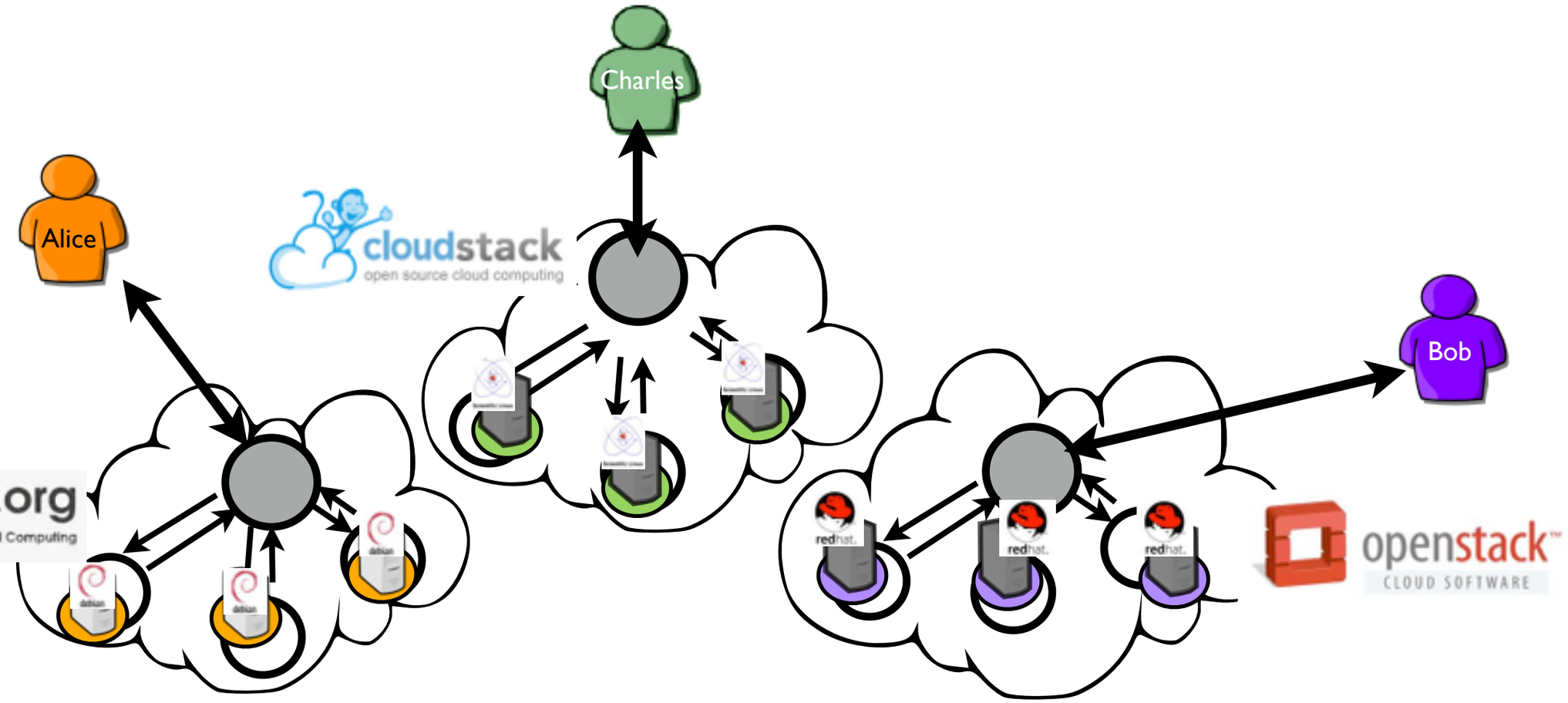
ployment of a PoP of the Orange French backbone

Sagrada Familia microDC
(Barcelona, Spain)

MDC Industry - Brazil

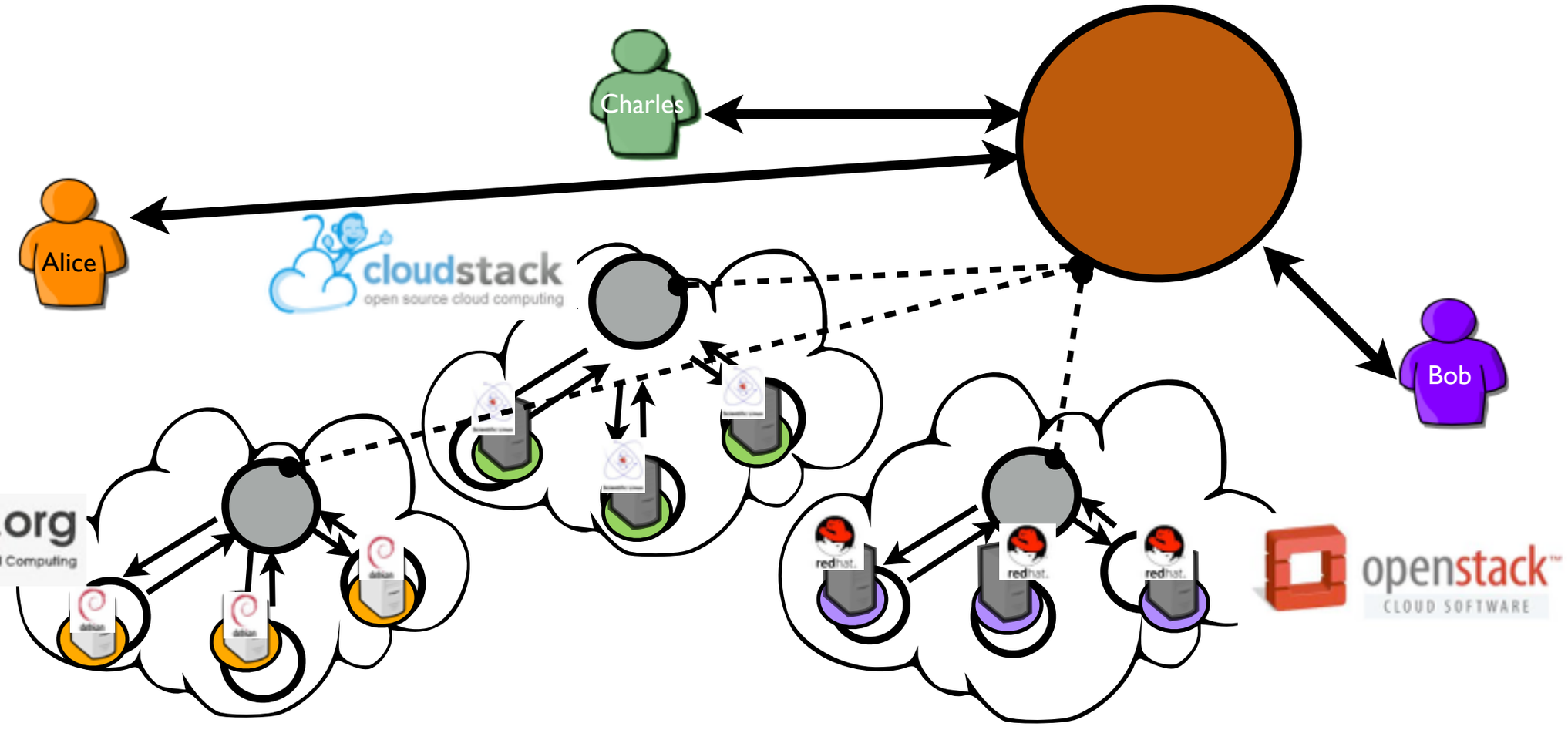
What's about Brokering Approaches?

- Sporadic (hybrid computing/cloud bursting) almost ready for production
- While standards are coming (OCCI…), current brokers are rather limited to simple usages and not advanced administration operations



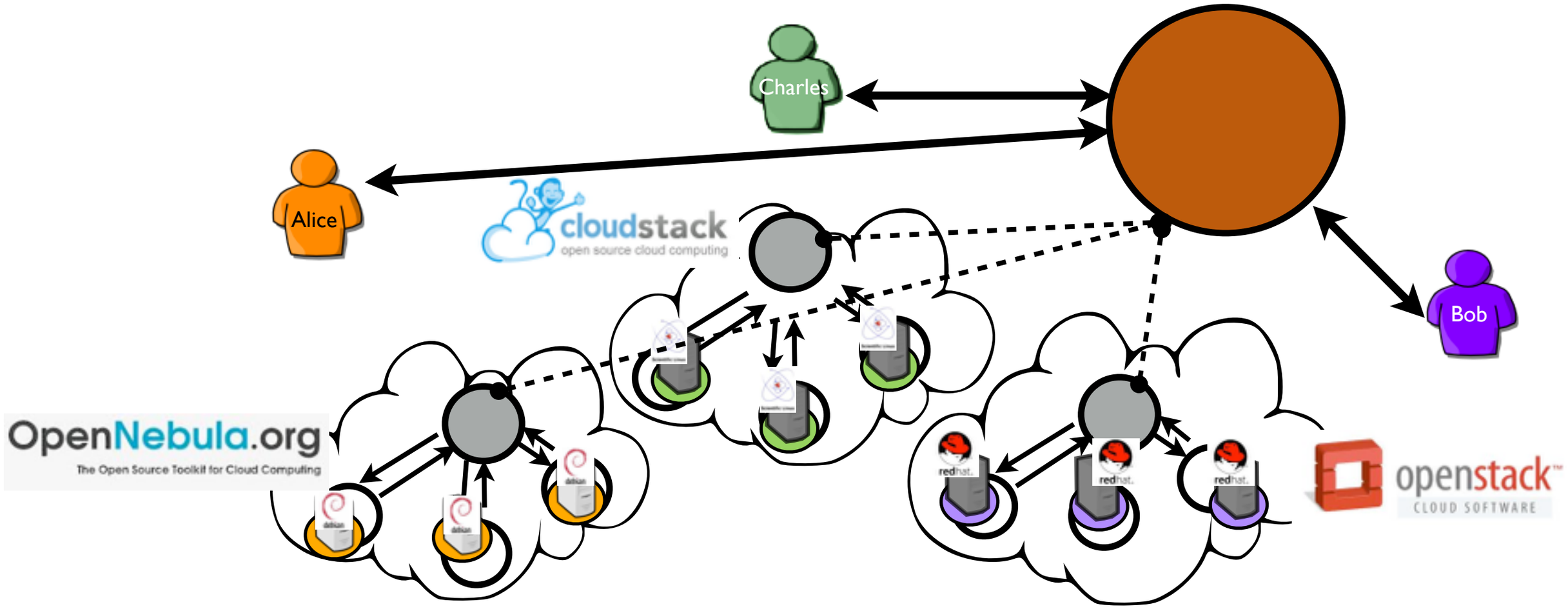
What's about Brokering Approaches?

- Sporadic (hybrid computing/cloud bursting) almost ready for production
- While standards are coming (OCCI…), current brokers are rather limited to simple usages and not advanced administration operations



What's about Brokering Approaches?

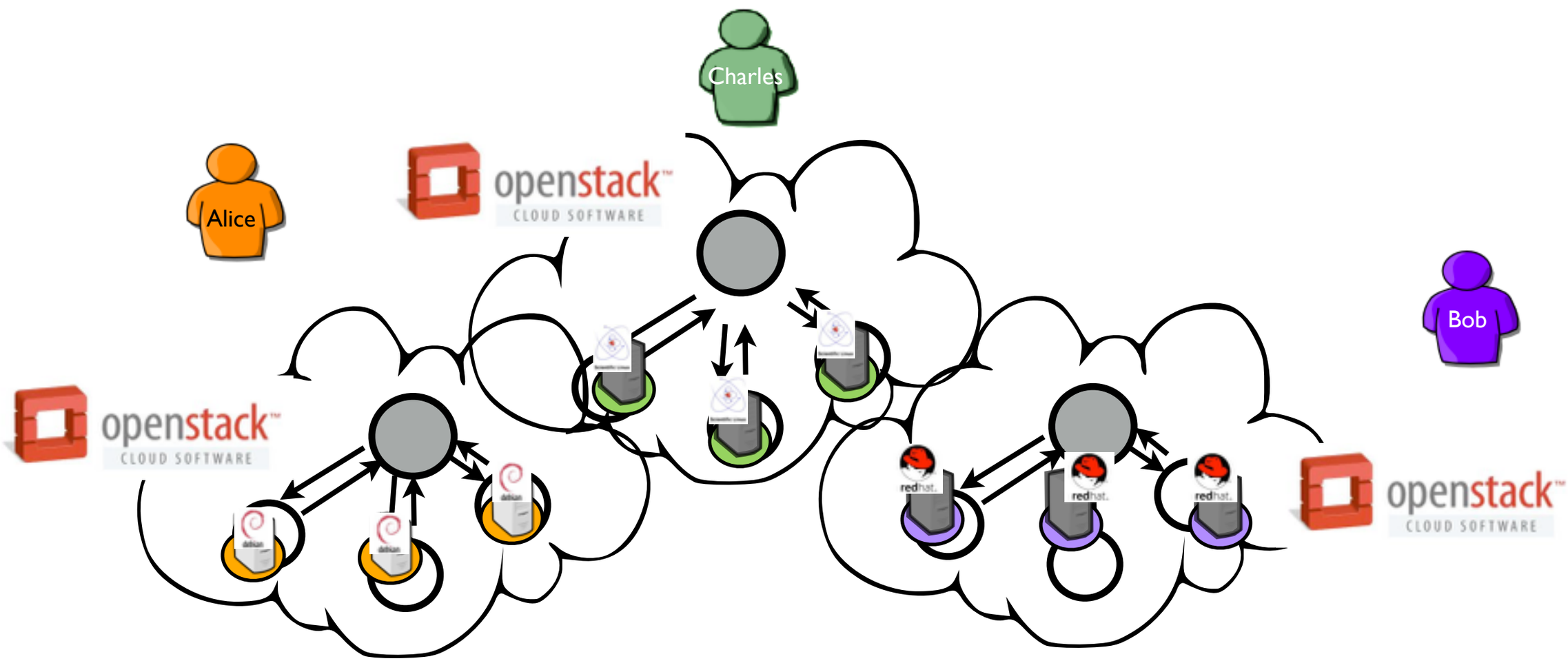
- Sporadic (hybrid computing/cloud bursting) almost ready for production
- While standards are coming (OCCI…), current brokers are rather limited to simple usages and not advanced administration operations



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

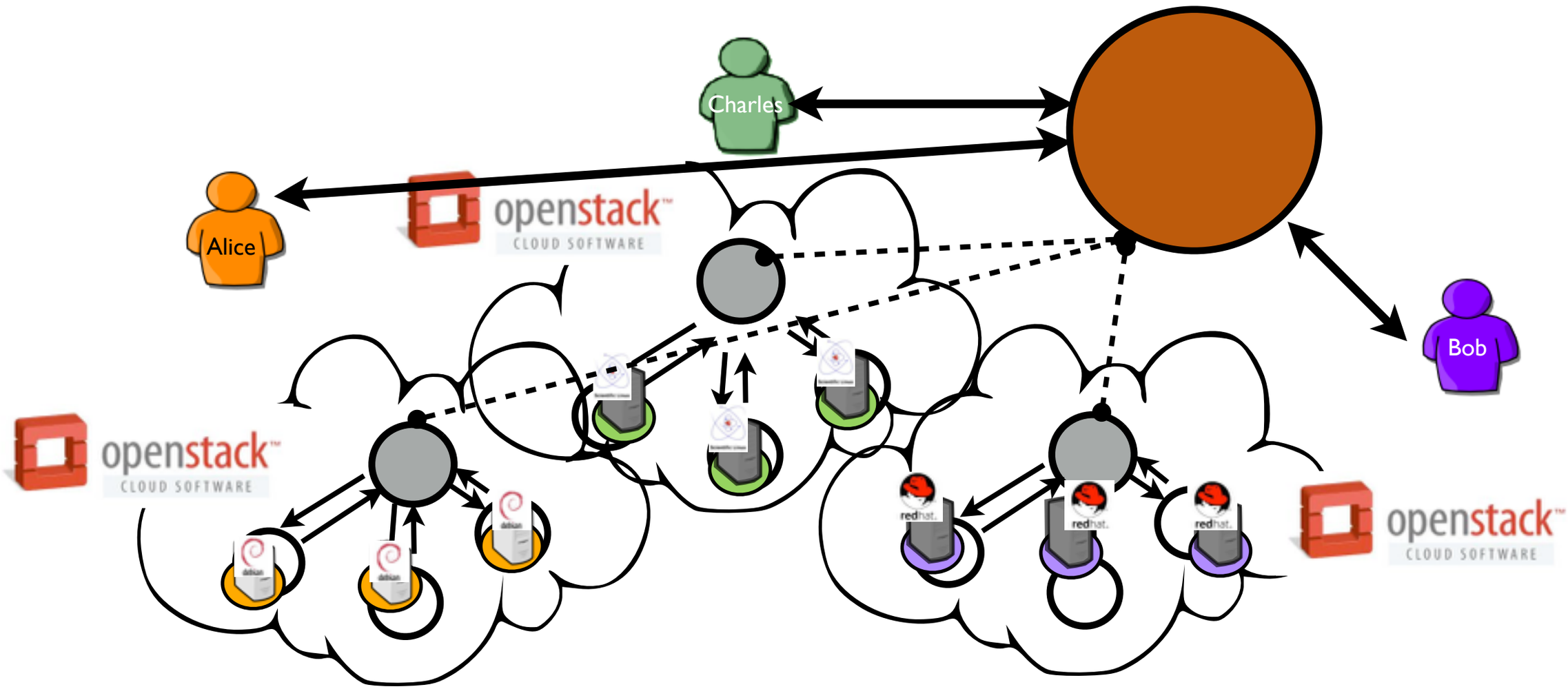
Would OpenStack be the solution?

- Do not reinvent the wheel… it is too late



Would OpenStack be the solution?

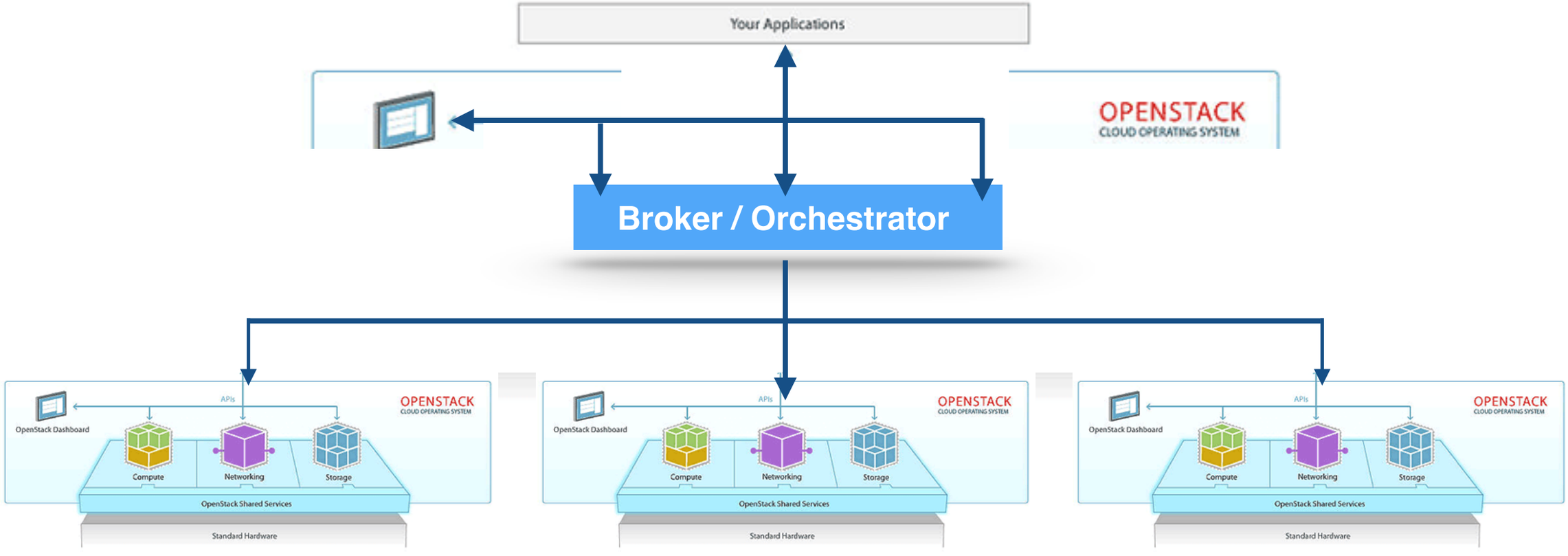
- Do not reinvent the wheel… it is too late



Would OpenStack be the solution?

- Do not reinvent the wheel… it is too late
- Few proposals to federate/operate distinct OpenStack DCs

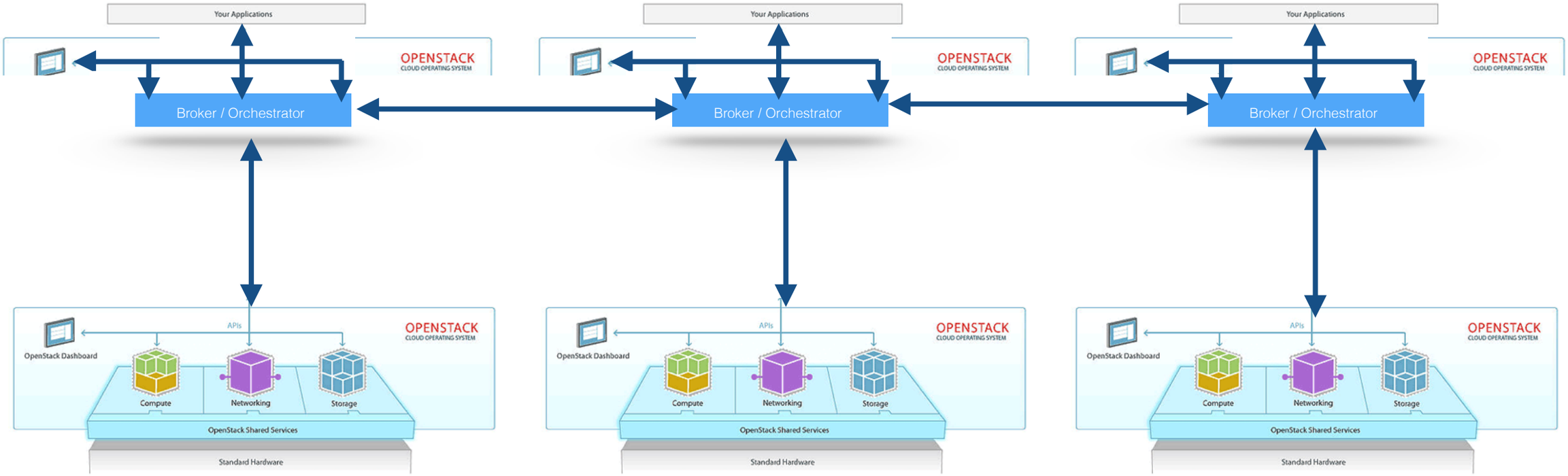
Top/Down: add a substrate to pilot independent OpenStack instances.



Would OpenStack be the solution?

- Do not reinvent the wheel… it is too late
- Few proposals to federate/operate distinct OpenStack DCs

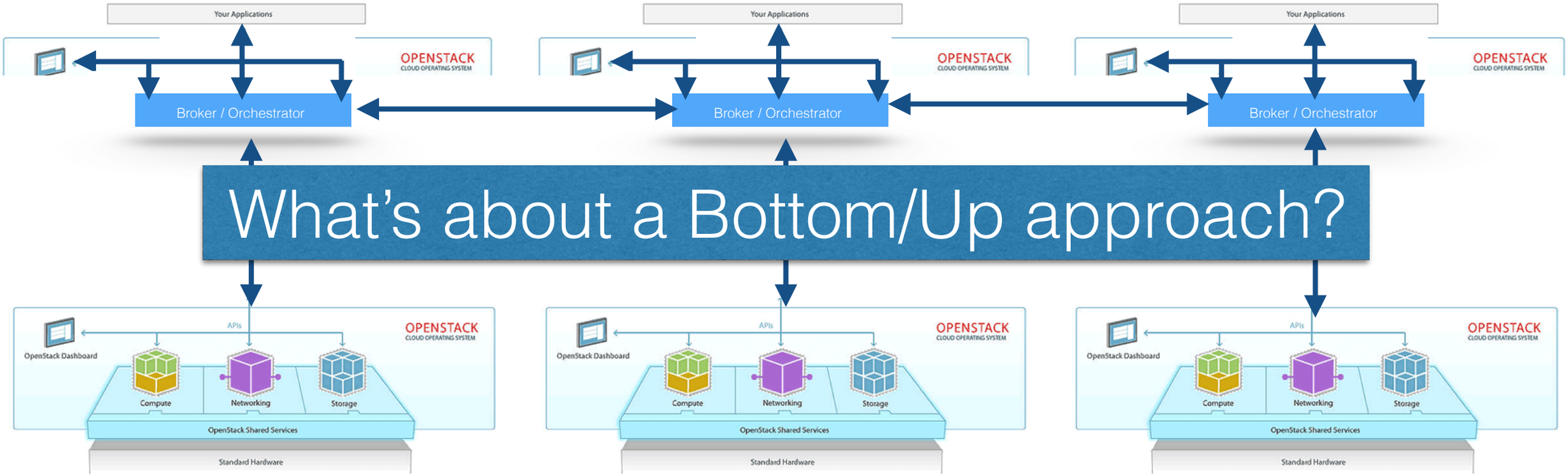
Top/Down: add a substrate to pilot independent OpenStack instances.



Would OpenStack be the solution?

- Do not reinvent the wheel… it is too late
- Few proposals to federate/operate distinct OpenStack DCs

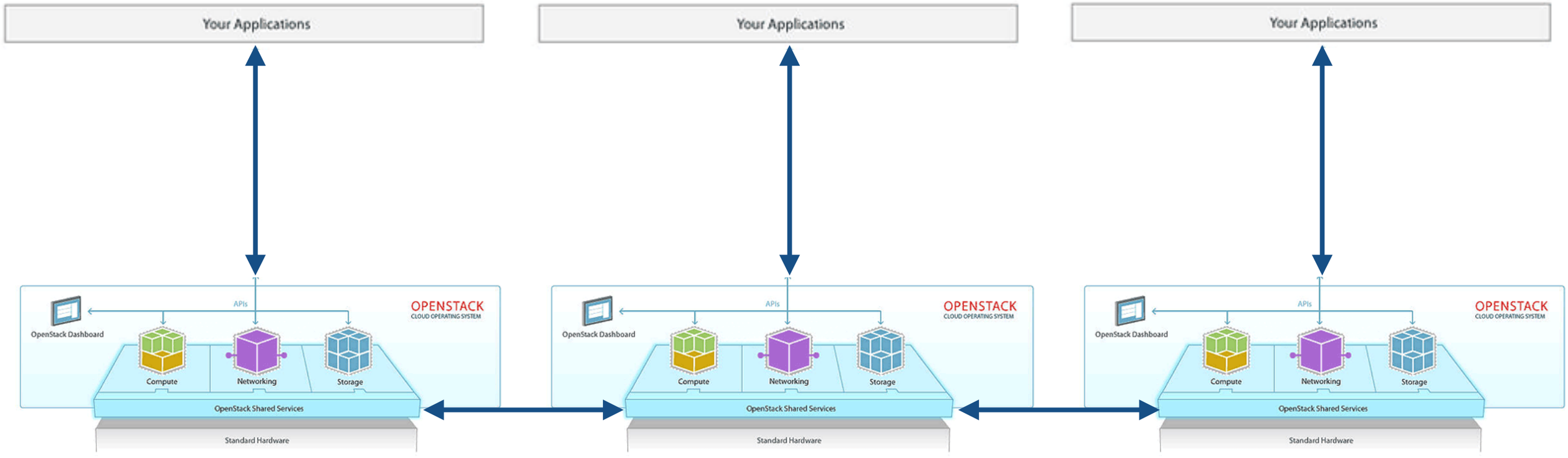
Top/Down: add a substrate to pilot independent OpenStack instances.



Would OpenStack be the solution?

- Do not reinvent the wheel… it is too late
- Few proposals to federate/operate distinct OpenStack DCs

Bottom/Up - investigate whether/how OpenStack core services can be cooperative by default using Self* and P2P mechanisms

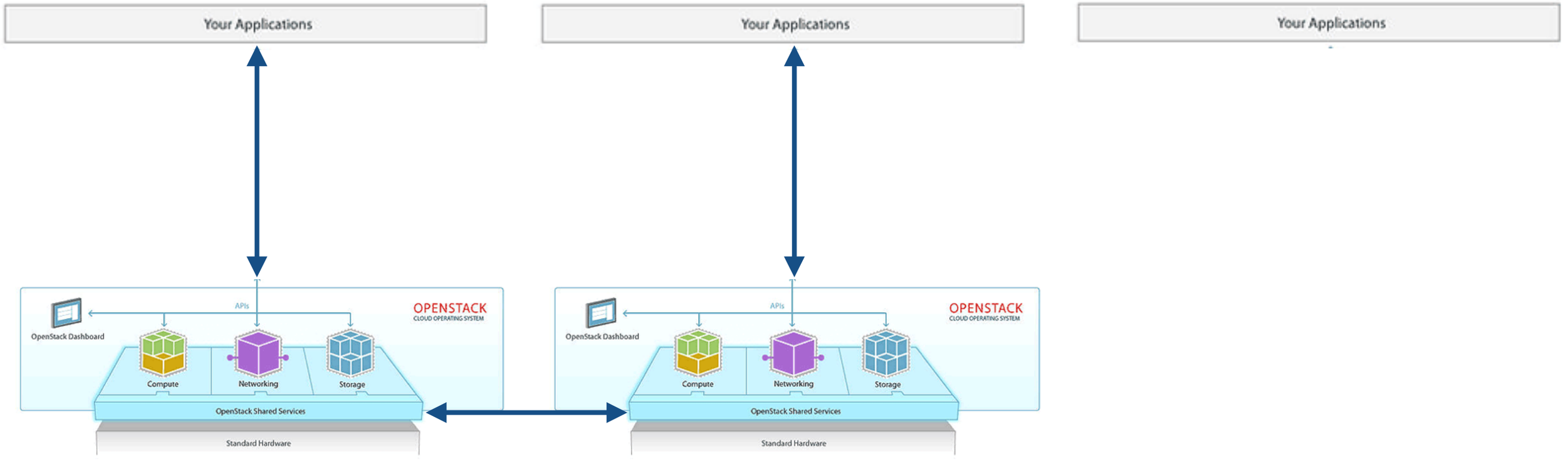


Natively distributed/cooperative

Would OpenStack be the solution?

- Do not reinvent the wheel… it is too late
- Few proposals to federate/operate distinct OpenStack DCs

Bottom/Up - investigate whether/how OpenStack core services can be cooperative by default using Self* and P2P mechanisms

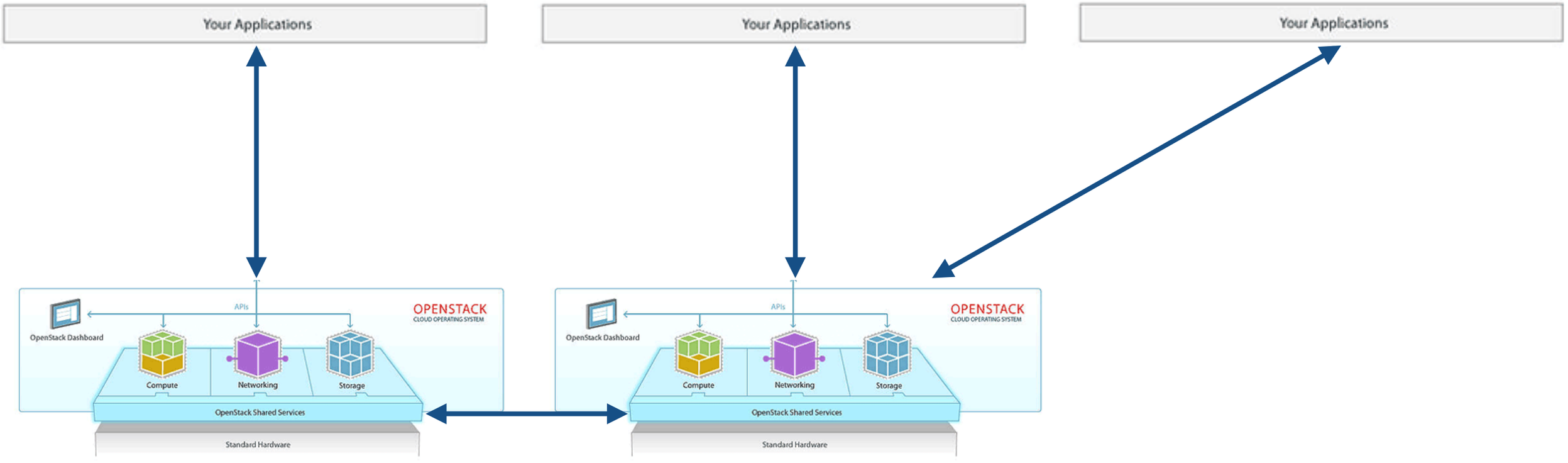


Natively distributed/cooperative

Would OpenStack be the solution?

- Do not reinvent the wheel… it is too late
- Few proposals to federate/operate distinct OpenStack DCs

Bottom/Up - investigate whether/how OpenStack core services can be cooperative by default using Self* and P2P mechanisms

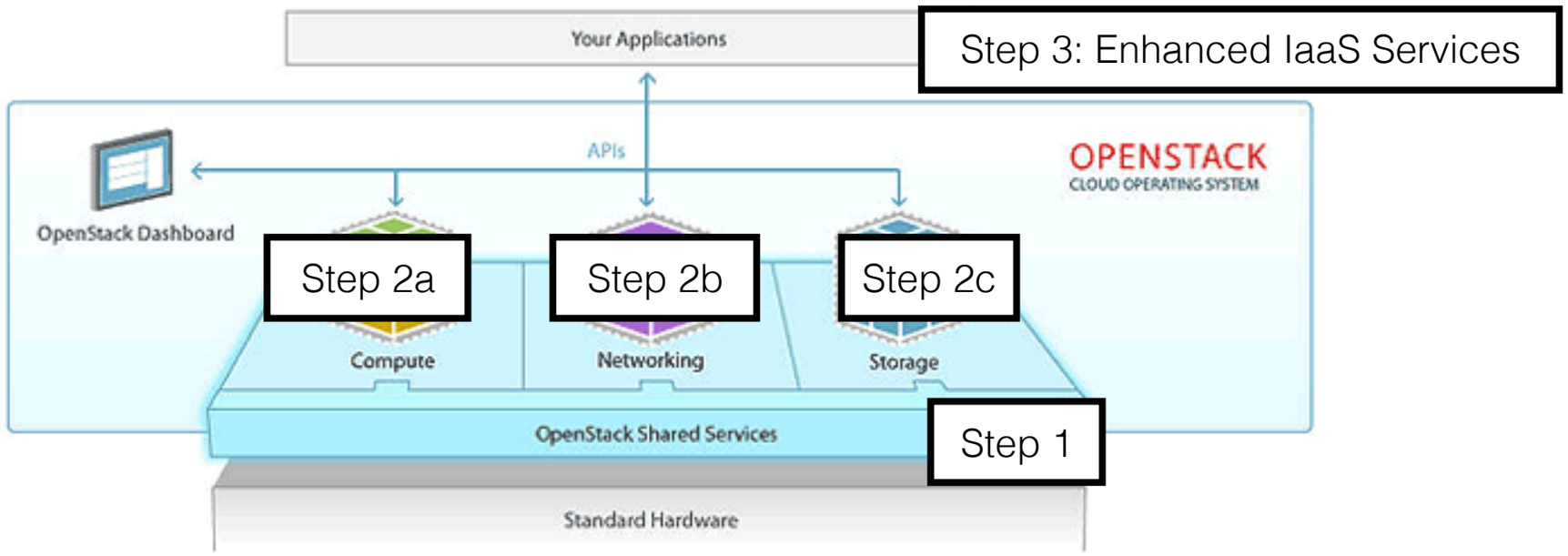


Natively distributed/cooperative

Would OpenStack be the solution?

- Do not reinvent the wheel… it is too late
- Few proposals to federate/operate distinct OpenStack DCs

Bottom/Up - investigate whether/how OpenStack core services can be cooperative by default using Self* and P2P mechanisms



Looking back to the Future

- Austin Summit - May 2016 - Nova PoC

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.

Looking back to the Future

Technical considerations

SIDEBAR ◂ PREV | 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.

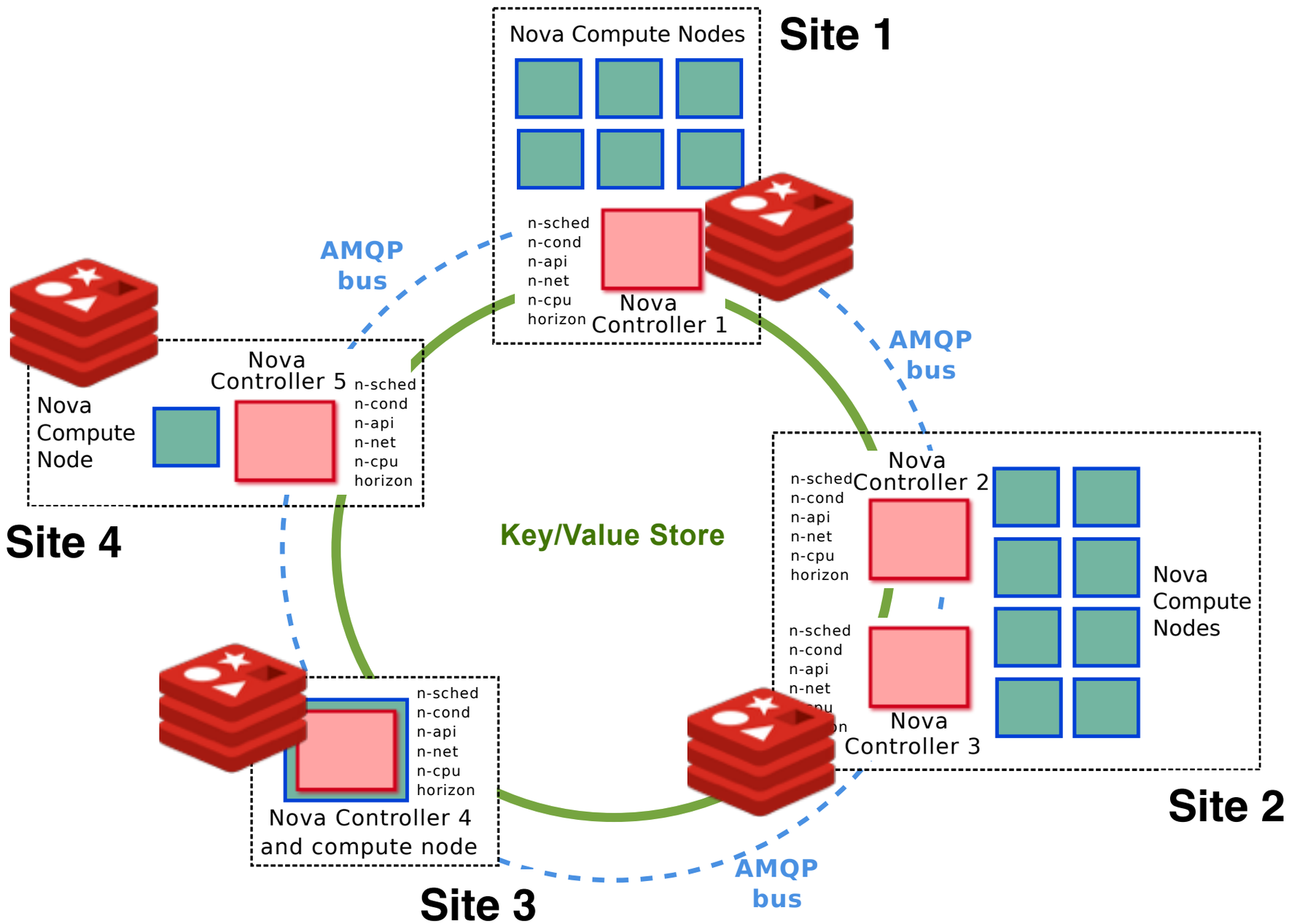
Infrastructure segregation

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.

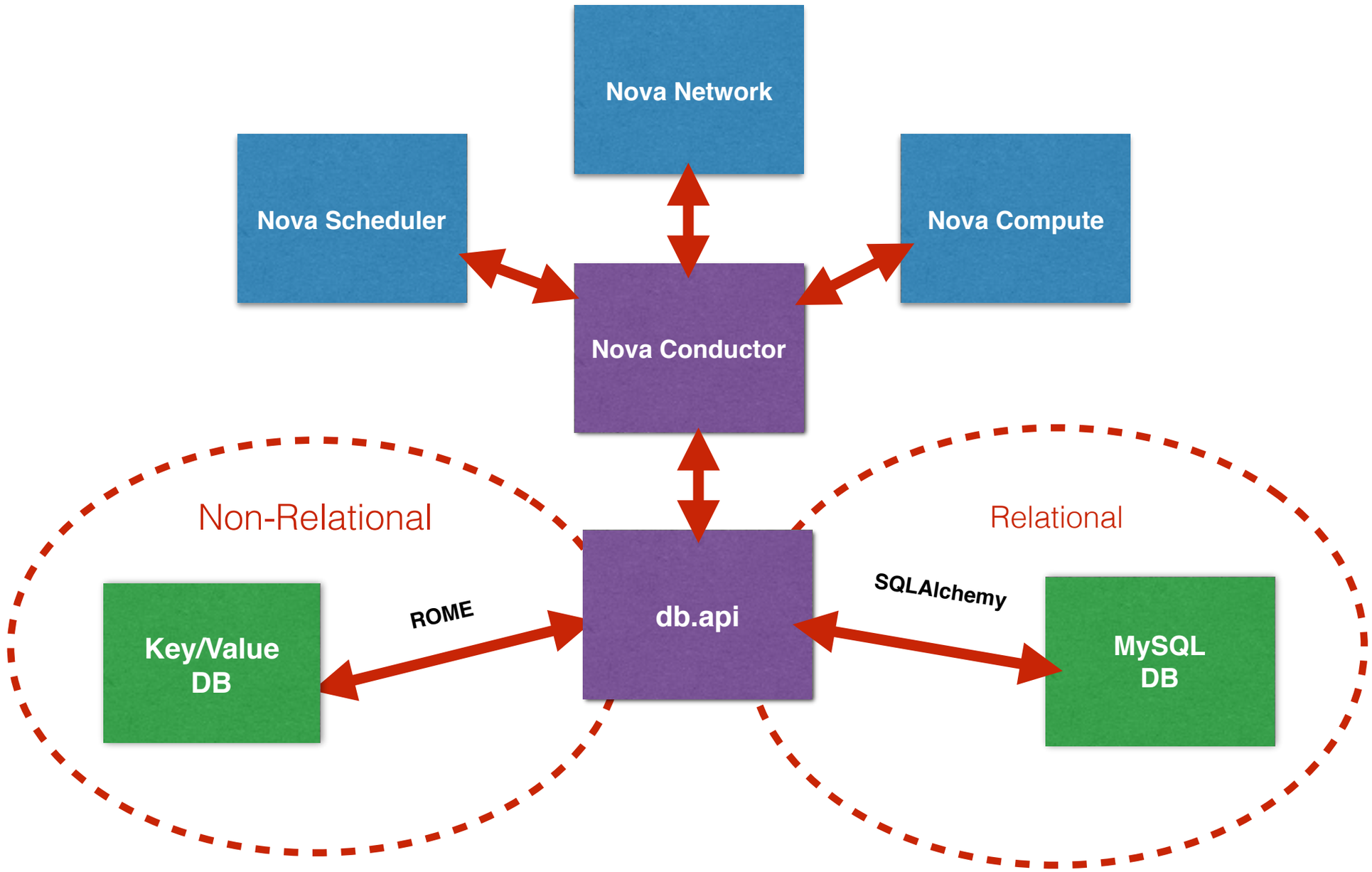
Traditional clustering techniques typically provide high availability and some additional scale for these environments. In the quest for massive scale, however, you must take additional steps to relieve the performance pressure on these components in order to prevent them from negatively impacting the 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.

Looking back to the Future

- Austin Summit - May 2016 - Nova PoC (based on Juno)
  Replaced MySQL DB by Reddis (NoSQL backend)



Leveraging a Key/Value Store DB

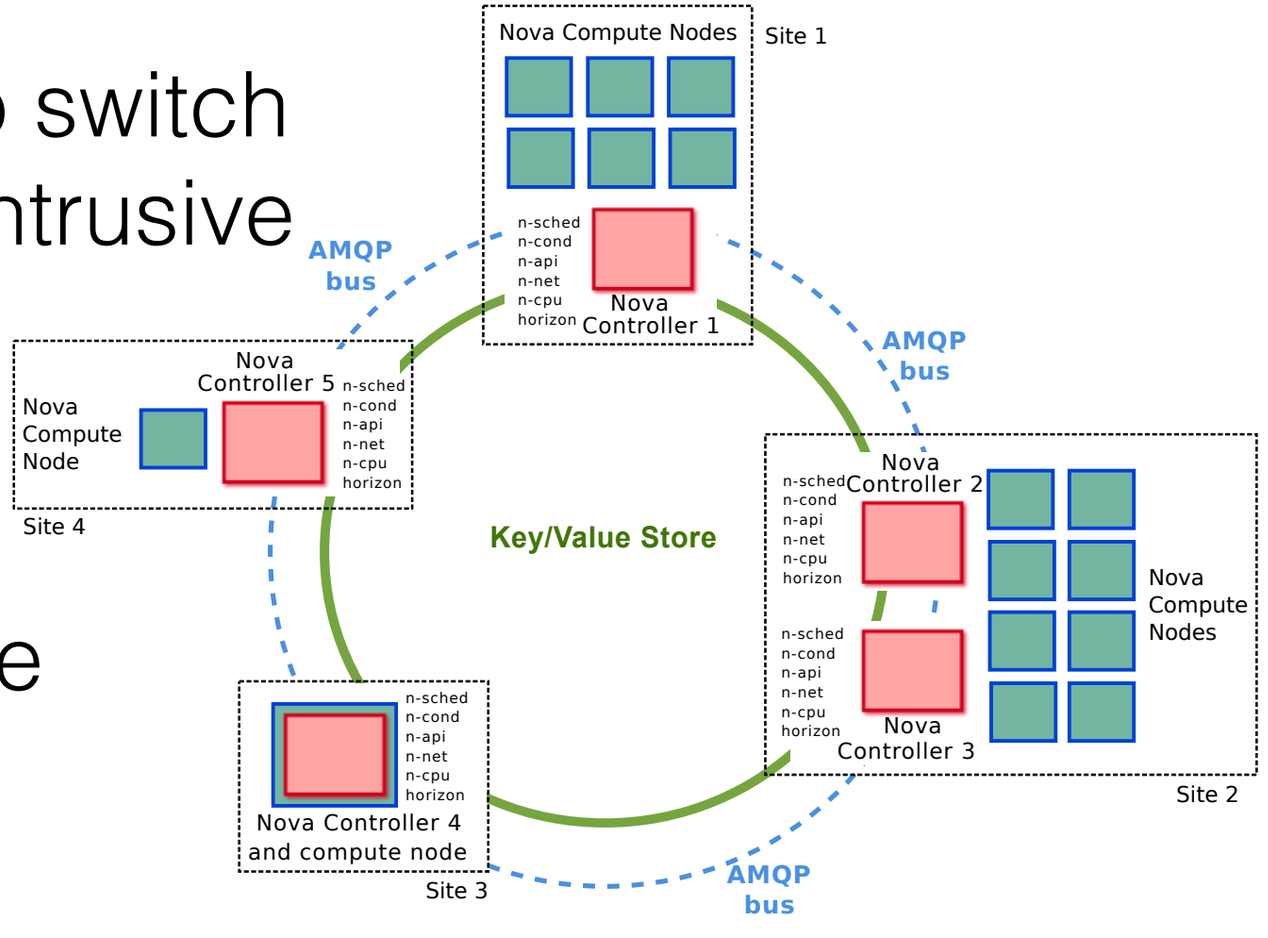


Nova (compute service) - software architecture

ROME

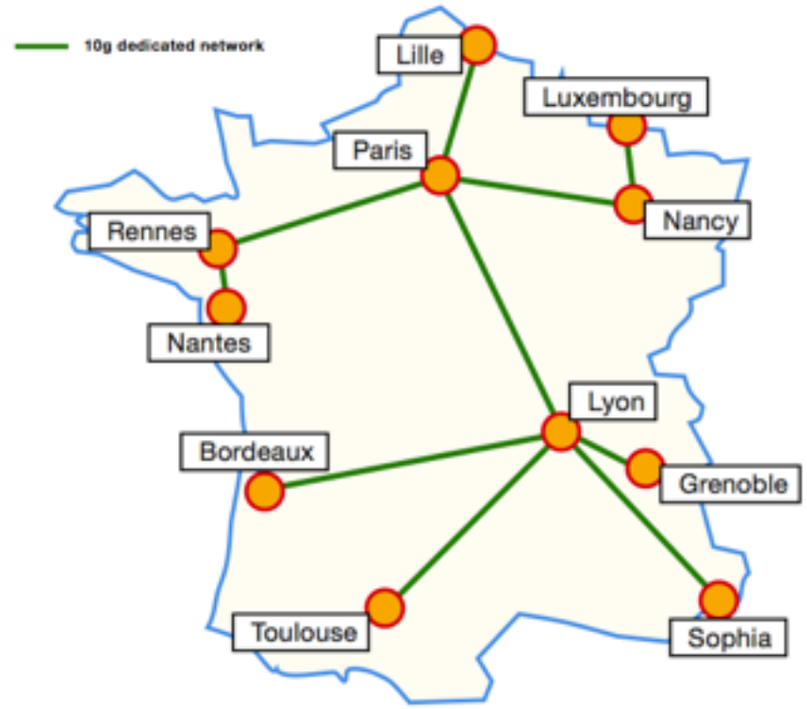
- Relational Object Mapping Extension for key/value stores
  Jonathan Pastor's Phd
  https://github.com/BeyondTheClouds/rome
- Enables the query of Key/Value Store DB with the same interface as SQLAlchemy

- 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



Experiments

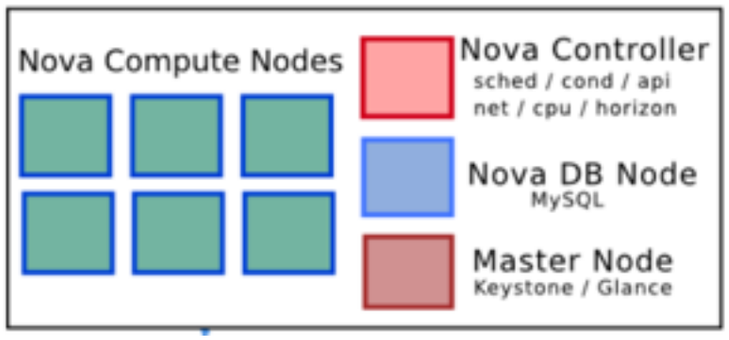
- Experiments have been conducted on Grid'5000
- Mono-site experiments
  ⟹ Evaluate the overhead of using ROME/Redis and the network impact.
- Multi-site experiments
  ⟹ Determine the impact of latency.
  ⟹ Validate compatibility with higher level mechanisms validation



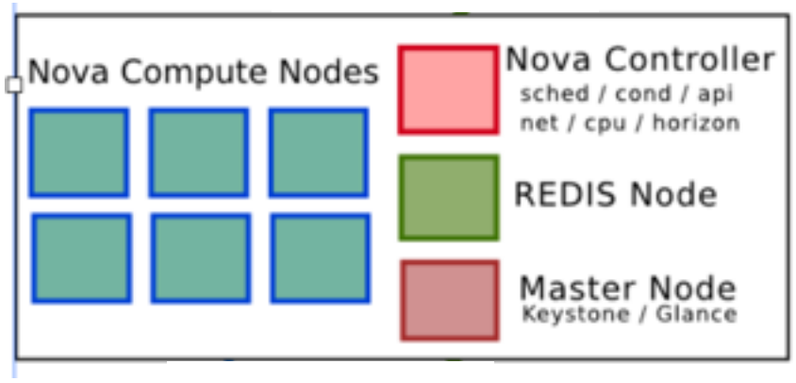
www.grid5000.fr
1500 servers, spread across 10 sites
Full admin rights

Mono-Site Experiments

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



MySQL/SQLAlchemy

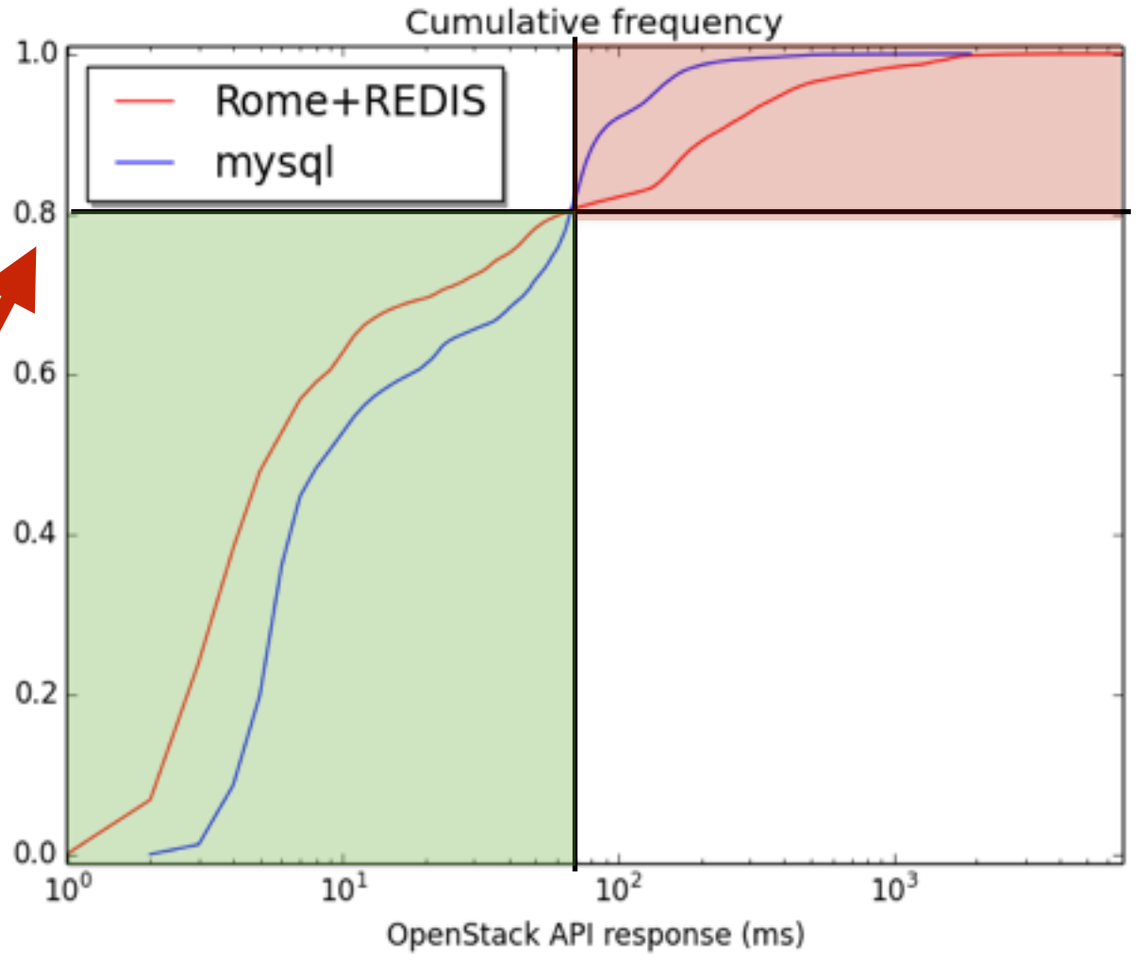


ROME/Redis

Mono-Site Experiments

- Evaluate the overhead of using ROME/Redis
- ROME 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

Mono-Site Experiments

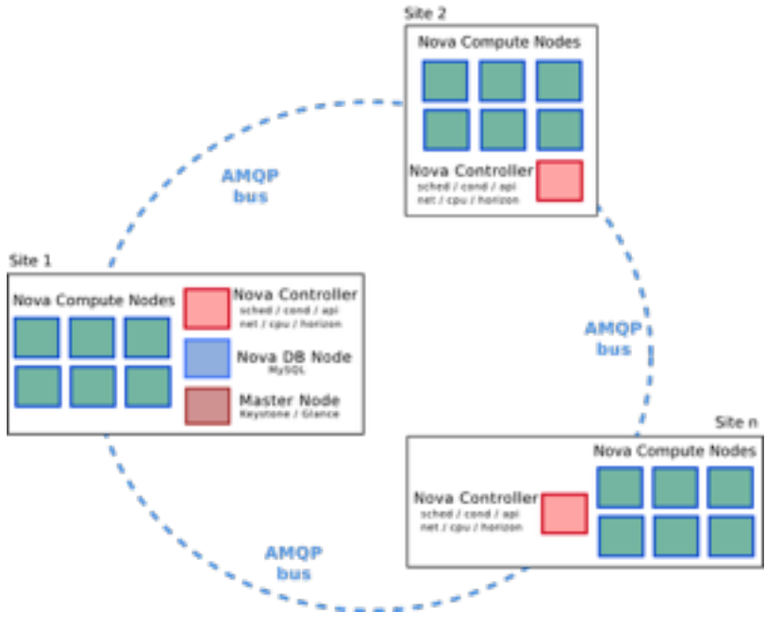
- Evaluate the overhead of using ROME/Redis
- ROME stores objects in a JSON format: *serialization/deserialization cost*
- ROME reimplements some mechanisms: *join, transaction/session, …*

Table 2: Time used to create 500 VMs on a single cluster configuration (in sec.)

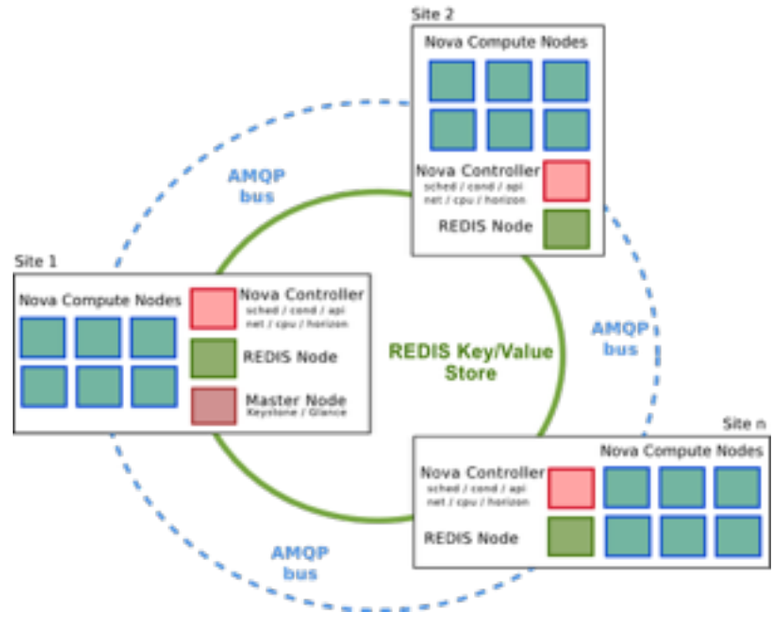
Backend configuration	REDIS	MySQL
1 node	322	298
4 nodes	327	-
4 nodes + repl	413	-

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
- Galera experiments have been performed but due to reproducible issues with more than 4 sites, results are not satisfactory enough to be discussed (RR available on demand)



SQLAlchemy+MySQL



ROME+Redis

Multi-Site Experiments

Table 3: Time used to create 500 VMs with a 10ms inter-site latency (in sec.).

Nb of locations	REDIS	MySQL
2 clusters	271	209
4 clusters	263	139
6 clusters	229	123
8 clusters	223	422

Table 4: Time used to create 500 VMs inter-site latency (in sec.).

Nb of locations	REDIS	MySQL
2 clusters	723	268
4 clusters	427	203
6 clusters	341	184
8 clusters	302	759

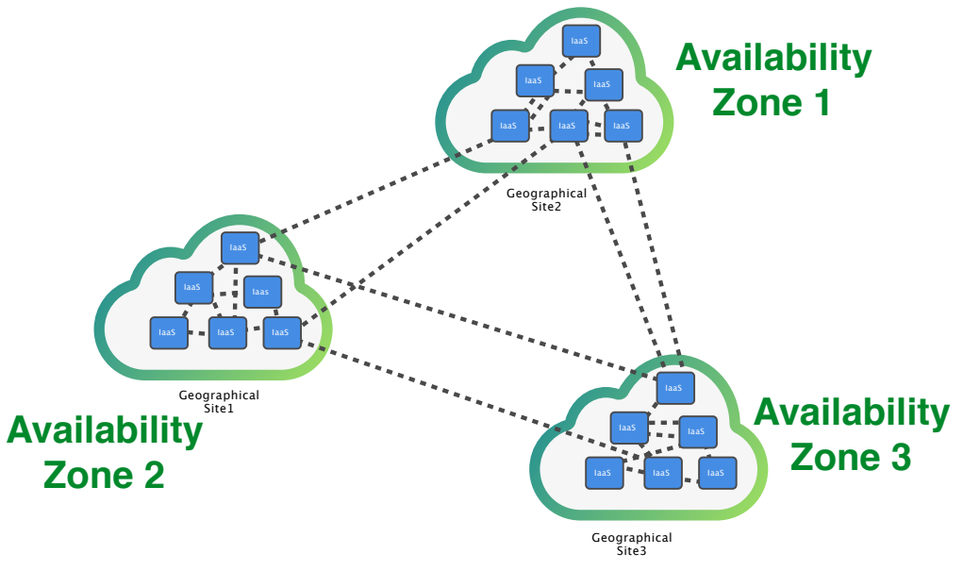
SQL scalability bottleneck

(one SQL server for the whole infrastructure)

Increasing the nb of nodes leads to better reactivity
From 8 clusters, MySQL becomes a bottleneck

Compatibility with Higher Level Features

- Asses the usage of advanced OpenStack feature:
  host-aggregates / availability zones
- As we targeted a low-level component, ROME is compatible with most of the existing features.
- Performance is not impacted (same order of magnitude)
- VM Repartition is correctly achieved
  (without availability zones the distribution was respectively 26%, 20%, 22%, 32% of the created VMs for a 4 clusters experiments).

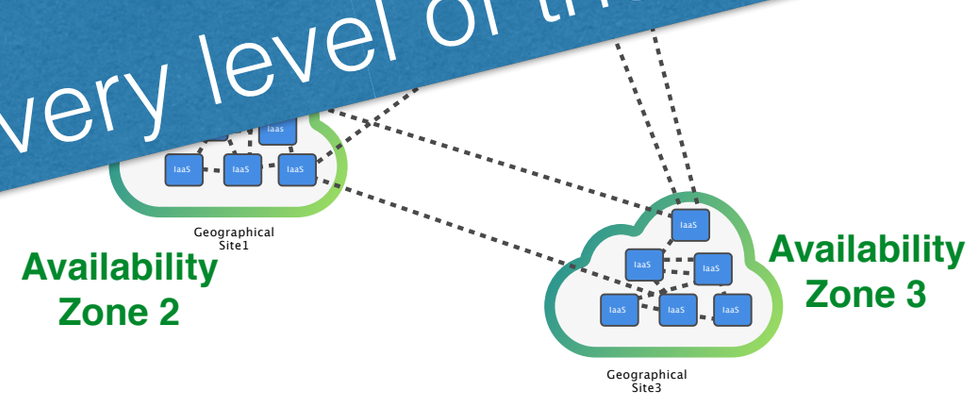


Can we go beyond a research POC ?

Compatibility with Higher Level Features

- Asses the usage of advanced OpenStack feature:
  host-aggregates / availability zones
- As we targeted a low-level component, ROME is compatible with most of the existing features.
- Performance is not impacted (same order of magni
- VM Repartition is correctly achieved
  (without availability zones
  20%, 22%, 32%

Interesting but …just the top of the Iceberg !
Glance, Neutron, Cinder…?
Scalability of the AMQP bus?
HA?
Reify locality aspects at every level of the stack?



Can we go beyond a research POC ?

Takeaway message

- Goal of the WG: do not reinvent the wheel (upstream first).

  Study to what extent current mechanisms can handle Fog/Edge infrastructures

  Propose revisions/extensions of internal mechanisms when appropriate.

  Investigate how should current cloud APIs be extended to take the advantage of the geo-distribution (latency-aware applications…)

- Ongoing action: Analyze OpenStack Performance under the Fog/Edge perspective (scalability, traffic characterisation…) using EnOS (a dedicated framework for conducting performance evaluations of OpenStack)

EnOS - https://github.com/BeyondTheClouds/enos

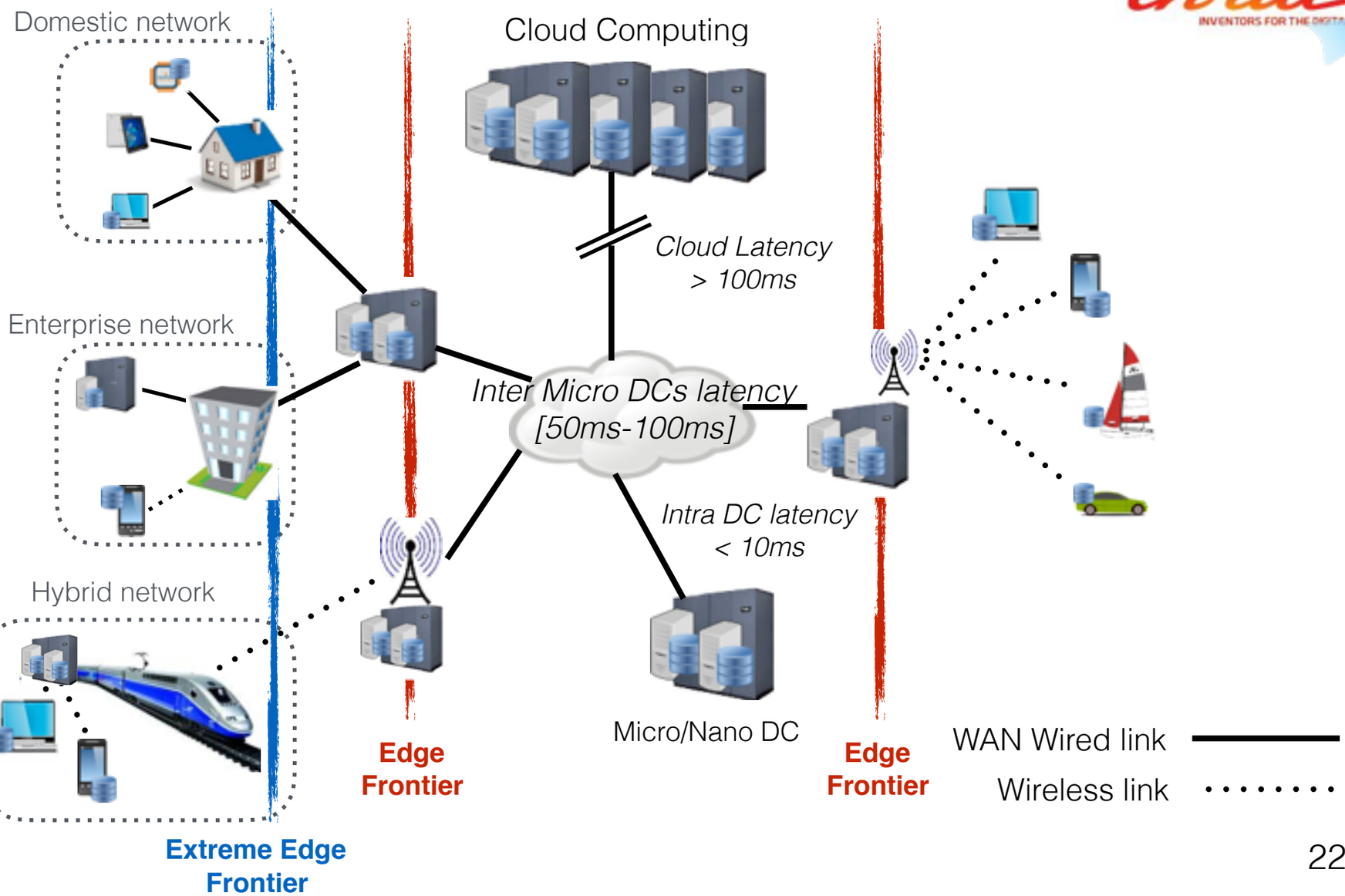
OpenStack on the Edge
BoF Session - Boston 2017

Cockroach Labs

Adrien Lebre
Inria
The Discovery Initiative
Fog Computing / Edge Computing/ Massively Distributed Clouds Working Group

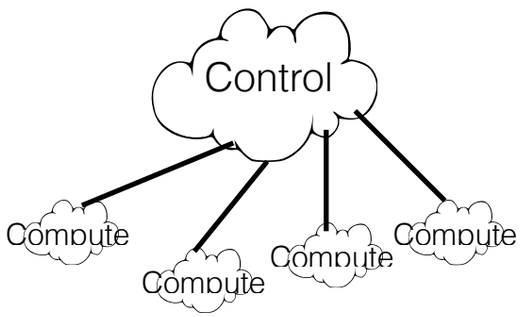
Ongoing Action

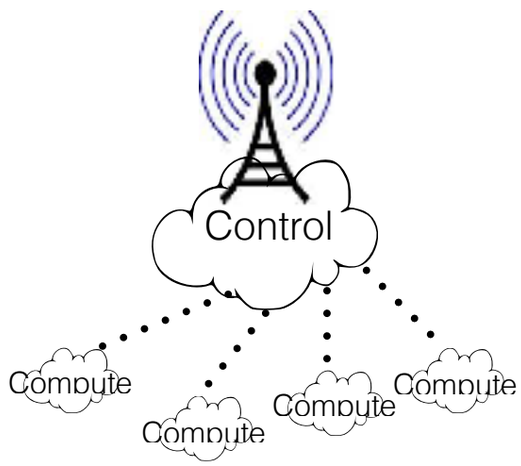
- Collaboration with the Performance Team to understand OpenStack Performance (scalability, traffic characterisation…)
- EnOS: Experimental Environment for Conducting performance evaluations of OpenStack

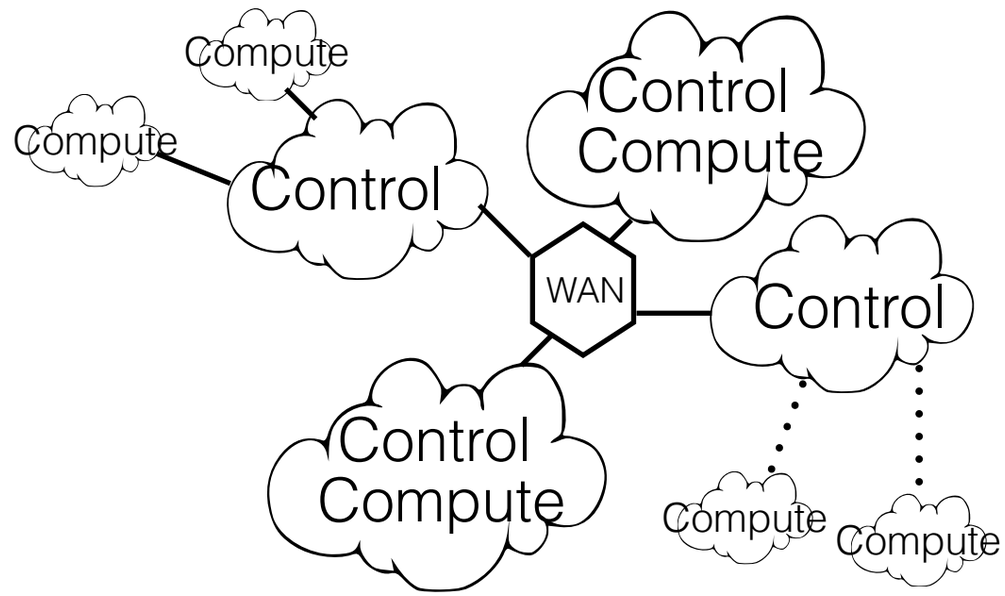


Ongoing Action

- Ongoing collaboration with the Performance Team to understand OpenStack Performance (scalability, traffic characterisation…)
- EnOS: Experimental Environment for Conducting performance evaluations of OpenStack
- Current focus: placement constraints/opportunities
  how many instances of each service? one global bus? one central Glance? severals? Where should we locate them?…







WAN Wired link ——————

Wireless link · · · · · · · · ·