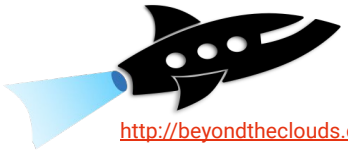
Collaborative On-Demand OpenStack Clouds and Beyond

— Collaboration by composition with the scope-lang, IPL Discovery

Inria



http://beyondtheclouds.github.io

Managing Resources of an Edge Infrastructure?

— What does it mean?

Edge Infra?

*A kind of **Distributed Cloud Infrastructure***

Particularities

- 100s/1000s of locations (*i.e.*, Data Centers)
- Dozen of servers per Data Center
- WAN links (10 to 300 ms RTT)

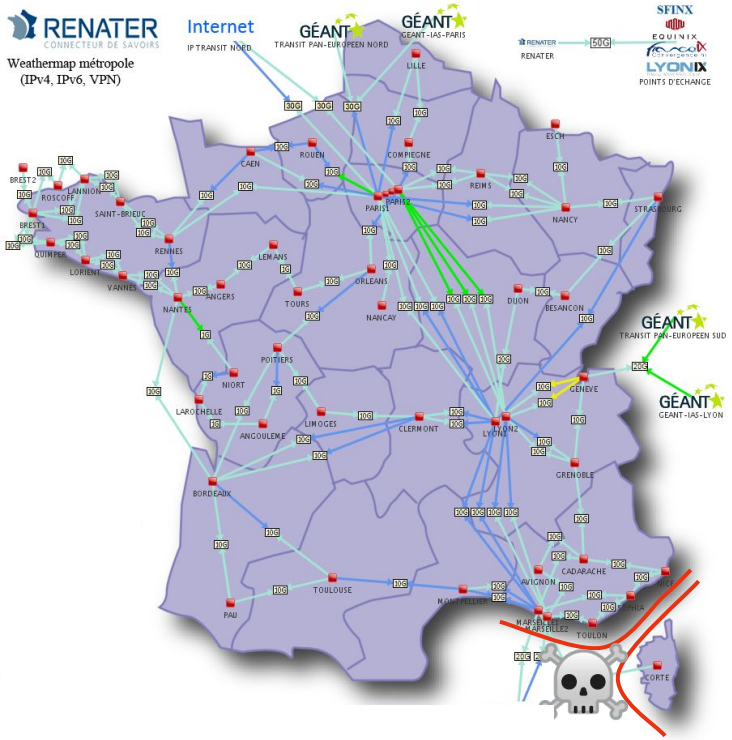
- Intermittent connectivity
- Network partitioning issues

Example of an Edge Infrastructure

Renater backbone

- Point of Presence (**PoP**) in red
- **Micro DC** in each **PoP**
  - Dozen of servers
- **WAN links** interconnect **PoPs**
  - `10ms`, Paris ↔ Marseille
  - `300ms`, Paris ↔ Vancouver

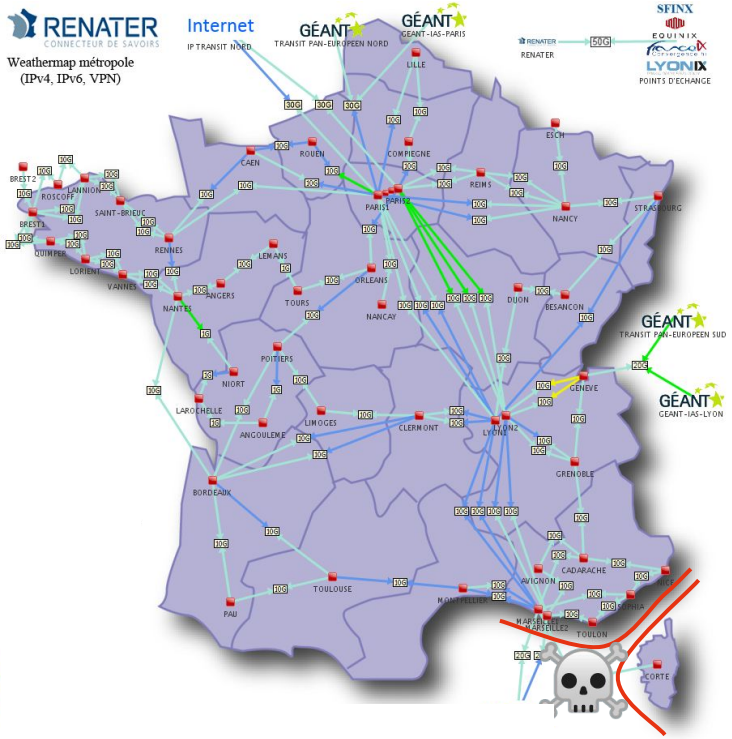
- **Net. partitions risks** (☠) between **PoPs**
  - Marseille/Corte



Managing Resources of an Edge Infra?

__Same__ as in Cloud Computing. __Tuned__ for the Edge.[‡]

1. Operate/use a **single DC**
   - Manage users, flavors, quotas
   - Provision computes, storages, nets
2. Operate/use **several DCs**
   - **Cross-DC** collaborative provisioning (intra/inter services)
   - Manage **multiple** DC **simultaneously**
3. Robustness w.r.t. network delay & **disconnections**
   - Access/Manage reachable resources (full isolation)



‡. https://www.openstack.org/videos/summits/vancouver-2018/can-we-operate-and-use-an-edge-computing-infrastructure-with-openstack

Managing Resources of an Edge Infra with OpenStack

— Review past and ongoing actions.

Red Thread: Boot of a VM

```
openstack server create my-vm --image Debian
```

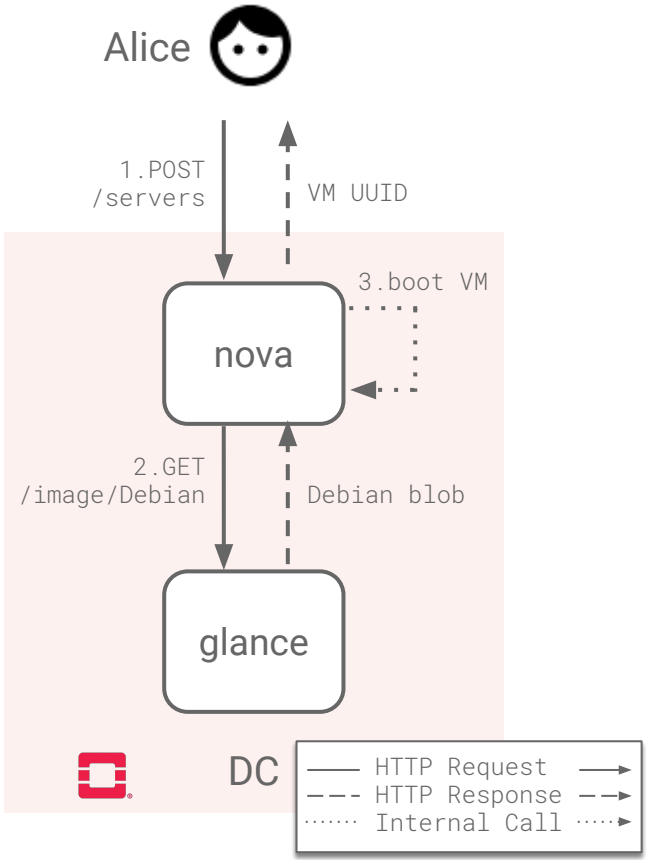
Boot of a **Debian VM** (simplified)

1. Operator requests a `boot` to nova
2. Nova contacts glance to get Debian
3. Nova boots VM internally

Boot VM **scenarios**

- in a single DC (**1-DC**)
- in one DC with an image from another DC (**x-DC**)
- in multiple DC (***-DC**)
- **Globally** vs. **partially** connected infra.

Boot VM	1-DC	x-DC	*-DC
global	??	??	??
partial	??	??	??



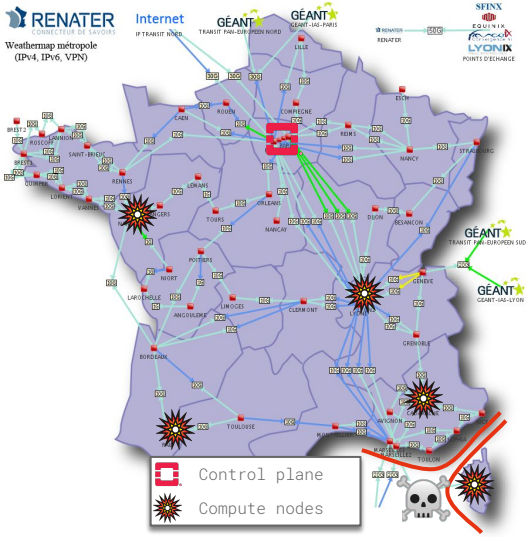
Approach 1: Centralized Management

One DC** hosts the **control plane**; **Other DCs** host **compute nodes

Theoretically, a normal OpenStack deployment

Practically, a lot of issues/challenges‡ †

- Impact of latency, throughput, intermittent connectivity
- What are the deployment rules for each service?
- Deployment/Upgrade of the system



Boot VM	1-DC	x-DC	*-DC
global	✓	-	-
partial	✗	✗	✗

→ **Operational, but focuses on specific use-cases**

‡. https://www.openstack.org/videos/summits/boston-2017/toward-fog-edge-and-nfv-deployments-evaluating-openstack-wanwide
†. https://www.openstack.org/videos/summits/Paris-2018/rabbitmq-or-qpid-dispatch-router-pushing-openstack-to-the-edge

Approach 2: Distributed Management

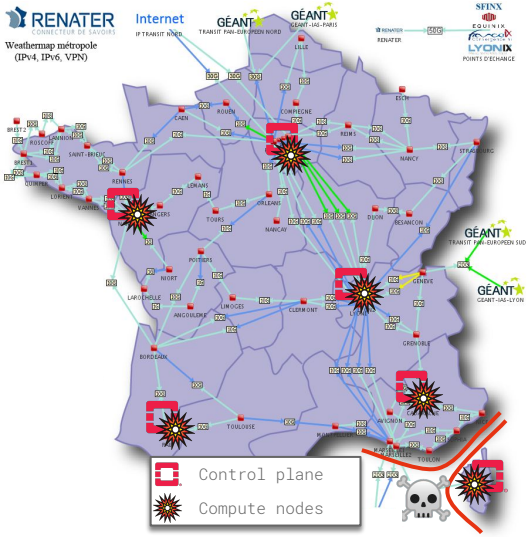
***Every DC** is one OpenStack that **collaborates** with others (à la p2p)*

Theoretically, should fulfill our needs

- One **autonomous** control plane per DC (*partial ✓*)
- DCs are **collaborative** with each other
  - Share resources with others (*benefits from natural sharding*)
  - Replicate resources at some locations (*preserve from delay/partial*)

Practically, a sophisticated solution

- Implementing collaboration is a **conundrum**
- *OpenStack doesn't provide a general solution*



Boot VM	1-DC	x-DC	*-DC
global	✓	✓	✓
partial	✓	✓	✓

Theoretical

Service-to-Service Collaboration

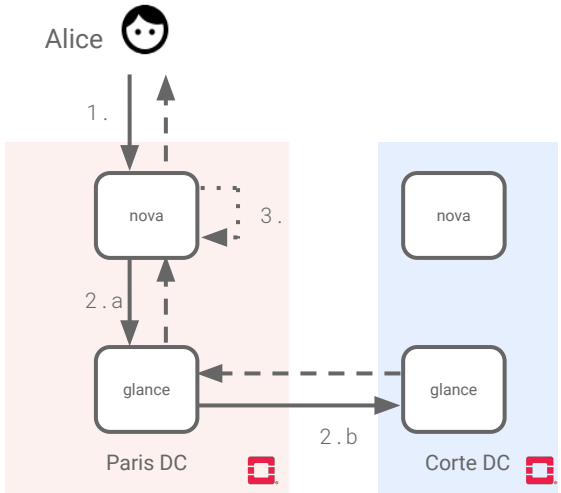
*Make the **service natively collaborative** (K2K‡, Glance to Glance†)*

Pro

- Efficient/Optimal implementation (optimistically)

Issues

- **Tangle sophisticated collaboration** code with **vanilla** code
  - Force core developers to maintain collaboration code, make new features collaborative
  - Intrusive collaboration is not an option for some services (not everyone wants to do edge or need collaboration)



Boot VM	1-DC	x-DC	*-DC
global	✓	✓?	✓?
partial	✓?	✓?	✓?

→ Collaboration code should be decoupled from vanilla code

‡. https://docs.openstack.org/security-guide/identity/federated-keystone.html
†. https://wiki.openstack.org/wiki/Image_handling_in_edge_environment

Broker Collaboration

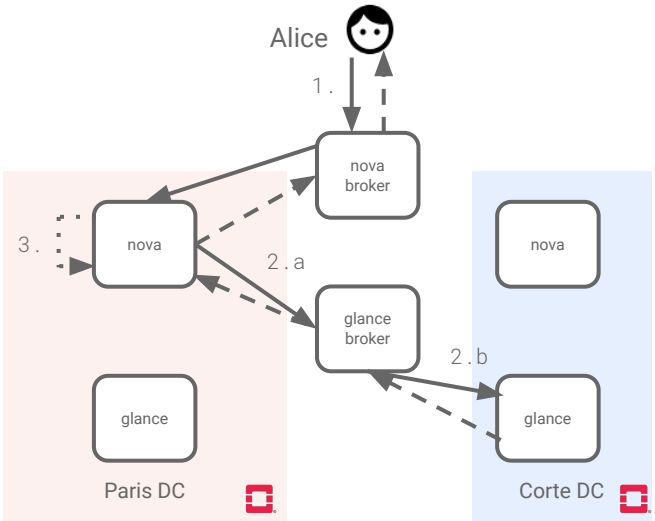
***Broker** on top **orchestrates** the **collaboration** (Tricircle‡, Mixmatch†, ...)*

Pro

- Put **collaboration code outside** of vanilla code (in the broker)
- Enable **enhancement of APIs** for sharing/replication

Issues

- Current implementations
  - Rely on a central broken (partial: ✗)
  - Miss mechanism for replication (*-DC: ✗)
- **Broker** has to be **exhaustive** with the underlying **APIs**
  - **Lot of code** to simply **expose APIs** at **broker level**



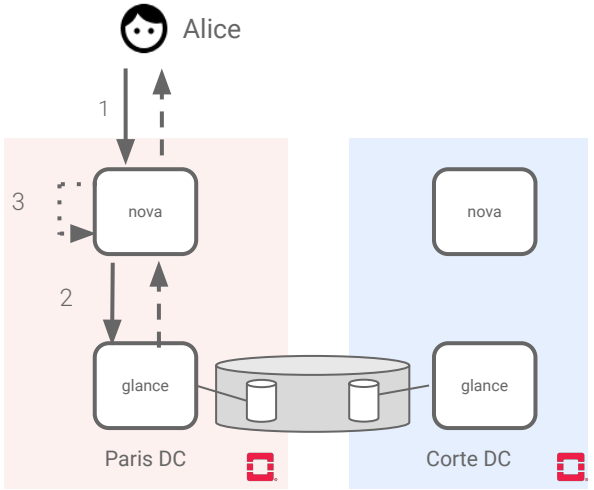
Boot VM	1-DC	x-DC	*-DC
global	✓	✓	✗
partial	✗	✗	✗

→ Broker should not reimplement API to the risk of developing a new OpenStack on top of OpenStack

‡. https://wiki.openstack.org/wiki/Tricircle
†. https://mixmatch.readthedocs.io/en/latest/

DataBase Collaboration

*Implement **collaboration** by making **resources reachable** via the **DB** (Rome, active-active Galera, CockroachDB, …)*[‡] [†]



Pro

- Do not need to modify OpenStack code
- Allow data replication (mitigate delay/partial ✓)

Issues

- Galera: replicate data at **every location** does not scale
- Maintain **consistency** of **all data across all DCs** forbids writes in network partition (partial ✗)
- DataBase **only considers data**
  - A **resource** is made of **data and effects**
  - Collaboration via DB **misses effects** (x-DC/*-DC: Keystone ✓, Neutron ✗, ...)

Boot VM	1-DC	x-DC	*-DC
global	✓	✓, ✗	✓, ✗
partial	✗	✗	✗

→ Resources could not be global
→ Resources have to come with there side effects

‡. https://www.openstack.org/videos/summits/vancouver-2018/keystone-in-the-context-of-fogedge-massively-distributed-clouds
†. https://hal.archives-ouvertes.fr/tel-01416099v1

Collab. techno.	Issue	What we want
Service-to-service	**Tangled invasive** collab. code ➔ Not always an option	**Ad-hoc** collaboration ➔ Modular confined collab. code ➔ Plugged if need be
Broker	**Re-implement** API and logic at Broker ➔ Do the job twice ➔ Specific code for one app.	**Generic** collab. for **any app.** ➔ Don't do the job!
DataBase	**Replicate** data **everywhere** (Galera) ➔ Do not scale at Edge	**On-demand** replication ➔ Keep clusters small for scaling
	Maintain consistency **everywhere** ➔ Forbid partial writes	**On-demand** collaboration ➔ Keep clusters small for resiliency
	Omit effects ➔ Work for data only app.	**Include** effects ➔ Work for any app.

Draw Conclusions

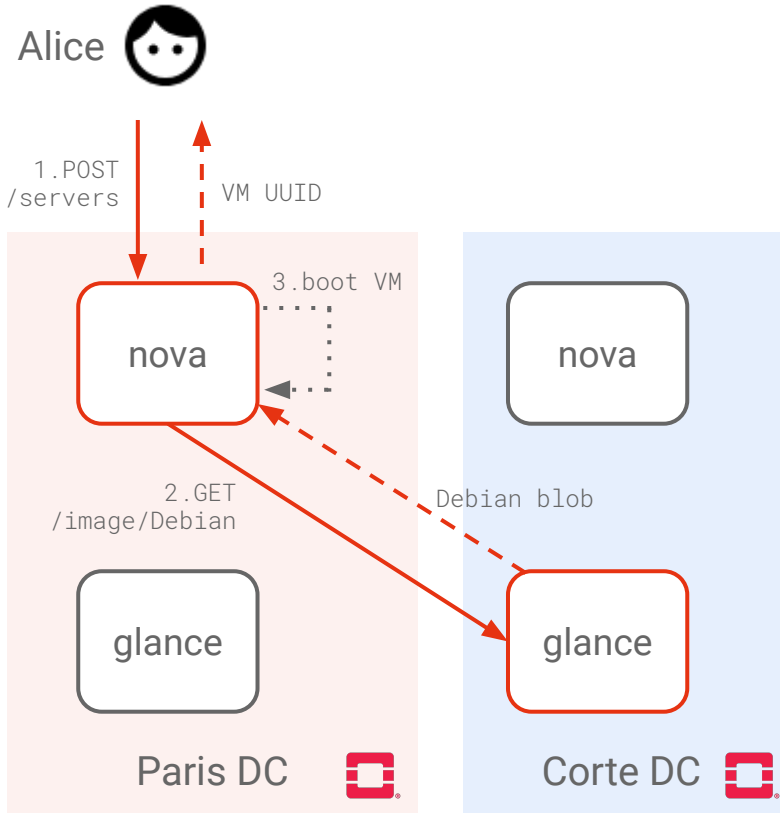
Distributed Management with the scope-lang

— Ad-hoc, generic, on-demand and effectful collaboration.

scope-lang

Alice defines the scope of the request into the CLI. The scope specifies **where the request applies**

```
openstack server create my-vm
   --image Debian
   --scope { nova:   Paris
           , glance: Corte }
```



Boot VM in Paris with Debian from Corte

scope-lang in Actions

Scope for **1-DC** operations

```
OS@Paris$ openstack server create my-vm --image Debian
    --scope {nova: Paris, glance: Paris}
```

Scope for **x-DC** operations

```
OS@Paris$ openstack server create my-vm --image Debian
    --scope {nova: Paris, glance: Corte}
```

Scope for ***-DC** operations

```
OS@Paris$ openstack server create my-vm --image Debian
    --scope {nova: Paris&Corte, glance: Corte}
```

*-DC: and ‘&’

Do the operation **here and there**

- Create a user in Paris and Corte

```
openstack user create Alice
  --password-prompt
  --scope {keystone: Paris&Corte}
```

- List VMs in Paris and Corte

```
openstack server list
  --scope {nova: Paris&Corte}
```

Properties

- **On-demand** partial replication
  - Replication at scope locations
  - Keep cluster small for scaling and resiliency
- Query **multiple** DCs at once

*-DC: or '|'

Do the operation **here or there**

- Boot a VM in Paris with image from Corte or Marseille

```
openstack server create my-vm
   --image Debian
   --scope { nova: Paris
           , glance: Corte|Marseille }
```

Properties

- Let the operator **implements retries workflow**

  No matter if one is down or don't have the image, till the other is up and has the image.

Manage	Needs	scope
Single DC	Manage resources **locally** • boot VM in Paris • List VMs in Corte	1-DC†‡ • {nova:Paris, glance:Paris} • {nova:Corte}
Multiple DCs	**Cross-DC** collaboration • boot VM in Paris with Debian from Corte	x-DC† • {nova:Paris, glance:Corte}
	Manage resources **simultaneously** • create image in Paris and Corte • boot VM in Paris with Debian from Corte or Marseille	*-DC • {glance:Paris&Corte} • {nova:Paris, glance:Corte|Marseille}

†. **PoC**: http://github.com/BeyondTheClouds/openstackoid
‡. Unnecessary; scope is implicitly local

Managing Resources of an Edge Infrastructure with the scope-lang

How to Implement the scope-lang?

— Collaboration by composition

Collaboration by Composition Principle

Service is an **abstraction** that

- **Encapsulates** the management of one or various **resources**
- Provides a **transparent access** to these resources
- Relies on **composition** to build an **app**.

Modularity says we can *interchange* one service by another one with the same abstraction

E.g., the OpenStack app.

- Nova/Glance web services **encapsulate** the management of **computes/images**
- Web services **REST API provide access** to these resources
- Web services **composition** builds the **manager**

Collaboration by Composition + scope-lang

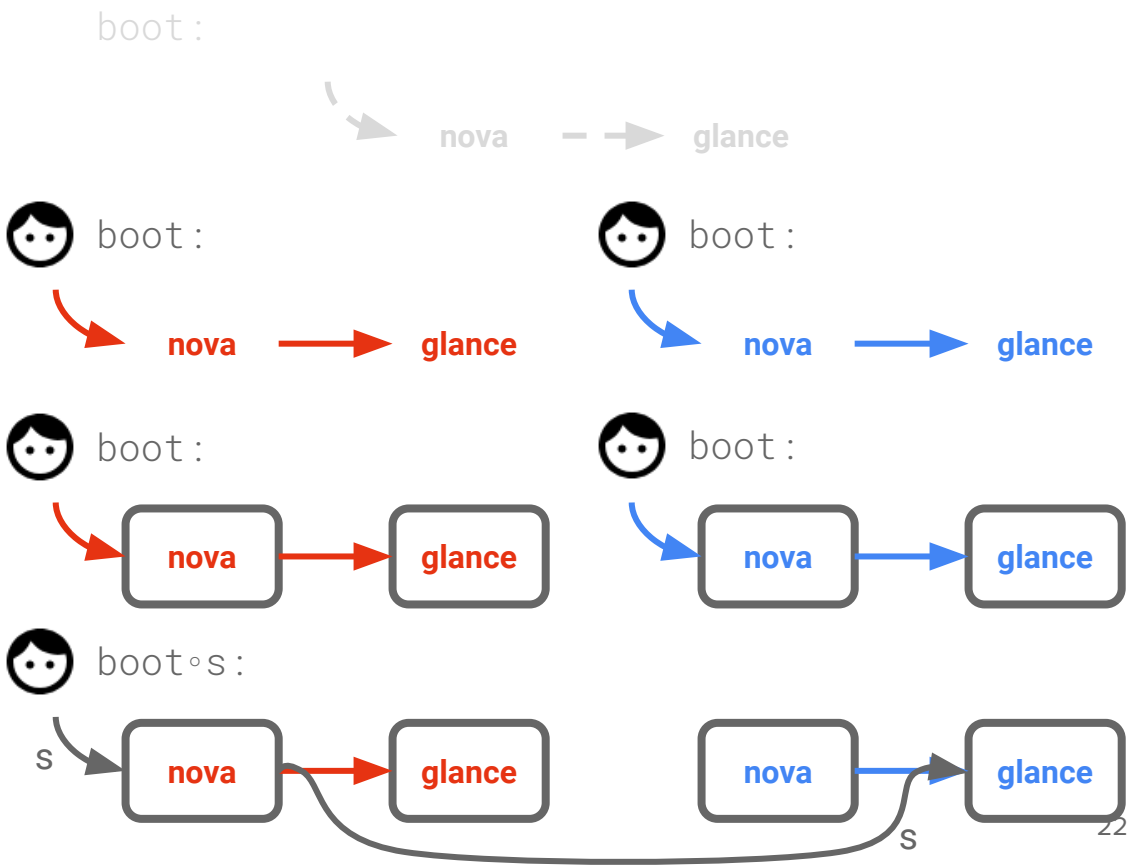
CLI operation: `boot`

Two DCs:

- Paris
- Corte

Wrappers intercept service calls. But, proceed by default.

A **scope** specifies the OS composition: `s = {nova: Paris, glance: Corte}`. In presence of a scope, wrappers forward service calls to the correct OS.



The Next 700 Brokers

The scope-lang wrapper/interpreter is

- **Generic**
  - Changing the composition does not require a code specific to an app.
- **Ad-hoc**
  - Changing the composition does not presume any intent of collaboration at start
- **Effectful**
  - Encapsulation property of services ensures to tracks effects
- **On-demand**
  - Define the scope of a collaboration

→ **Works for any app. built by service composition and under well modularization**

Limitations & Problems

Limited to **inter-service** collaborations

- Boot VM in Paris with image in Corte (Nova to Glance service compo. ✓)
- Live-migrate a VM from Paris to Corte (Nova to Nova intra-service operation ✗)

Risk of **bad collaborations** (x-DC)

- **Resources unreachability:** boot a VM in Paris with local network in Corte; it is not yet possible to extend network resources across DCs (API limitations, technical issues)
- **Local state:** verify in Keystone of Paris the Glance service token from Corte

Ph. D. thesis starts on Oct. 2019

- Formally define the semantic of the scope-lang (&, |, …)
- Statically reject bad collaborations

Wrap Up

Takeaway

- **Collaboration** between Edge should be done **on-demand** (and only if needed)
  - Cope with thousands of independent sites
  - Deal with Intermittent connections

- **scope-lang** for **on-demand** collaboration
  - 1-DC: `{nova:Paris, glance:Paris}`
  - x-DC: `{nova:Paris, glance:Corte}`
  - *-DC: `{nova:Paris&Corte}`
  - x/*-DC: `{nova:Paris&Corte, glance:Corte|Marseille}`

- Harness **service abstraction** for a **collaboration by composition**
  - Generic, Ad-hoc and effectfull approach
  - **Tested** on OpenStack‡. **Should work** for K8S, ShareLaTeX, ...

‡. **PoC**: http://github.com/BeyondTheClouds/openstackoid