INDIGO - DataCloud

INFN Istituto Nazionale di Fisica Nucleare

Lisa Zangrando

INFN Padova

Synergy

a new approach for optimizing the resource usage in OpenStack

Overview

Synergy

cloud service developed in the context of the INDIGO-DataCloud European project which aims to develop a new cloud software platform for the scientific community

- https://www.indigo-datacloud.eu/

Main objective

enable a more effective and flexible resource allocation and utilization in open clouds such as OpenStack

The issue

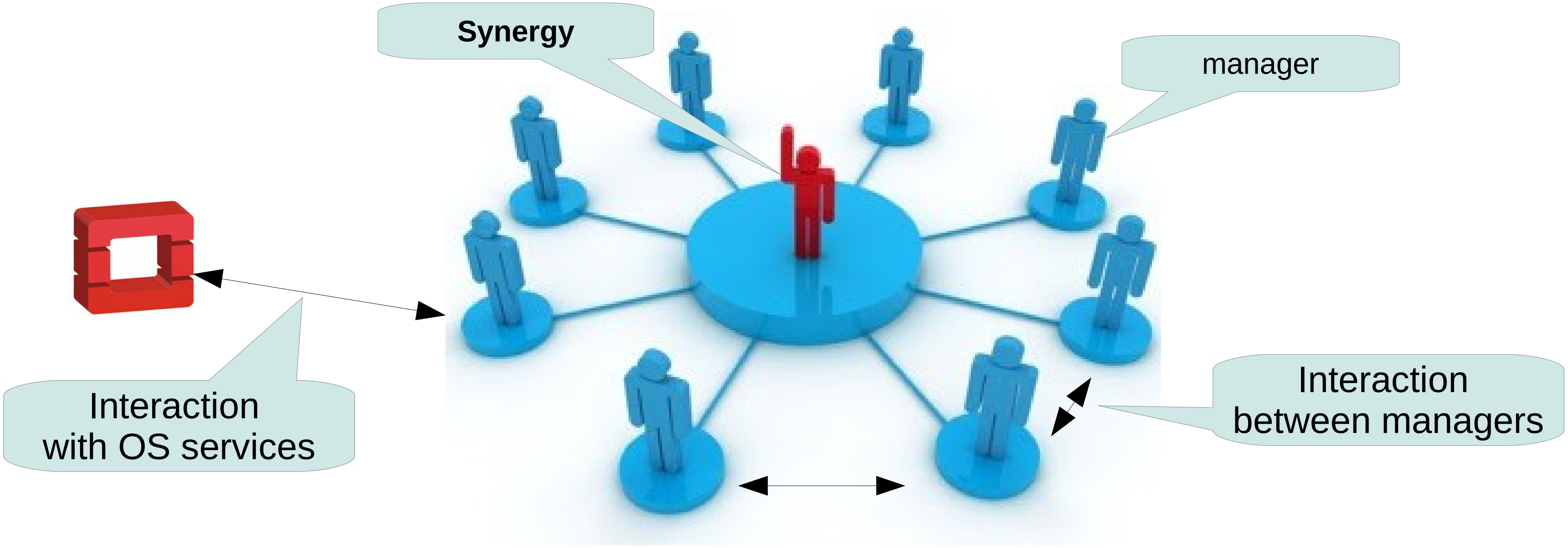
- **In the current OpenStack model:**

- resource allocation model: static partitioning
  - based on granted and fixed quotas (one per project)
  - the quotas cannot be exceeded
  - the quotas cannot be shared among projects
- scheduler too simple
  - based on the immediate First Come First Served (FCFS)
  - user requests are rejected if not immediately satisfied

- data center: very low global efficiency and increased cost
- 20 years old problem we solved by adopting batch systems
  - enhancement of our data center resources utilization from <50 to 100%
- **INDIGO addresses this issue through Synergy**

Synergy

- It is a cloud service designed for executing tasks in OpenStack
- It is composed by a collection of specific and independent pluggable functionality (managers) executed periodically or interactively through a RESTful API



Synergy

manager

Interaction with OS services

Interaction between managers

The manager interface

Any new manager can be easily implemented by extending a Synergy python abstract base class “Manager”:

```
class Manager(Thread):
  def getName(self): #returns the manager name
  def getStatus(self): #returns the manager status
  def isAutoStart(self): #is AutoStart enabled or disabled?
  def setup(self): #allows custom initialization
  def destroy(self): #invoked before destroying
  def execute(self, cmd): #executes user command synchronously
  def task(self): #executed periodically at fixed rate
```

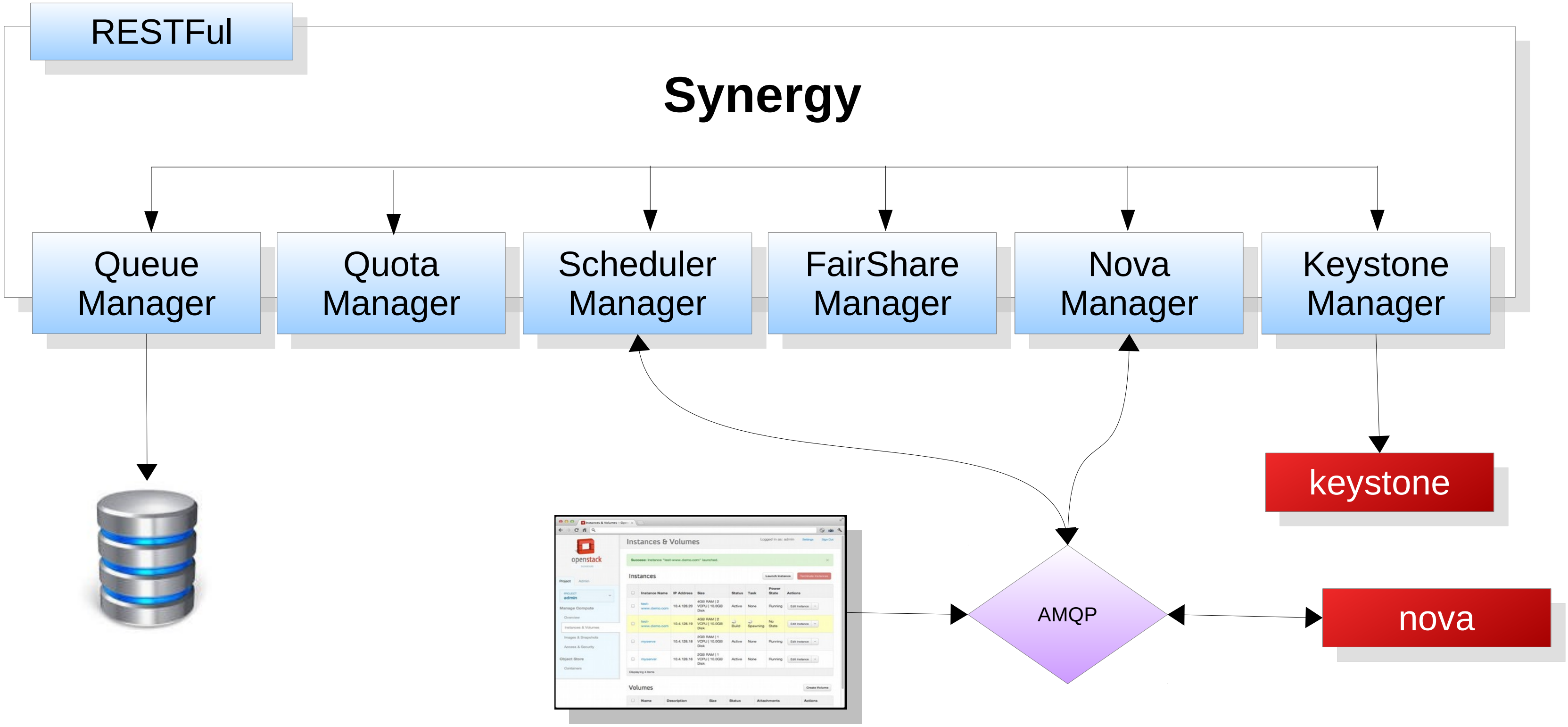
synchronous and asynchronous activities are respectively implemented by the last two methods: execute() and task().

How Synergy addresses the OS issues

- By implementing six specific managers which provide an advanced resource allocation and scheduling model

- **cloud resources can now be shared among different OpenStack projects**
  - overcomes the static partitioning limits
  - maximizes the resource utilization
- **shared resources are fairly distributed among users and projects**
  - user priority
  - project share
- **requests that can’t be immediately fulfilled are enqueued (not rejected!)**

Synergy scheduler managers



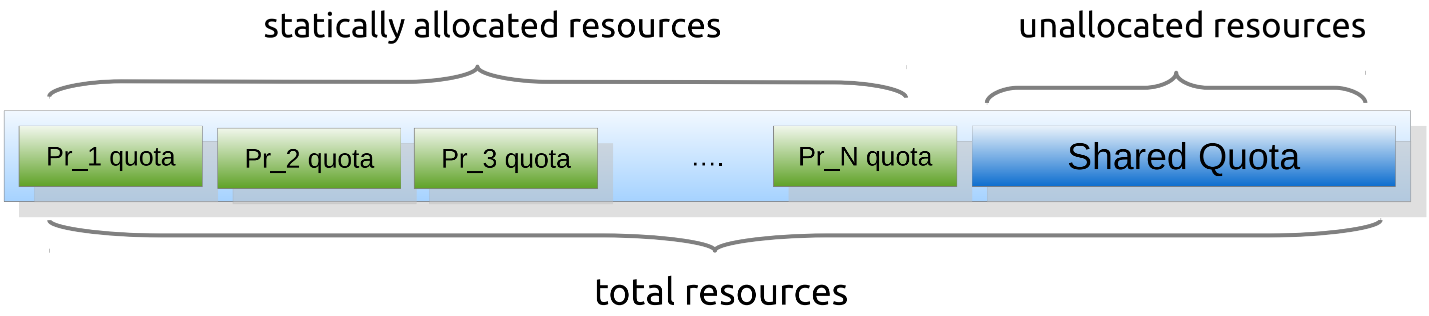
Resource allocation model

- With Synergy the OpenStack projects can now consume extra shared resources in addition to those statically assigned
- Projects can access to two quota kinds:

- **private quota:**
  - the standard (i.e. fixed and statically allocated) OpenStack quota
- **shared quota:**
  - extra resources shared among projects and handled by Synergy
  - its size can change dynamically: amount of resources not statically allocated
  - the user requests that cannot be immediately satisfied are inserted in a **persistent priority queue**

The Shared Quota

- The shared quota is a subset of the total resources not statically allocated
- its size is calculated as the difference between the total amount of cloud resources and the total resources statically allocated to the private quotas
- It is periodically calculated by Synergy



- Only the projects selected by the administrator can access to the shared quota beside to their own private quota

The scheduling model

- Fair-share algorithm: **SLURM Priority Multifactor**
  - https://slurm.schedmd.com/priority_multifactor.html
- shared resources fairly distributed among users according to specific fair-share policies defined by the administrator:

- **list of projects** allowed to access the shared quota
- **definition of shares** (%) on resource usages for the selected projects (e.g. project A=70%, project B=30%)
- **the maximum allowed lifetime** (e.g. 48 hours) of the relevant instances
  - VMs and Containers (instantiated via nova-docker)
  - this is needed to enforce the fair-sharing

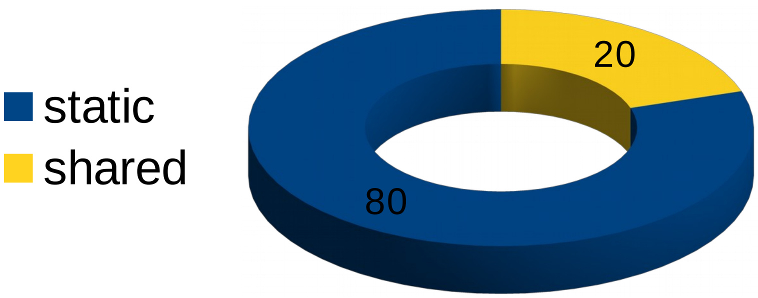
Remark

- **Synergy will not replace any existing OpenStack service (e.g Nova)**
  - it may complement their functionality as an independent service
- **no changes in the existing OpenStack components are required**
- **both resource allocation models coexist**

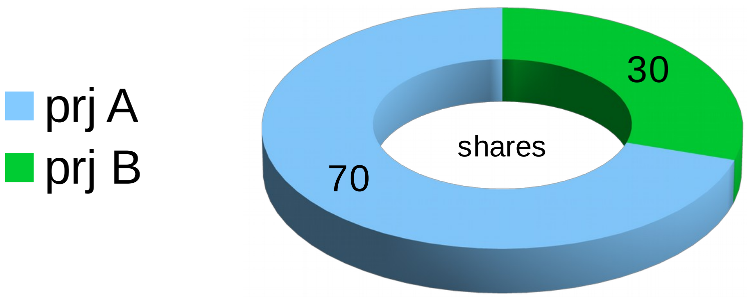
Testing setup

- Synergy was first deployed at INFN-Padova OpenStack production site of the EGI Federated Cloud
  - the goal: to test its behavior and stability under real usage conditions typical of a production environment
- EGI Fed Cloud infrastructure at INFN-Padova:
  - 1 controller and 6 compute nodes (centos7, Liberty)
  - total capacity: 144 VCPUs, 283 GB of RAM and 3.7 TB of block storage
- Resource allocation and the project's shares were defined as:

total resources



shared resources



Testing results

- **automatic robot** instantiates VMs at the same constant rate on both projects by using different users

- **testing session:** > 20,000 VMs executed over two days
  - Cirros images with different flavors
  - VM lifetime limited to 5 min to speed up testing

- **measured project resource usage:** as expected (70% and 30%) within 1%
  - user share tested in two configurations:
    - same share for all users
    - different share for each user: confirmed the expected limitation of the SLURM Multifactor algorithm, as documented in https://slurm.schedmd.com/fair_tree.html

- **tests coexisted and did not interfere/degrade the activities of other production projects/VOs** (not involved in fair-share computation)

The development status

- **Synergy released by INDIGO**
  - support for Liberty, Mitaka and Newton
  - next release: March 2017
- **Integrated in Launchpad and the OpenStack Continuous Integration system**
  - https://launchpad.net/synergy-service
  - https://launchpad.net/synergy-scheduler-manager
  - https://review.openstack.org
- **Code in GitHub**
  - https://github.com/openstack/synergy-service
  - https://github.com/openstack/synergy-scheduler-manager
- **Documentation**
  - https://indigo-dc.gitbooks.io/synergy/content

Next steps

- Implement a complete test suite
- test Synergy in the bigger CNRS's production site
- update Synergy for supporting the latest OpenStack versions
- improve the fair-share algorithm by implementing the SLURM Fair Tree
- improve the resource usage calculation by considering even CPU performance measured with HEPSPEC 2006 (HS06) benchmark (not only the CPU wall-clock time)
- **the ultimate goal is to have Synergy in the Official OpenStack distribution**

Questions?