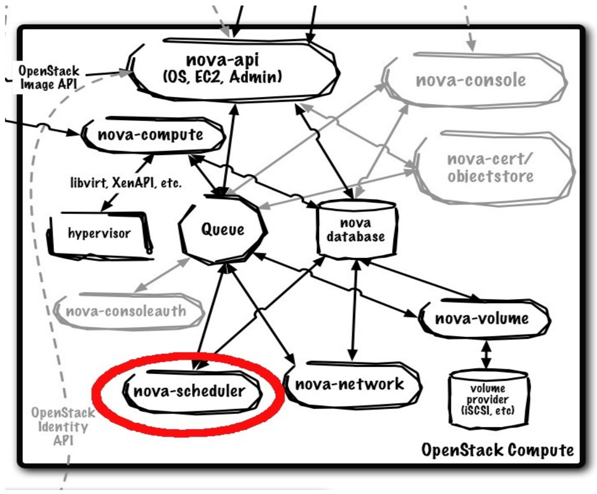


Lisa Zangrando, Eric Frizziero
INFN Padova

Fairshare based provisioning in OpenStack

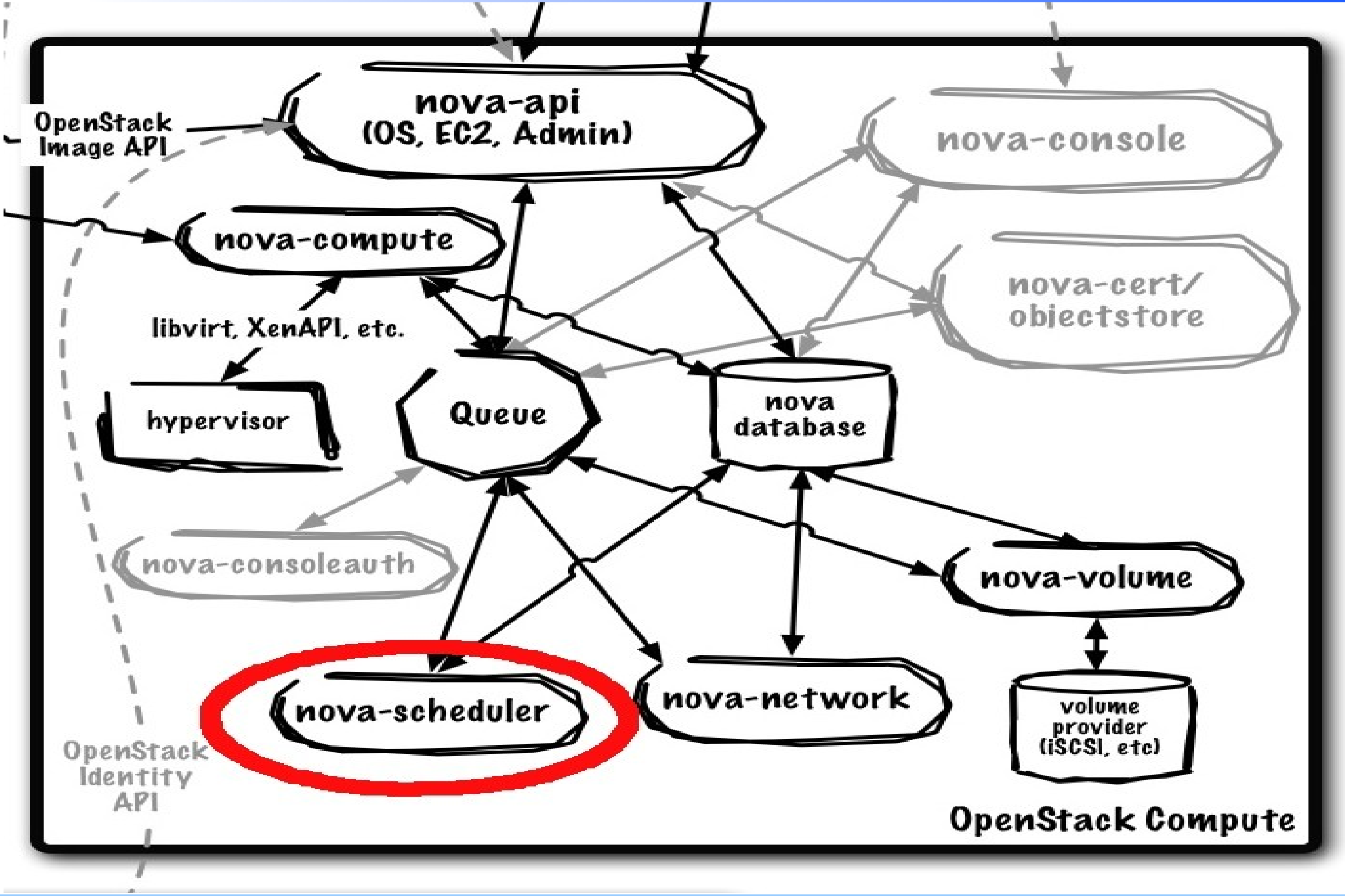
Drawbacks in OpenStack

- Nowadays in OpenStack
  - the user request fails (and immediately is forgotten) if no resource can satisfy it
  - static partitioning: the resource allocation to the user projects can be done only by granting fixed quotas
    - one project cannot exceed its own quota even if there are unused resources allocated to other projects
    - very low global efficiency and an increased cost in the Research Data Center's resource usage compared with alternative more flexible and dynamic approaches allowing a continuous full utilization of all available resources
- we need to find a better approach to enable a more effective and flexible resource allocation and utilization in OpenStack

Back to the past

- This resource allocation optimization problem has already been tackled and solved in the past initially by the batch systems and subsequently by the GRID model which was conceived on the awareness that Research Data Center resources are limited
- Advanced scheduling algorithms (i.e. fair-share) have been developed for batch systems
  - fair-share is a factor of the job's priority that affects the order in which user's queued jobs are scheduled to run
    - the priority is an integer and the larger the number, the higher the job will be positioned in the queue, and the sooner the job will be scheduled
    - historical resource utilization information is used for calculating the priority value
  - it guarantees the resources usage is equally distributed among users and groups by considering the portion of the resources allocated to them (i.e. share) and the resources already consumed
  - dynamic allocation allowing a continuous full utilization of all available resources

The nova-scheduler



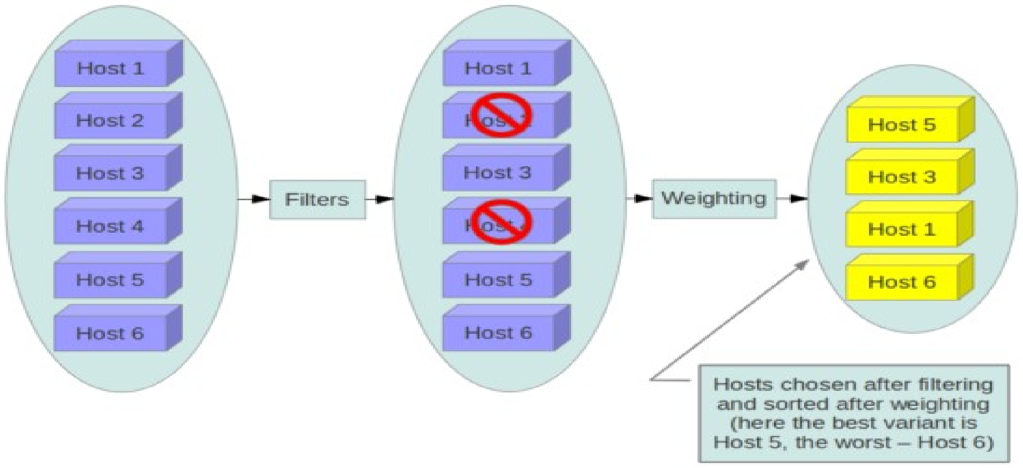
The nova-scheduler

- not a true scheduler: just a resource supplier
- user requests are processed sequentially (FIFO scheduling)
  - nova-scheduler doesn't provide any dynamic priority algorithm
- takes a VM instance request and determines where it should run (which host)
- it makes decisions by collecting information about compute resources
- not satisfied user requests (e.g. resource not available) fail and will be lost
  - on that scenario, nova-scheduler doesn't provide queuing of the requests for retrying mechanism
- pluggable component (default FilterScheduler)
- several configuration options (nova.conf)

The nova-scheduler

The scheduler process is divided into phases:

- **Getting the current state of the all compute nodes:** it will generate a list of hosts
- **filtering phase** will generate a list of suitable hosts by applying filters
- **weighting phase** will sort the hosts according to their weighted cost scores, which are given by applying some cost functions


Host 1
Host 2
Host 3
Host 4
Host 5
Host 6
Filters
Host 1
Host 3
Host 5
Host 6
Weighting
Host 5
Host 3
Host 1
Host 6
Hosts chosen after filtering and sorted after weighting (here the best variant is Host 5, the worst – Host 6)

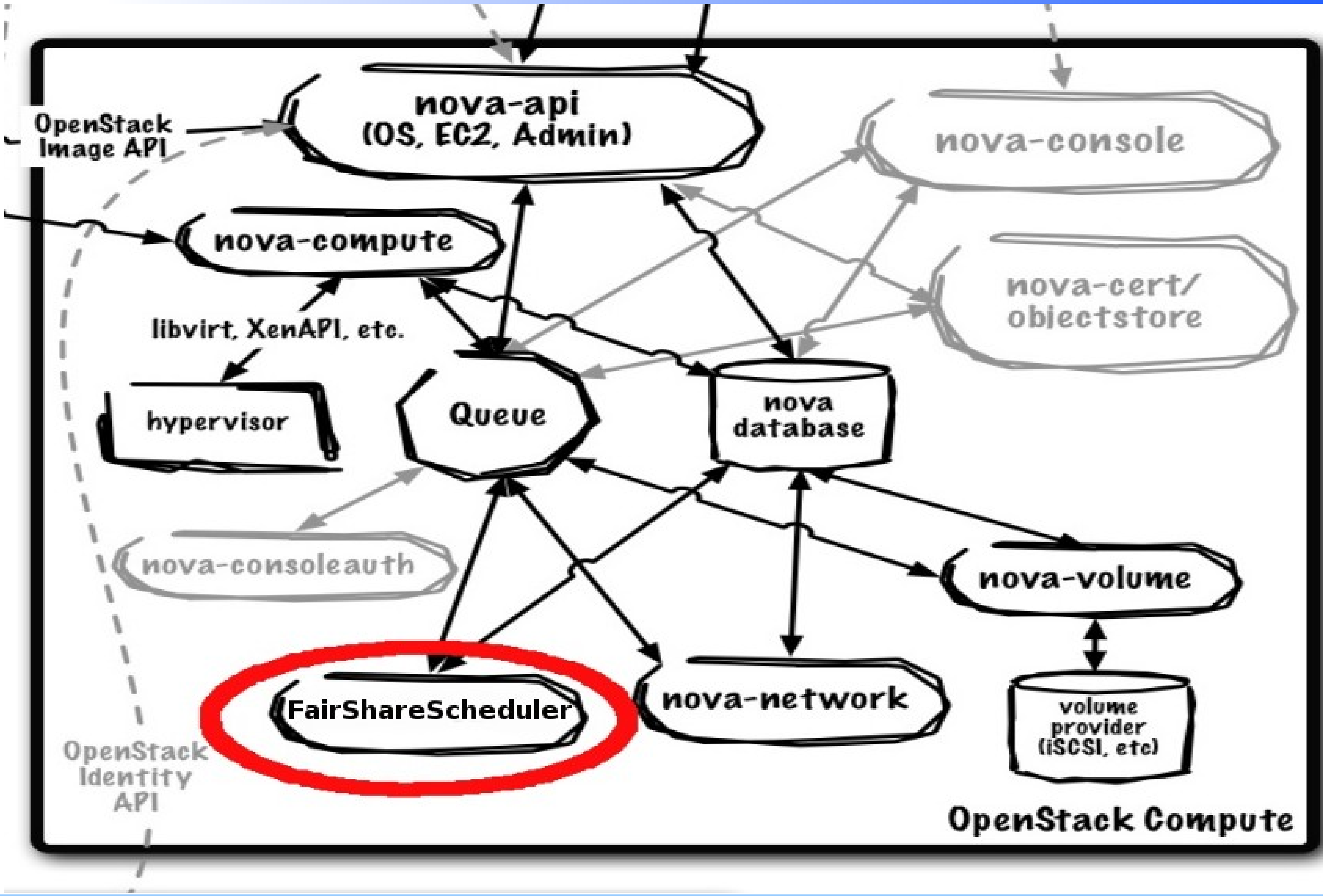
- The sorted list of hosts is candidates to fulfill the user's request

Our proposal: the FairShareScheduler

- Nova-scheduler is mainly missing of:
  - **queuing mechanism of the user requests**
  - **fair-share algorithm in the resources provisioning to guarantee at the same time the continuous full usage of all resources and the quota established for the different users teams**

- INFN has started to address the problem by developing a pluggable scheduler, named *FairShareScheduler*, as extension of the current OpenStack scheduler (i.e FilterScheduler)

Our proposal: the FairShareScheduler



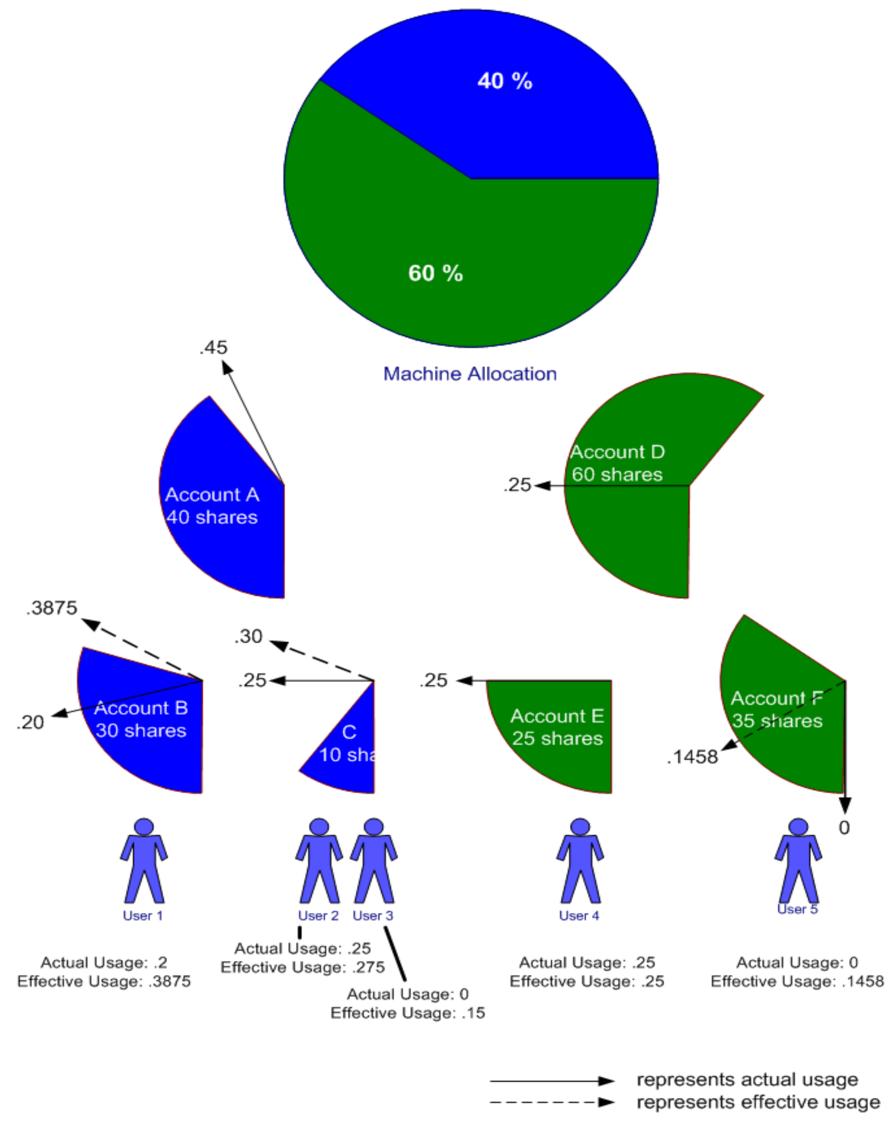
The FairShareScheduler

- FairShareScheduler assigns dynamically the proper priority to every user request
- the priority at any given time is a weighted sum of these factors (configurable): age and fair-share
  - priority = (PriorityWeightAge) * (age_factor) +

    (PriorityWeightVCPUFairshare) * (fair-share-vcpu_factor) +

    (PriorityWeightMemoryFairshare) * (fair-share-memory_factor)
- The weight expresses the interest for a specific factor
  - example: the admin would prefer to make the cpu factor dominant with respect to the others
- We analyzed the fair-share algorithms implemented by the most relevant LRMSes
  - selected the SLURM's Priority MultiFactor strategy, a sophisticated and complete fair-share algorithm
  - https://computing.llnl.gov/linux/slurm/priority_multifactor.html

The SLURM fair-share formula



its formula is:

```
F = 2**(-Ue/S)
```

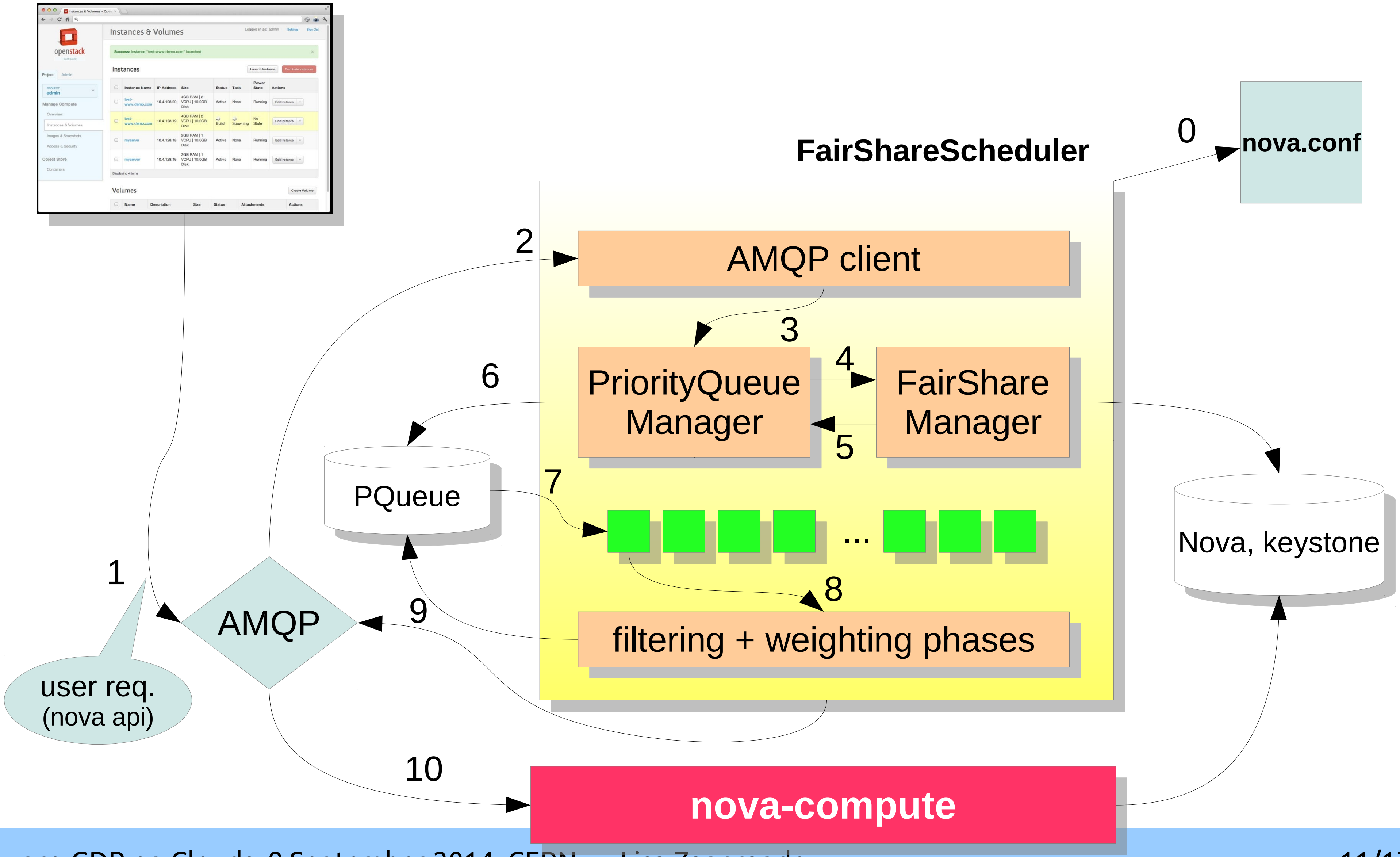
```
Ue: user's effective usage
S: user's normalized share
```

Consider account = tenant

The admin defines in nova.conf the granted share values for each tenant and sets the related quotas as unlimited (big value)

- this workaround gives to the scheduler the full resources management control

FairShareScheduler: the high level architecture



Instances & Volumes - Open

openstack DASHBOARD

Project | Admin

PROJECT admin

Manage Compute
- Overview
- Instances & Volumes
- Images & Snapshots
- Access & Security

Object Store
- Containers

Instances & Volumes — Logged in as: admin — Settings — Sign Out

Success: Instance "test-www.demo.com" launched.

Instances — Launch Instance — Terminate Instances

Instance Name	IP Address	Size	Status	Task	Power State	Actions
test-www.demo.com	10.4.128.20	4GB RAM | 2 VCPU | 10.0GB Disk	Active	None	Running	Edit Instance
test-www.demo.com	10.4.128.19	4GB RAM | 2 VCPU | 10.0GB Disk	Build	Spawning	No State	Edit Instance
myserve	10.4.128.18	2GB RAM | 1 VCPU | 10.0GB Disk	Active	None	Running	Edit Instance
myserver	10.4.128.16	2GB RAM | 1 VCPU | 10.0GB Disk	Active	None	Running	Edit Instance

Displaying 4 items

Volumes — Create Volume

Name	Description	Size	Status	Attachments	Actions

The FairShareScheduler

- all user requests are inserted in a (persistent) priority queue and then processed asynchronously by the dedicated process (filtering + weighting phase) when compute resources are available
- from the client point of view the queued requests remain in “Scheduling” state till the compute resources are available
  - no new states added: this prevents any possible interaction issue with the Openstack clients
- user requests are dequeued by a pool of WorkerThreads (configurable)
  - not a sequential processing as the original scheduler
- the failed requests at filtering + weighting phase may be inserted again in the queue for n-times (configurable)
- the priority of the queued requests will be recalculated periodically (see age_factor)

Current status

- FairShareScheduler prototype ready for HAVANA and IceHouse
  - the source code (for HAVANA and IceHouse) available in our github repository:
    - https://github.com/CloudPadovana/openstack-fairshare-scheduler
  - still open issues: token's expiration
- Testing in progress
  - in Bari's Cloud Testbed
  - University of Victoria will contribute to our work by testing the scheduler on its own cloud infrastructure
  - any contribution from the research community is welcome

Toward the OpenStack collaboration: GANTT

- Nowadays the FairShareScheduler is not integrated in the official OpenStack distribution
- every six months a new OpenStack version is released
  - it is not reasonable to update the scheduler each time (as external team)
  - we wish to contribute internally as OpenStack development team in order to minimize the effort
- It seems that Nova-Scheduler will be deprecated soon and will be replaced by GANTT (Scheduler-as-a-Service)
  - at the moment Gantt is a view of mind rather than an official project
- so, we tried joining GANTT for proposing our solution and for creating possibly a strict collaboration
  - our proposal doesn't match the GANTT business
  - the team suggested us to refer to BLAZAR, a different OpensStack project, which should have a better affinity with our needs

Toward the OpenStack collaboration: BLAZAR

- Now, as second chance, we are focusing on BLAZAR (Reservation-as-a-Service) OpenStack project
  - with BLAZAR user can request the resources of cloud environment to be provided ("leased") to his project for specific amount on time, immediately or in future
  - in terms of benefits added, Resource Reservation Service will:
    - improve visibility of cloud resources consumption (current and planned for future)
    - enable cloud resource planning based on current and future demand from end users
    - automate the processes of resource allocation and reclaiming
    - provide energy efficiency for physical hosts (both compute and storage ones)
    - potentially provide leases as billable items for which customers can be charged a flat fee or a premium price depending on amount/quality of reserved cloud resources and their usage
  - several lease types supported (Immediate reservation, Reservation with retries, Best-effort reservation, Delayed resource acquiring or scheduled reservation)

Toward the OpenStack collaboration: BLAZAR

- We tried to review our needs in terms of "lease"
  - used as a different approach for maximizing the resource utilization
  - typically a team requires virtual machines of three very different lifetime in relations to different types of activities they need to carry out in the Data Centers:
    - Type1 unlimited duration time (UVM) (e.g. web or ftp server hosting)
    - Type2 limited but planned for a well defined date and for a medium-long (typically from 1 to 3 weeks) duration time (LVM)
    - Type3 limited but with short (typically from few hours to 1 day) duration time (SVM)
- We wrote a document to be proposed to the BLAZAR team which describes our use cases and how to deal them in BLAZAR by defining a new lease type (fairShare lease)
  - the document will be sent asap to the BLAZAR community
- Thanks a lot to Tim Bell e Ulrich Schwickerath for their useful input

Any questions?

Thanks a lot!