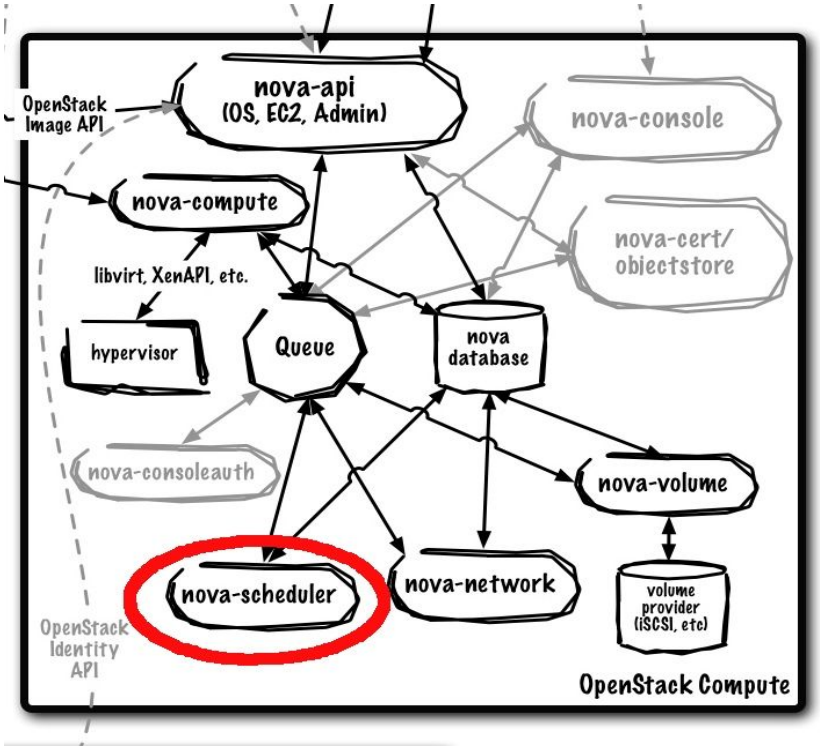
CLOUD AREA PADOVANA



Scheduling in Openstack

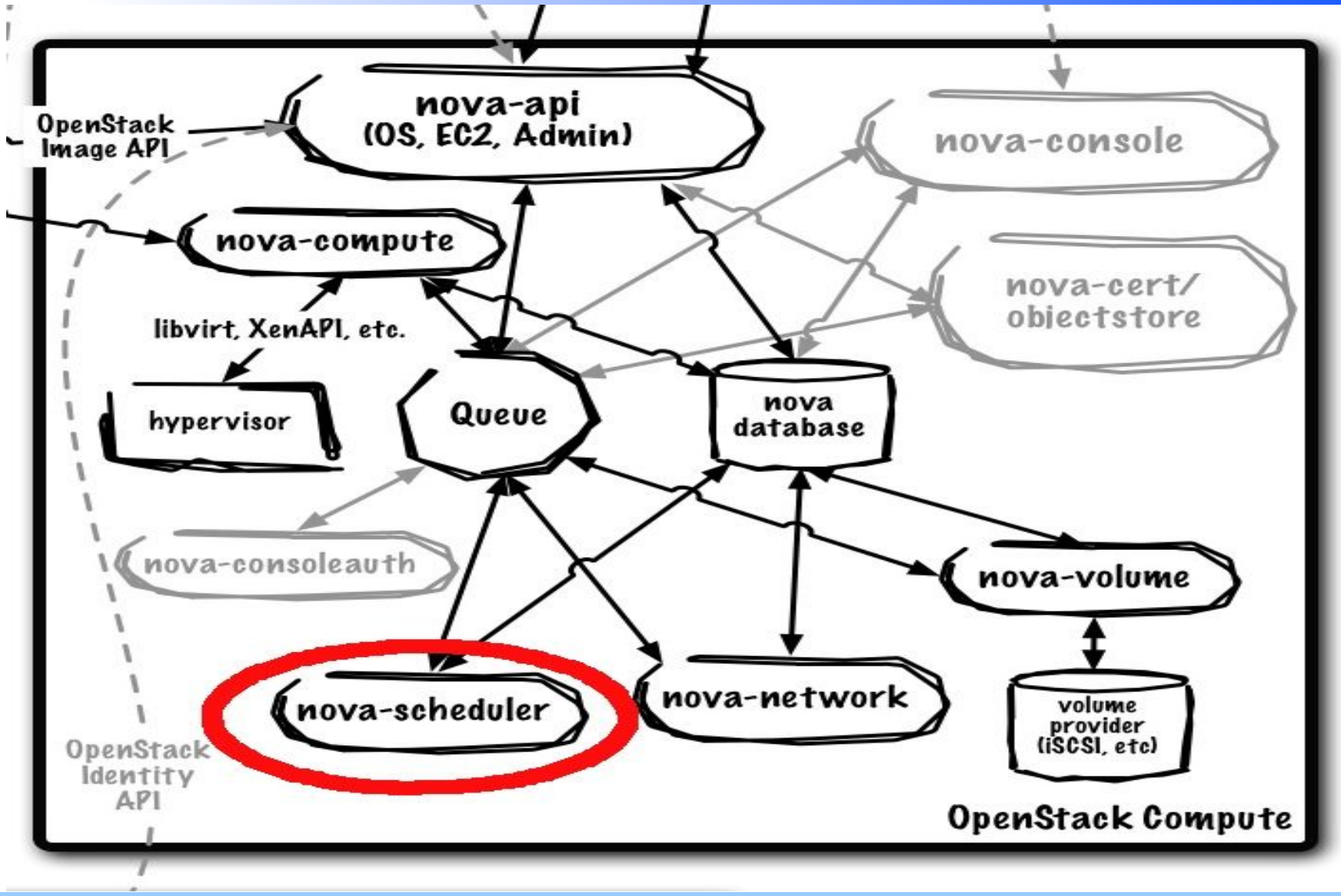
Workshop CCR-INFN
29 May 2014

Eric Frizziero (INFN - Padova)

The nova-scheduler

- Nova-scheduler is the component responsible to decide which compute node should be used when a VM instance is to be lunched
- it interacts with other components through queue
  - for scheduling, queue is the essential communications hub
- nova-scheduler makes decisions by collecting information about compute resources
- it has a number of configuration options that can be accessed and modified in the configuration file “nova.conf”
- FilterScheduler is the default

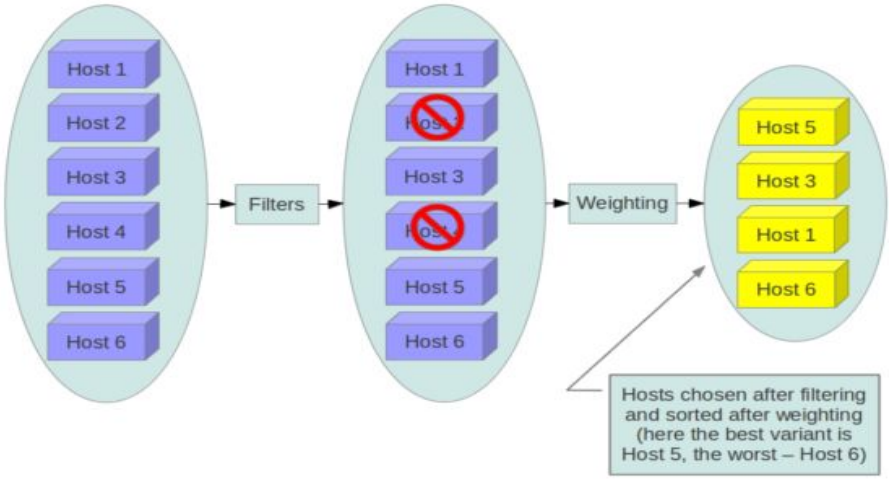
The nova-scheduler



The nova-scheduler

The scheduler process is divided into the following phases:

- **Getting the current state of all the 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



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

The user request

2013-12-17 17:10:30.895 3032 DEBUG qpid.messaging.io.raw [-] READ[3e8ea28]:
'\x0b\x01\x00\x16\x00\x01\x00\x00\x00\x00\x00\x00\x04\x01\x01\x00\x07\x00\x010\x01\x00\x03\x02\x00U\x00\x01\x00\x00\x00\x00\x00\x00\x00\x00\x00.\x04\x03\x10\x01\x08amqp/map\x00\x00\x00\x1d\x00\x00\x00\x01\x0cqpid.subject\x95\x00\tscheduler\x00\x00\x00\x13\x04\x01\x00\x03\x04nova\tscheduler\x07\x03\x12\xeb\x00\x01\x00\x00\x00\x00\x00\x00\x00\x00\x12\xdb\x00\x00\x00\x02\x0coslo.message\x95\x12\xb4{"_context_roles": ["_member_", "admin"], "_context_request_id": "req-066522b7-e5f2-4af2-845f-d13380270324", "_context_quota_class": null, "_context_service_catalog": [], "_context_tenant": "e70bb1af044648e9b1ccc62836c101b0", "args": {"legacy_bdm_in_spec": false, "request_spec": {"instance_type": {"root_gb": 20, "name": "m1.small", "ephemeral_gb": 0, "memory_mb": 2048, "vcpus": 1, "extra_specs": {}, "swap": 0, "rxtx_factor": 1.0, "flavorid": "2", "vcpu_weight": null, "id": 5}, "num_instances": 1, "block_device_mapping": [{"instance_uuid": "41b812a9-2bb7-4d7c-b1b2-ef1f6a5e9ee2", "guest_format": null, "boot_index": 0, "no_device": null, "connection_info": null, "image_id": "38a64756-827a-4c03-a22d-f601ea060019", "volume_id": null, "device_name": null, "disk_bus": null, "volume_size": null, "source_type": "image", "device_type": "disk", "snapshot_id": null, "destination_type": "local", "delete_on_termination": true}], "instance_properties": {"vm_state": "building", "availability_zone": "nova", "terminated_at": null, "ephemeral_gb": 0, "instance_type_id": 5, "user_data": null, "cleaned": false, "vm_mode": null, "deleted_at": null, "reservation_id": "r-c3dojj8b", "id": 5, "security_groups": {"objects": []}, "disable_terminate": false, "root_device_name": null, "display_name": "centos", "uuid": "41b812a9-2bb7-4d7c-b1b2-ef1f6a5e9ee2", "default_swap_device": null, "info_cache": {"instance_uuid": "41b812a9-2bb7-4d7c-b1b2-ef1f6a5e9ee2", "network_info": []}, "hostname": "centos", "launched_on": null, "display_description": "centos", "key_data": null, "deleted": false, "config_drive": "", "power_state": 0, "default_ephemeral_device": null, "progress": 0, "project_id": "e70bb1af044648e9b1ccc62836c101b0", "launched_at": null, "scheduled_at": null, "node": null, "ramdisk_id": "6580f36e-2f88-4f01-ad4e-a7696098dd3c", "access_ip_v6": null, "access_ip_v4": null, "kernel_id": "745d2e2b-9996-4a5a-a33e-4df75f6c691b", "key_name": null, "updated_at": null, "host": null, "user_id": "ccdc7c1d0d114b8a972e3e7f1a032c99", "system_metadata": {"image_kernel_id": "745d2e2b-9996-4a5a-a33e-4df75f6c691b", "image_min_disk": 20, "instance_type_memory_mb": 2048, "instance_type_swap": 0, "instance_type_vcpu_weight": null, "instance_type_root_gb": 20, "instance_type_name": "m1.small", "image_ramdisk_id": "6580f36e-2f88-4f01-ad4e-a7696098dd3c", "instance_type_id": 5, "instance_type_ephemeral_gb": 0, "instance_type_rxtx_factor": 1.0, "instance_type_flavorid": "2", "image_container_format": "ami", "instance_type_vcpus": 1, "image_min_ram": 0, "image_disk_format": "ami", "image_base_image_ref": "38a64756-827a-4c03-a22d-f601ea060019"}, "task_state": "scheduling", "shutdown_terminate": false, "cell_name": null, "root_gb": 20, "locked": false, "name": "instance-00000005", "created_at": "2013-12-17T16:10:30.788072", "locked_by": null, "launch_index": 0, "memory_mb": 2048, "vcpus": 1, "image_ref": "38a64756-827a-4c03-a22d-f601ea060019", "architecture": null, "auto_disk_config": false, "os_type": null, "metadata": {}}, "security_group": ["default"], "image": {"status": "active", "name": "centos", "deleted": false, "container_format": "ami", "created_at": "2013-12-17T16:09:34.000000", "disk_format": "ami", "updated_at": "2013-12-17T16:09:50.000000", "properties": {"kernel_id": "745d2e2b-9996-4a5a-a33e-4df75f6c691b", "ramdisk_id": "6580f36e-2f88-4f01-ad4e-a7696098dd3c"}, "min_disk": 0, "min_ram": 0, "checksum": "5eba5009290eb1534f8f22a3245e7c48", "owner": "e70bb1af044648e9b1ccc62836c101b0", "is_public": false, "deleted_at": null, "id": "38a64756-827a-4c03-a22d-f601ea060019", "size": 1073741824}, "instance_uuids": ["41b812a9-2bb7-4d7c-b1b2-ef1f6a5e9ee2"]}, "is_first_time": true, "filter_properties": {"instance_type": {"disabled": false, "root_gb": 20, "name": "m1.small", "flavorid": "2", "deleted": 0, "created_at": null, "ephemeral_gb": 0, "updated_at": null, "memory_mb": 2048, "vcpus": 1, "extra_specs": {}, "swap": 0, "rxtx_factor": 1.0, "is_public": true, "deleted_at": null, "vcpu_weight": null, "id": 5}, "scheduler_hints": {}}, "admin_password": "8xsGpfUPKBh9", "injected_files": [], "requested_networks": [["f6d1fb29-f212-4640-afb8-748879462eb8", null, null]]}, "_unique_id": "da1731c3e72947fab6ec39670b9bfe01", "_context_timestamp": "2013-12-17T16:10:30.484714", "_context_user_id": "ccdc7c1d0d114b8a972e3e7f1a032c99", "_context_project_name": "admin", "_context_read_deleted": "no", "_context_auth_token": "0ecf459e8dcf5bd35e4ec1e4d167db10", "namespace": null, "_context_instance_lock_checked": false, "_context_is_admin": true, "version": "2.9", "_context_project_id": "e70bb1af044648e9b1ccc62836c101b0", "_context_user": "ccdc7c1d0d114b8a972e3e7f1a032c99", "_context_user_name": "admin", "method": "run_instance", "_context_remote_address": "193.206.210.48"}\x0coslo.version\x95\x00\x032.0' readable /usr/lib/python2.6/site-packages/qpid/messaging/driver.py:416

The nova-scheduler

- User requests are processed sequentially (FIFO scheduling)
  - nova-scheduler doesn't provide any dynamic priority strategy algorithm
- User requests not satisfied (e.g. resource not available) fails and will be lost
  - on that scenario, nova-scheduler doesn't provide queuing of the requests
- OpenStack simply provides a partitioning of resources among more projects / experiments (use of quotas)
  - if a project has free quota (underutilized its resources), and another project instead has consumed its quota, the only solution is to change the related quotas by the Cloud Administrator

Nova-scheduler is mainly missing of:

- ✔ **queuing of the requests**
- ✔ **fair-share algorithm in the resources provisioning**

The nova-scheduler

All Instances

Logged in as: admin | Settings | Help | Sign Out

Instances

Filter | Filter | Terminate Instances

Project	Host	Name	Image Name	IP Address	Size	Status	Task	Power State	Uptime	Actions
p1	-	p1_u1_VM5	centos_6.4		m1.tiny | 512MB RAM | 1 VCPU | 1.0GB Disk	Error	None	No State	0 minutes	Edit Instance / More
p1	gilda-11.pd.infn.it	p1_u1_VM4	centos_6.4	192.168.252.52	m1.tiny | 512MB RAM | 1 VCPU | 1.0GB Disk	Active	None	Running	2 minutes	Edit Instance / More
p2	gilda-11.pd.infn.it	p2_u1_VM1	centos_6.4	192.168.252.51	m1.tiny | 512MB RAM | 1 VCPU | 1.0GB Disk	Active	None	Running	5 hours, 43 minutes	Edit Instance / More
p2	gilda-11.pd.infn.it	p2_u1_VM2	centos_6.4	192.168.252.50	m1.tiny | 512MB RAM | 1 VCPU | 1.0GB Disk	Active	None	Running	5 hours, 44 minutes	Edit Instance / More
p2	gilda-11.pd.infn.it	p2_u1_VM1	centos_6.4	192.168.252.49	m1.tiny | 512MB RAM | 1 VCPU | 1.0GB Disk	Active	None	Running	5 hours, 44 minutes	Edit Instance / More

Resources unavailable: any new request fails and is lost.

Fair-share algorithm

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

The fair-share is a component of the job's priority that influences the order in which a user's queued jobs are scheduled to run

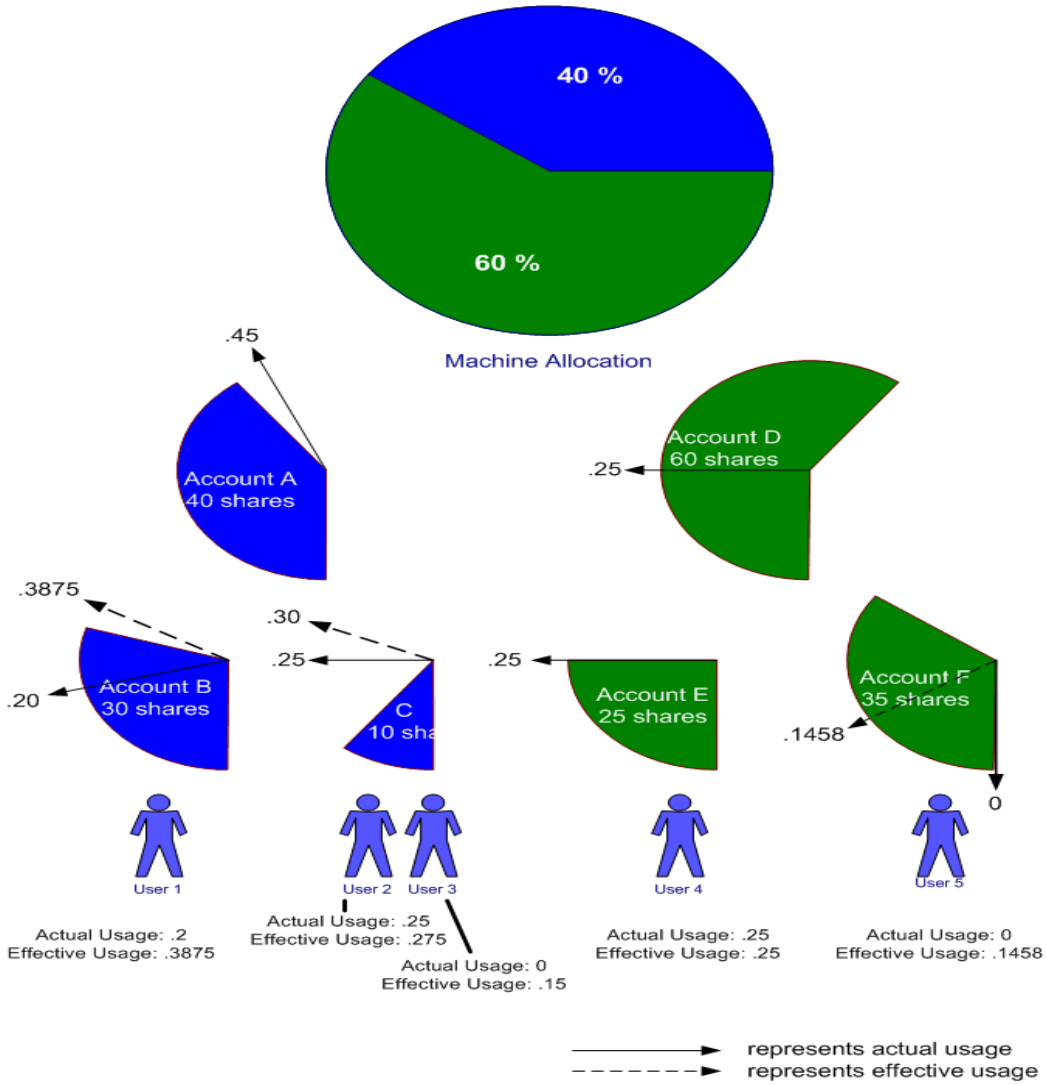
It guarantees the usage of the resources is equally distributed among users and groups by considering the portion of the resources allocated to them (i.e. share) and the resources they already consumed

- historical resource utilization information is incorporated into priority decisions

We analyzed the fair-share algorithms implemented by the most relevant LRMS

- Selected **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



SLURM fair-share formula

$$F = 2^{(-Ue/S)}$$

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

Consider account = tenant

The share values must be defined in the nova.conf

FairShareScheduler

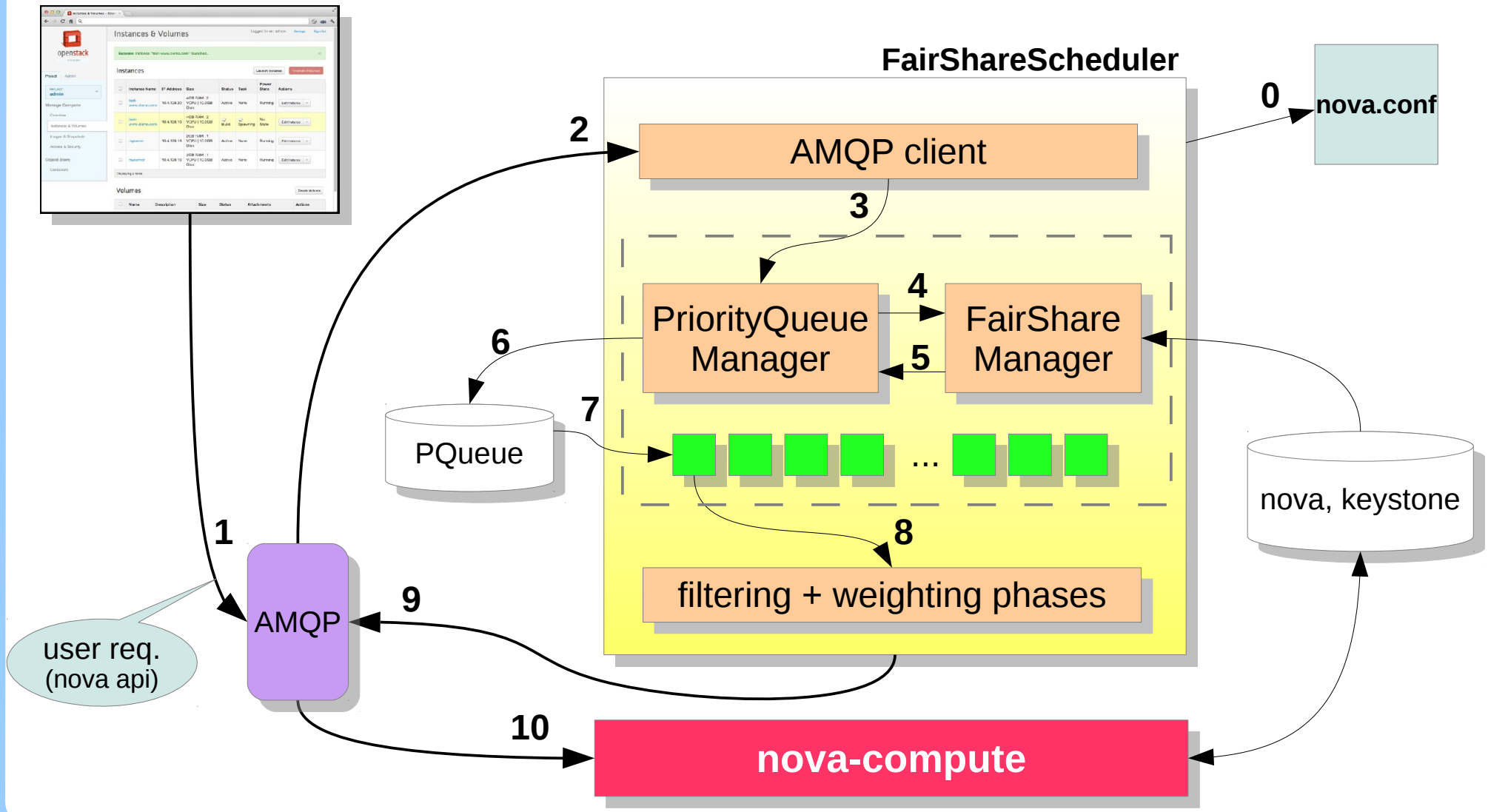
FairShareScheduler, a pluggable scheduler with the objective to extend the existing OpenStack scheduler by integrating a (batch like) dynamic priority algorithm has been developed by INFN-PD

- Selected the "**Multifactor Priority**" **SLURM** algorithm
- FairShareScheduler will assign dynamically the proper priority to every user request
- the priority at any given time will be 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: you can configure fair-share-vcpu to be the dominant factor (say 80%), set the age factor to contribute 20%, and set the fair-share-memory influences to zero in the priority decision

FairShareScheduler

- all user requests will be 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)
  - no sequential processing of the requests
- 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)

FairShareScheduler: High level architecture



FairShareScheduler

All Instances

Logged in as: admin | Settings | Help | Sign Out

Instances

Filter | Filter | Terminate Instances

Project	Host	Name	Image Name	IP Address	Size	Status	Task	Power State	Uptime	Actions
p2	-	p2_u1_VM1	centos_6.4		m1.tiny | 512MB RAM | 1 VCPU | 1.0GB Disk	Build	Scheduling	No State	0 minutes	Edit Instance / More
p1	gilda-11.pd.infn.it	p1_u2_VM2	centos_6.4	192.168.252.52	m1.tiny | 512MB RAM | 1 VCPU | 1.0GB Disk	Active	None	Running	0 minutes	Edit Instance / More
p1	gilda-11.pd.infn.it	p1_u2_VM1	centos_6.4	192.168.252.51	m1.tiny | 512MB RAM | 1 VCPU | 1.0GB Disk	Active	None	Running	0 minutes	Edit Instance / More
p2	gilda-11.pd.infn.it	p2_u1_VM2	centos_6.4	192.168.252.50	m1.tiny | 512MB RAM | 1 VCPU | 1.0GB Disk	Active	None	Running	1 minute	Edit Instance / More
p2	gilda-11.pd.infn.it	p2_u1_VM1	centos_6.4	192.168.252.49	m1.tiny | 512MB RAM | 1 VCPU | 1.0GB Disk	Active	None	Running	1 minute	Edit Instance / More

Resources unavailable: new requests remain Scheduling (no host, no ip address)

FairShareScheduler

All Instances

Logged in as: admin | Settings | Help | Sign Out

Instances

Filter | Filter | Terminate Instances

Project	Host	Name	Image Name	IP Address	Size	Status	Task	Power State	Uptime	Actions
p2	-	p2_u1_VM1	centos_6.4		m1.tiny | 512MB RAM | 1 VCPU | 1.0GB Disk	Build	Scheduling	No State	0 minutes	Edit Instance / More
p1	gilda-11.pd.infn.it	p1_u2_VM2	centos_6.4	192.168.252.52	m1.tiny | 512MB RAM | 1 VCPU | 1.0GB Disk	Active	None	Running	0 minutes	Edit Instance / More
p1	gilda-	_2_VM1	centos_6.4	192.168.252.51	m1.tiny | 512MB RAM | 1 VCPU | 1.0GB Disk	Active	None	Running	0 minutes	Edit Instance / More
p2	gilda-11.pd.infn.it	p2_u1_VM2	centos_6.4	192.168.252.50	m1.tiny | 512MB RAM | 1 VCPU | 1.0GB Disk	Active	None	Running	1 minute	Edit Instance / More
p2	gilda-11.pd.infn.it	p2_u1_VM1	centos_6.4	192.168.252.49	m1.tiny | 512MB RAM | 1 VCPU | 1.0GB Disk	Active	None	Running	1 minute	Edit Instance / More

Terminate Instances

Resources unavailable. Terminate one instance.

FairShareScheduler

All Instances

Logged in as: admin | Settings | Help | Sign Out

Instances

Filter | Filter | Terminate Instances

	Project	Host	Name	Image Name	IP Address	Size	Status	Task	Power State	Uptime	Actions
	p2	gilda-11.pd.infn.it	p2_u1_VM1	centos_6.4	192.168.252.51	m1.tiny | 512MB RAM | 1 VCPU | 1.0GB Disk	Active	None	Running	2 minutes	Edit Instance / More
	p1	gilda-11.pd.infn.it	p1_u2_VM2	centos_6.4	192.168.252.52	m1.tiny | 512MB RAM | 1 VCPU | 1.0GB Disk	Active	None	Running	3 minutes	Edit Instance / More
	p2	gilda-11.pd.infn.it	p2_u1_VM2	centos_6.4	192.168.252.50	m1.tiny | 512MB RAM | 1 VCPU | 1.0GB Disk	Active	None	Running	3 minutes	Edit Instance / More
	p2	gilda-11.pd.infn.it	p2_u1_VM1	centos_6.4	192.168.252.49	m1.tiny | 512MB RAM | 1 VCPU | 1.0GB Disk	Active	None	Running	3 minutes	Edit Instance / More
	p1	gilda-11.pd.infn.it	p1_u2_VM8	centos_6.4	192.168.252.48	m1.tiny | 512MB RAM | 1 VCPU | 1.0GB Disk	Active	None	Running	9 minutes	Edit Instance / More

The previous scheduled request goes Running (assigned both host and ip address).

FairShareScheduler: Use Case 1

QUACK (Queues in Openstack)

CCR Cloud Working Group task (Bari, CNAF, Padova)

Objective: **to implement the integration of Cloud in an existing Grid environment (LRMS)**

- use the same resources for Cloud computing and Grid computing (**no partitioning**)
- All requests for resources (batch jobs, Cloud) will result in requests for allocations of VMs to OpenStack
- FairShareScheduler provides both queuing of the requests and fair-share algorithm in the resources provisioning

FairShareScheduler: Use Case 2

Assessing a possible evolution of the CREAM architecture to submit jobs directly to IaaS (with FairShareScheduler)

- CREAM responsabile of the VM's life-cycle management
  - jobs executed in the VMs
- The batch system is not more necessary
  - fair-share and queueing provided by the FairShareScheduler
- Nothing changes for the Grid users
  - No updates of the WS interfaces are foreseen

Virgo experiment has shown interest in this type of architecture

- for further details, please see the yesterday Virgo presentation

Conclusion and next steps

- FairShareScheduler source code (for HAVANA) available in github:
  - https://github.com/CloudPadovana/openstack-fairshare-scheduler
  - Authors: Eric Frizziero (INFN-PD), Lisa Zangrando (INFN-PD)
- Testing in progress in Bari's Cloud Testbed
- Installation of the FairShareScheduler in the “Cloud Area Padovana”
- Started the integration process of the FairShareScheduler in IceHouse Openstack release
- Evaluate how to integrate this scheduler in the official OpenStack distribution

- FairshareScheduler's live demo at coffee break area

Thank you for your attention!

Questions?