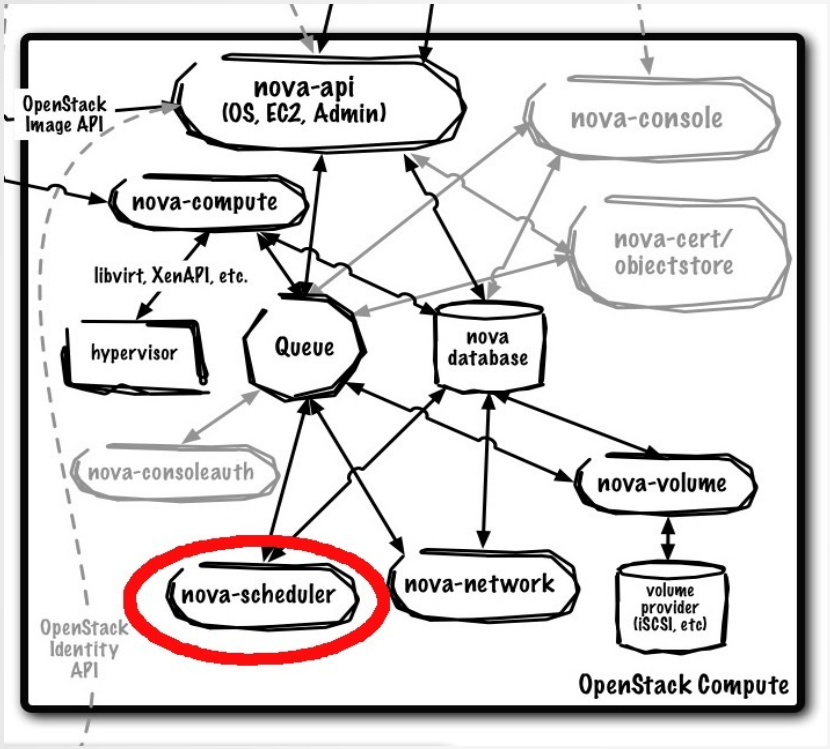
Openstack scheduler enhancement

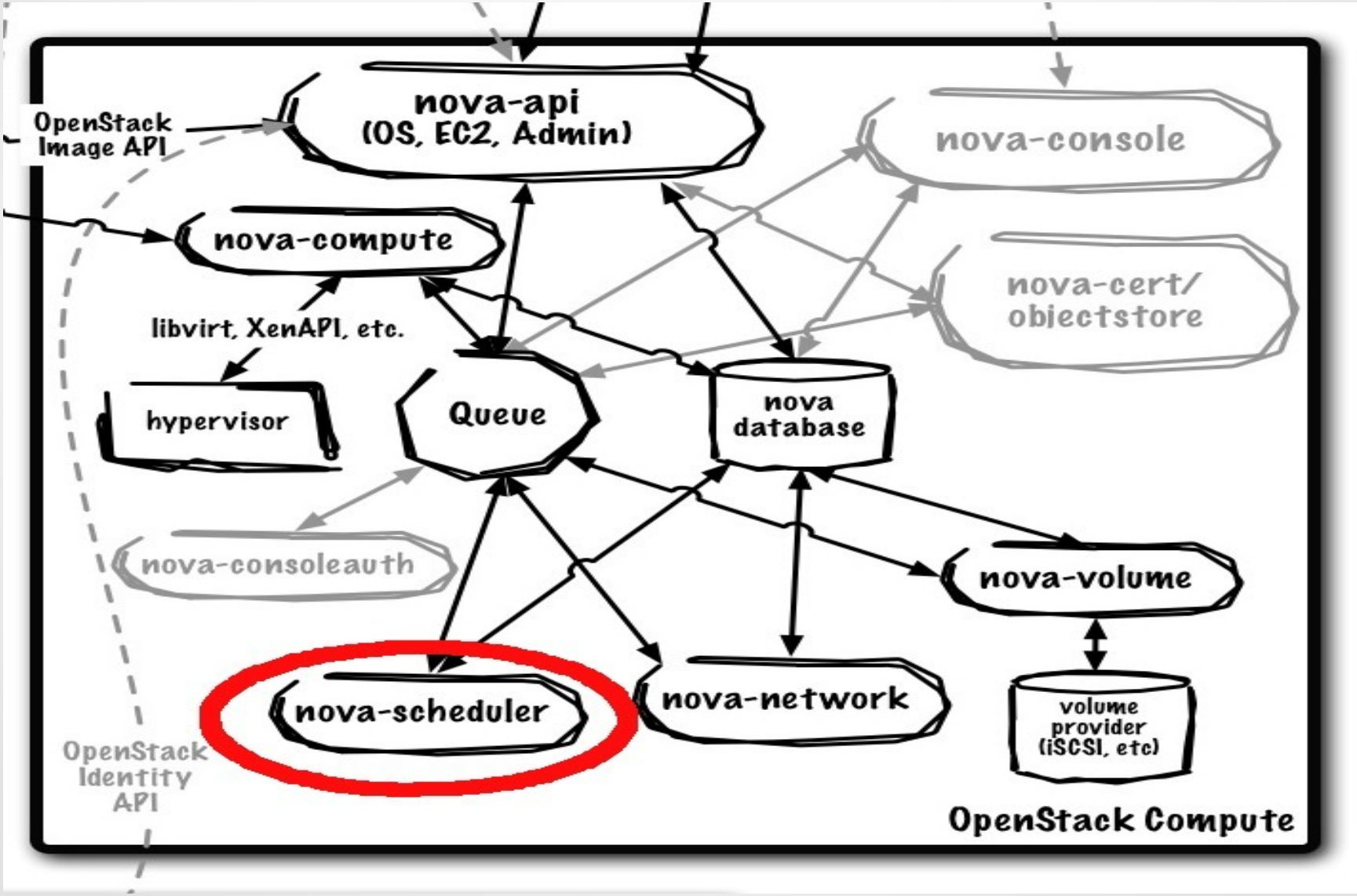


Lisa Zangrando, Eric Frizziero
INFN-Padova

The nova-scheduler

- nova-scheduler is responsible to decide which compute node host should launch an image instance (server)
- it interacts with other components through queue and central database repo
  - for scheduling, queue is the essential communications hub
- all compute nodes (hosts) periodically publish their status, resources available and hardware capabilities to nova-scheduler through the queue
- nova-scheduler then collects this data and uses it to make decisions when a request comes in

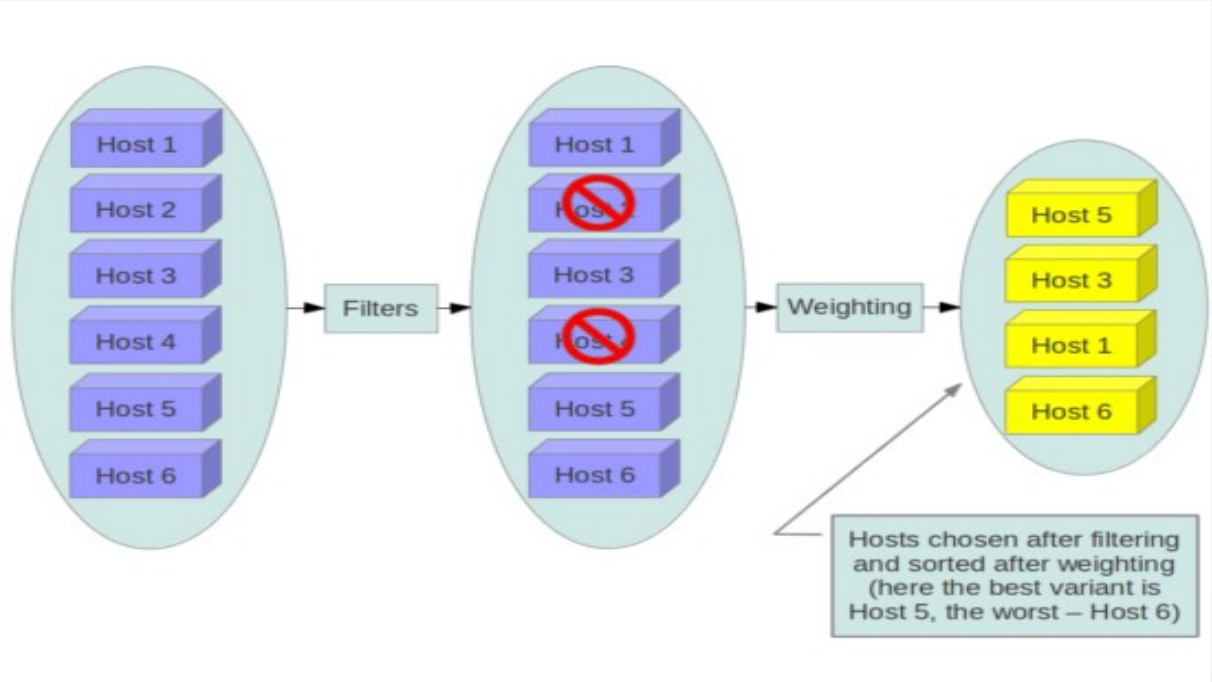
The nova-scheduler



The nova-scheduler

The whole process is divided into two phases:

- **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 candidates to fulfill the user's request. How many hosts in this list will be used depends on the number of instances requested by the user.

The nova-scheduler

- user requests are processed sequentially (FIFO scheduling)
  - nova-scheduler doesn't provide any dynamic priority strategy algorithm
- requests dispatched by the AMQP based message broker
  - RabbitMQ / Apache Qpid™
  - AMQP (Advanced Message Queueing Protocol) is an open internet protocol for reliably sending and receiving messages. It makes it possible for everyone to build a diverse, coherent messaging ecosystem
- Request not satisfied (e.g. resource not available) fails and will be lost

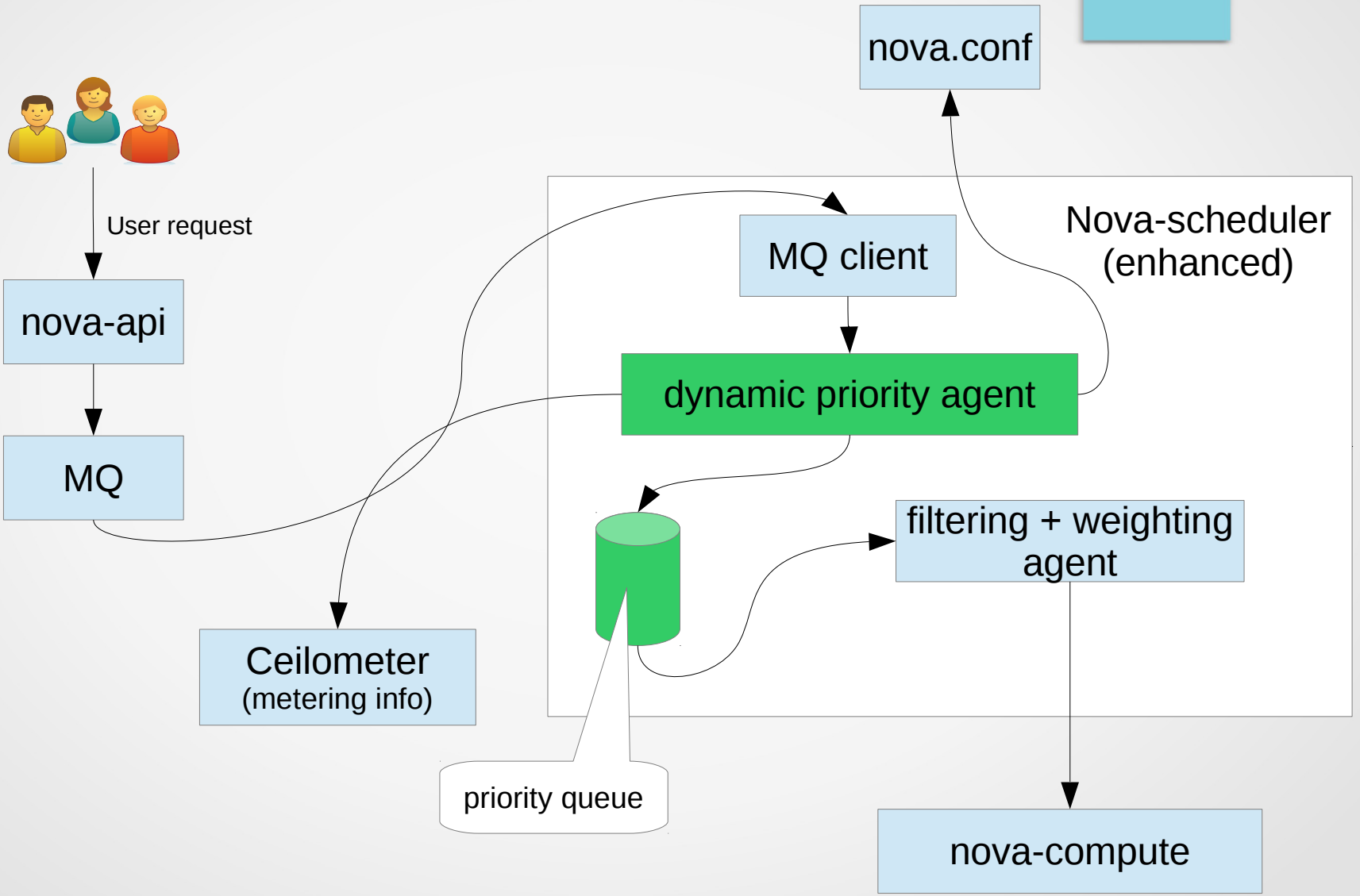
The user request (qpid)

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\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 scheduling algorithm enhancement

- Objective: extend the existing OpenStack scheduler by integrating a (batch like) dynamic priority algorithm
- the new scheduler will be enabled in nova.conf by the admin (plug-in)
- the scheduler will assign dynamically the proper priority to every user request
- the priority at any given time will be a weighted sum of two factors: age and fair-share
  - priority = (PriorityWeightAge) * (age_factor) +
    (PriorityWeightFairshare) * (fair-share_factor)
- all user requests will be inserted in a (persistent) priority queue and processed asynchronously by the dedicated process (filtering + weighting phase)
- 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

the new high level architecture



The priority

- In general any LRMS defines the priority at any given time as a weighted sum of several factors
  - e.g. SLURM: age, fair-share, job-size, partition, QOS etc
  - job_priority = (PriorityWeightAge) * (age_factor) +

    (PriorityWeightFairshare) * (fair-share_factor) +

    (PriorityWeightJobSize) * (job_size_factor) +

    (PriorityWeightPartition) * (partition_factor) +

    (PriorityWeightQOS) * (QOS_factor)
  - Age: the length of time a job has been waiting in the queue, eligible to be scheduled
  - Fair-share: the difference between the portion of the computing resource that has been promised and the amount of resources that has been consumed
  - Job size: the number of nodes a job is allocated
  - Partition: a factor associated with each node partition
  - QOS: a factor associated with each Quality Of Service

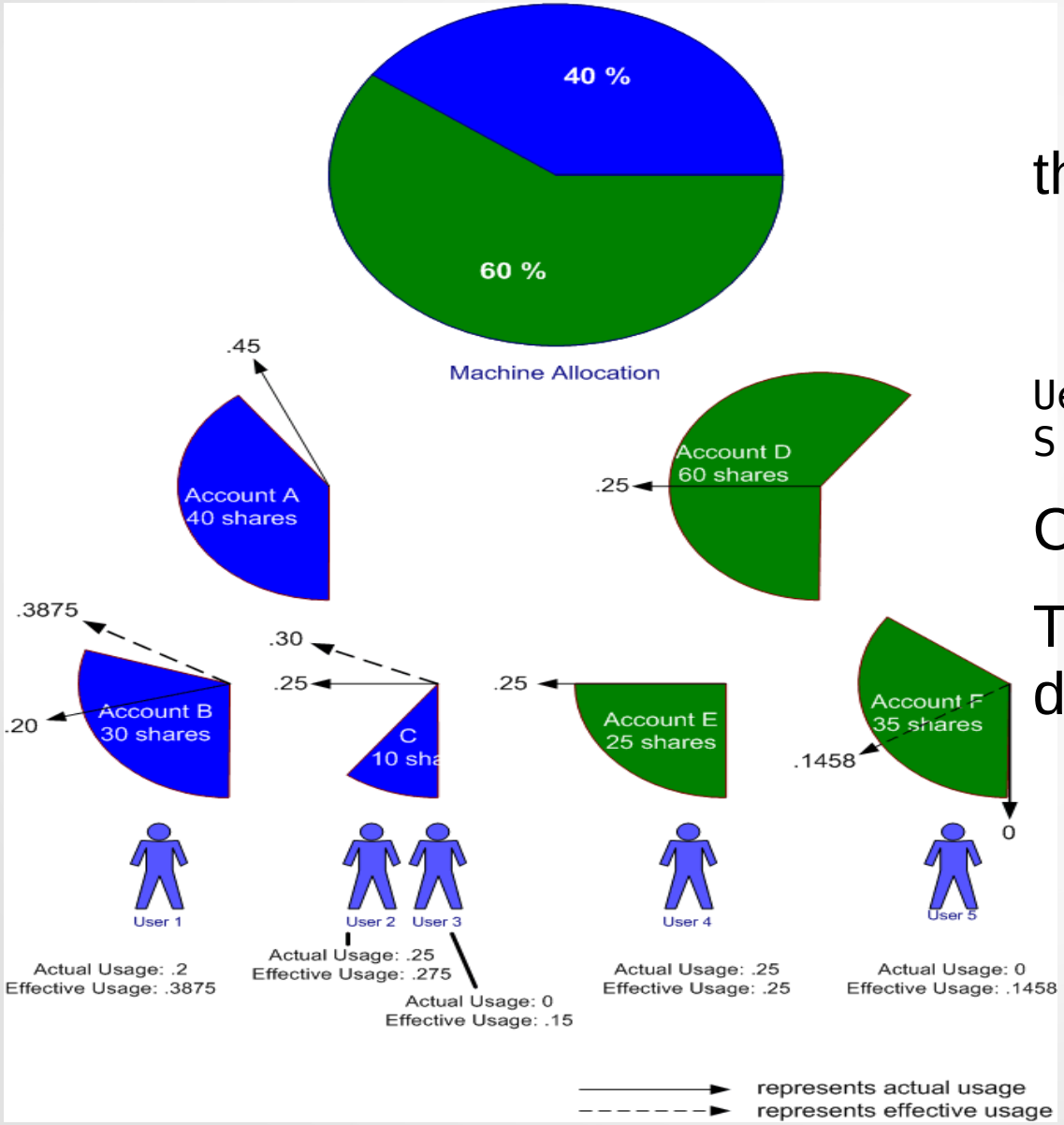
The priority

- 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 weight expresses the interest of the site admin on a specific factor
  - for example, a site could configure fair-share to be the dominant factor (say 70%), set the job size and the age factors to each contribute 15%, and set the partition and QOS influences to zero

The fair-share algorithm

- Fair-share: mechanism which allows historical resource utilization information to be incorporated into priority decisions
- The fair-share component to a job's priority influences the order in which a user's queued jobs are scheduled to run based on the portion of the computing resources they have been allocated and the resources their jobs have already consumed
- Analyzed the most important fair-share algorithms (LSF, MAUI, SLURM)
  - selected the SLURM algorithm
    - https://computing.llnl.gov/linux/slurm/priority_multifactor.html
  - CEA (Commissariat à l'énergie atomique)
    - provides an optimized version of the Slurm fair-share algorithm
      - http://slurm.schedmd.com/SUG13/cea-site-report-0.6.pdf
    - Tera-100 supercomputer

The SLURM fair-share formula



the 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

Toward the first prototype

- Source code of exiting Havana scheduler analyzed
  - we are able to build it
- AMQP message broker analyzed
  - RabbitMQ / Apache Qpid™
  - we are able to access to the message queue, consume messages and access to the contained info
- Creation of a priority queue in python
  - persistence not yet tested
- fair-share algorithm understood
  - implementation is ongoing

Toward the first prototype

- A proof-of-concept was executed accessing directly to the Openstack databases in order to retrieve the information needed for the “fair-share” among users or projects.

  keystone db - Tables: user, project.

  nova db - Table: instances.
- It's not recommended to access directly to the Openstack databases
- Could Ceilometer (Telemetry) be used for retrieving the necessary information to executed the “fair-share”?

Testbed environment

- Host “cream-mstr-018.pd.infn.it” with one network card (eth0).
  - Switch port configured in trunk mode
  - virtualization network card (VLAN 19): eth0.19
- Openstack “Havana” installed by using the packstack tool
  - Packstack seems to be not able to configure the eth0.X properly
    - OVS (Open vSwitch) plug-in configuration
  - Ceilometer installation performed by packstack fails
    - workaround: installed by hand.
  - Neutron configuration doesn't work fine at the moment
    - Internal network reachable by all WMs (e.g. ping, ssh, etc)
    - External network not yet reachable
  - A lot of documentation but not often useful

Conclusion

- OpenStack “Havana” scheduler architecture and source code analyzed
- AMQP message brokering analyzed
- Fair-share algorithm analyzed
- New scheduler architecture defined (non-invasive)
- Testbed @ PD almost ready (external network problem)