CHINA
OpenStack Days

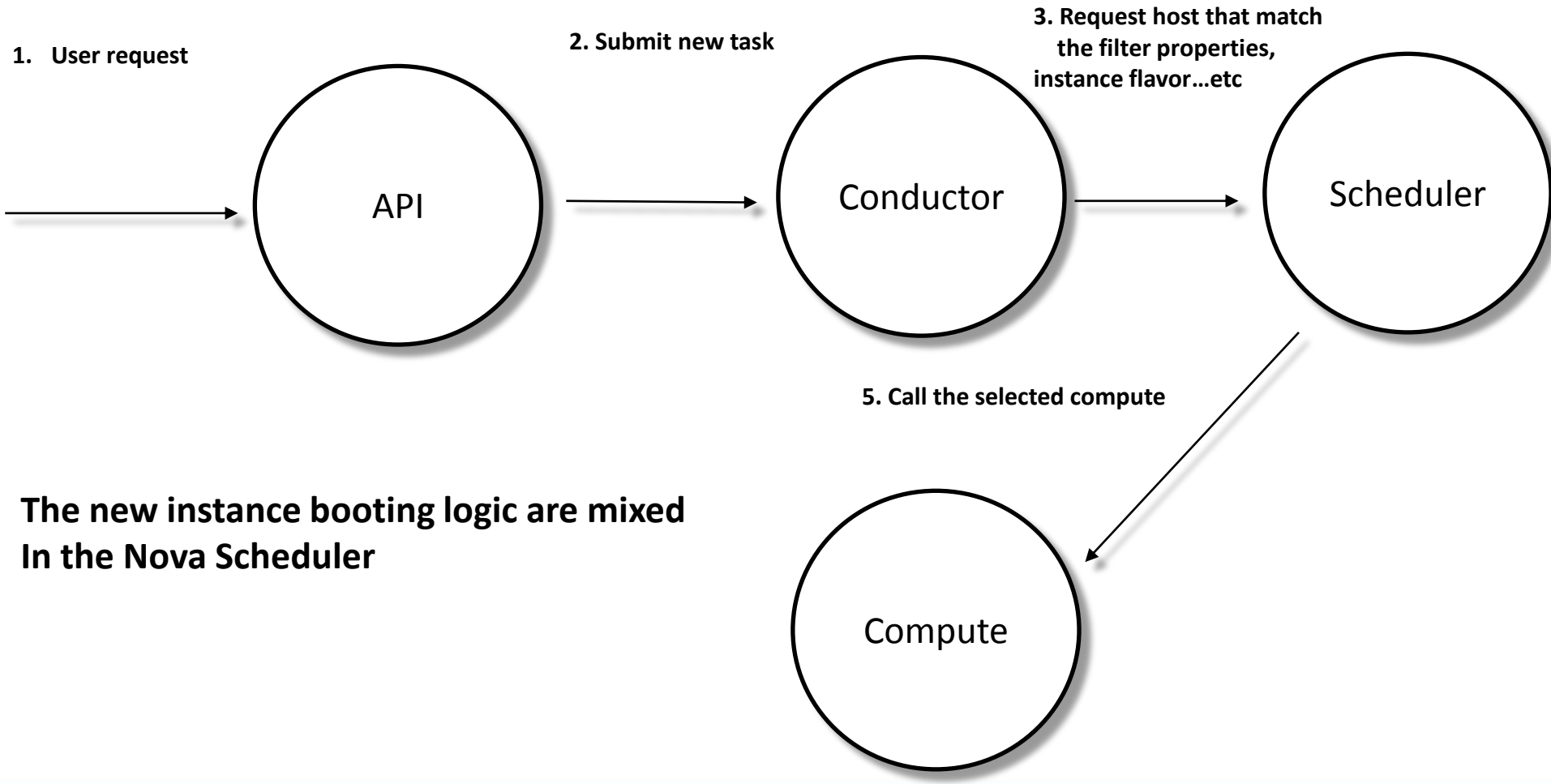
OpenStack Nova Scheduler 更新 & Placement

徐贺杰 **Alex Xu**
OpenStack Nova Core reviewer

Agenda

- The change of Nova Scheduler…
- The new Placement service

Before Juno…



The new instance booting logic are mixed In the Nova Scheduler

People wants a separate Scheduler

- A common scheduler, can be used by other OpenStack components
- Consider multiple resources, like, compute, storage, network…

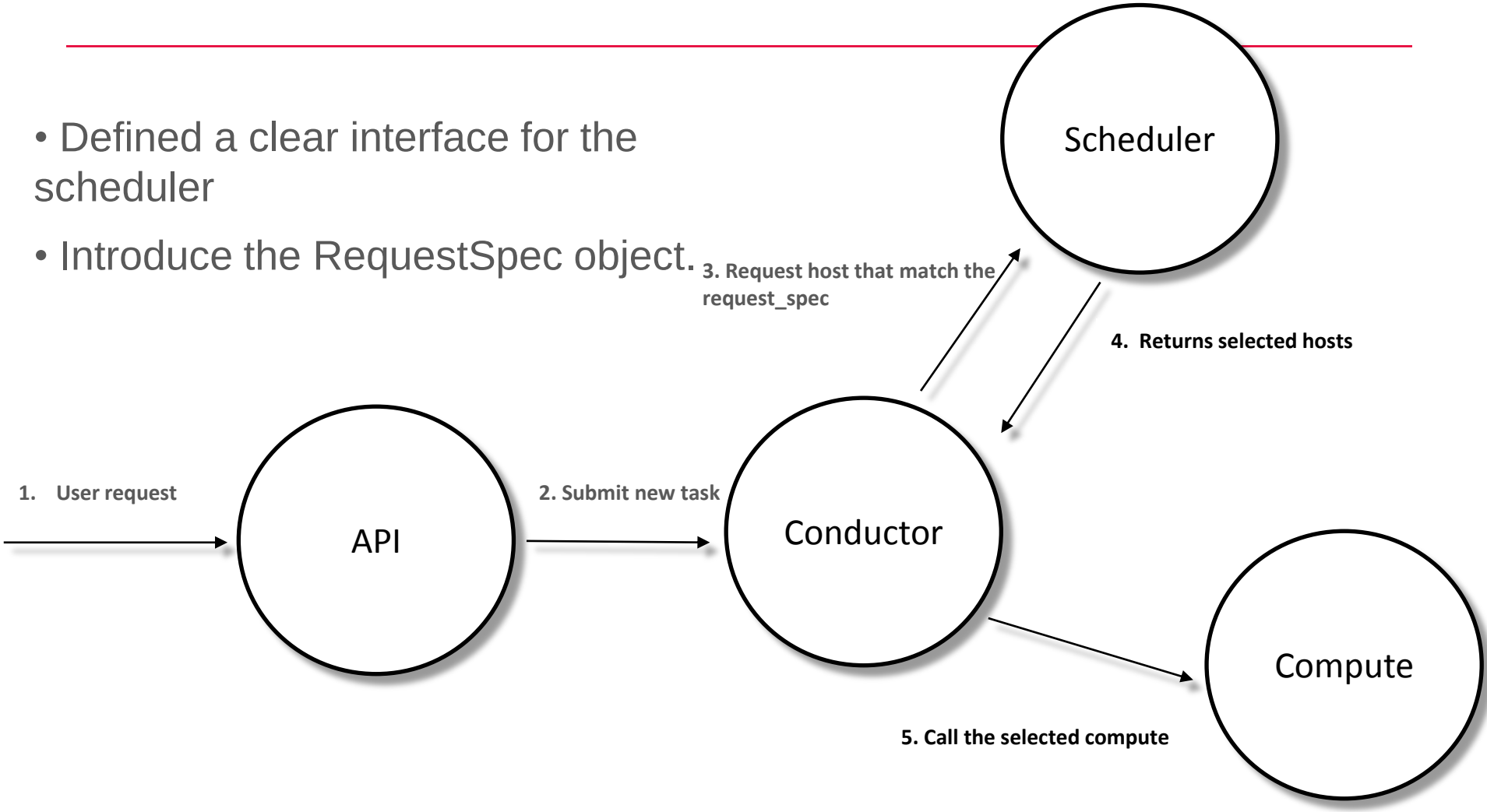
- Then begin to refactor the nova scheduler for separating it out.

Gnatt

If you heard about it…
It dead..
Why?

Then Kilo…

- Defined a clear interface for the scheduler
- Introduce the RequestSpec object.

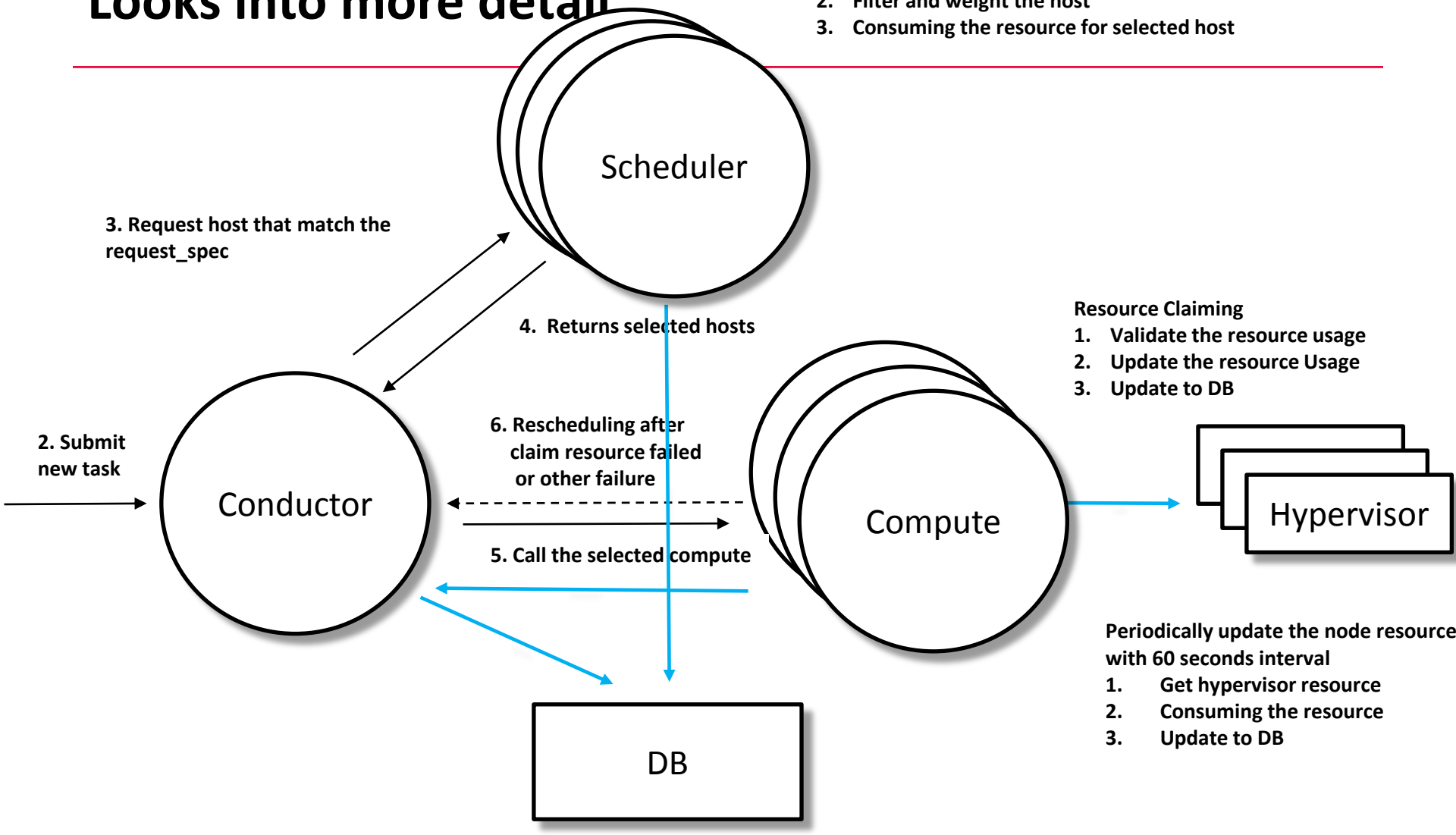


Then Liberty…

- There isn't too much progress

Looks into more detail

1. Fetch newest compute node stats for each call
2. Filter and weight the host
3. Consuming the resource for selected host



Resource Claiming
1. Validate the resource usage
2. Update the resource Usage
3. Update to DB

Periodically update the node resource with 60 seconds interval
1. Get hypervisor resource
2. Consuming the resource
3. Update to DB

The problems…

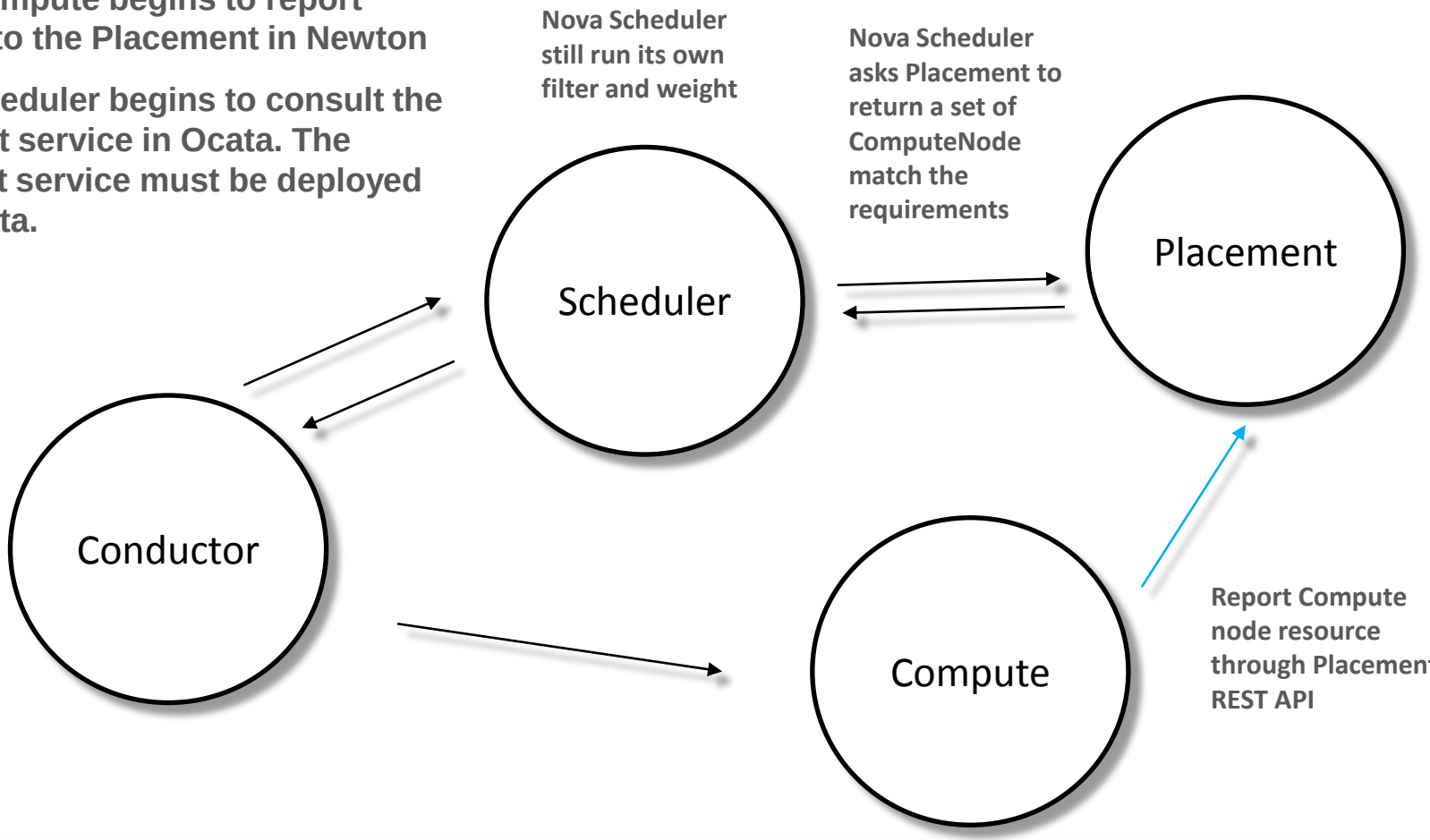
- The data model isn't extendable
  - Everything is in single `compute_nodes` table.
  - New resource means new column in the table.
- Different resource managed by different way
  - NUMA
  - PCI Devices
  - Ironic
- Flavor Extra Specs
  - Capabilities in extra specs
- Shared storage pool

The Placement Service

- Lightweight (Basically, it is just a REST API server with few data model)
  - Horizontally Scalable
  - Deploy in standard WSGI container: Runnable with Apache/Ngnix
  - REST API
- Placement service still in Nova repo
  - All the data are stored in the `nova_api` DB
  - But Nova talk with Placement with REST API
- nova-compute begins to report resource to the Placement in Newton
- nova-scheduler begins to cosult the Placement service in Ocata. The placement service must be deployed since Ocata.

The Placement Service

- nova-compute begins to report resource to the Placement in Newton
- nova-scheduler begins to consult the Placement service in Ocata. The placement service must be deployed since Ocata.



The New Data Model: Managing Quantitative and Qualitative aspects of Resource

- Resource Provider
- Resource Class
- Inventory
- Allocation
- Trait
- Aggregate

The New Data Model: Resource Provider

- A resource Pool.
- A Resource Provider can provide multiple resources.
- The Compute Node is A Resource Provider. But, it is very generic, it can be a Storage Pool, SRIOV NIC…etc
- Managing the Quantitative Qualitative aspects for the Resource with Inventory, Allocation, ResourceClass, and Trait.

The New Data Model: Quantitative Aspect: Resource Class

- The Resource Name for countable Resource
  - Counted by integer amount
- Standard Resource Class: Defined by Nova Code
  - VCPU, MEMORY_MB, DISK_GB…etc
- Custom Resource Class: Defined by Cloud Admin or Other Service.
  - Prefix with CUSTOM_
  - Ironic:
    - CUSTOM_HIGH_PERFORMANCE_BAREMETAL
    - CUSTOM_LOW_PERFORMANCE_BAREMETAL

The New Data Model: Quantitative Aspect: Inventory

- A resource provider can includes multiple inventories for different resource class
- `total`
- `allocation_ratio`
- `step_size`, `min_unit`, `max_unit`
  - You can't request 1MB memory.
  - You only can request 128MB, 256MB memory
- `reserved`: reserved resource for system

The New Data Model: Quantitative Aspect: Allocation

- Consumer indicates by UUID
- One consumer can consume resource from multiple resource providers

The New Data Model: Quantitative Aspect: Example

Let's say: ComputeNode1 has 64 VCPUs and 262144MB memory. And one instance boot up on the node, which consumed 8 VCPUs and 4096MB

ResourceProvider1 for ComputeNode1

- Inventory:
  - RP: ResourceProvider1
  - ResourceClass: VCPU
  - Total: 64
  - StepSize: 1
  - MinUnit: 1
  - MaxUnit: 62
  - Reserved: 2
  - AllocationRatio: 8

- Inventory :
  - RP: ResourceProvider1
  - ResourceClass: MEMORY_MB
  - Total: 262,144
  - StepSize: 128
  - MinUnit: 256
  - MaxUnit: 8192
  - Reserved: 16,384
  - AllocationRatio: 1

- Allocation:
  - RP: ResourceProvider1
  - ResourceClass: VCPU
  - used: 8

- Allocation:
  - RP: ResourceProvider1
  - ResourceClass: MEMORY_MB
  - used: 4096

The New Data Model: Quantitative Aspect: Trait

- Standard Traits: defined in `os-traits` library (https://github.com/openstack/os-traits)
  - HW_CPU_X86_AVX
  - HW_GPU_API_DIRECTX_V12
  - HW_NIC_OFFLOAD_TSO
  - Namespaces separated by '_': HW_CPU, HW_GPU, HW_STORAGE…
- Custom Traits: prefix with CUSTOM_

The New Data Model: Quantitative Aspect: Example

Let's say: ComputeNode1 has 64 VCPUs and 262144MB memory. And one instance boot up on the node, which consumed 8 VCPUs and 4096MB. The CPU of ComputeNode1 supports CPU features AVX and AVX2

ResourceProvider1 for ComputeNode1
With Traits:
HW_CPU_X86_AVX, HW_CPU_X86_AVX2

- Inventory:
  - RP: ResourceProvider1
  - ResourceClass: VCPU
  - Total: 64
  - StepSize: 1
  - MinUnit: 1
  - MaxUnit: 62
  - Reserved: 2
  - AllocationRatio: 8

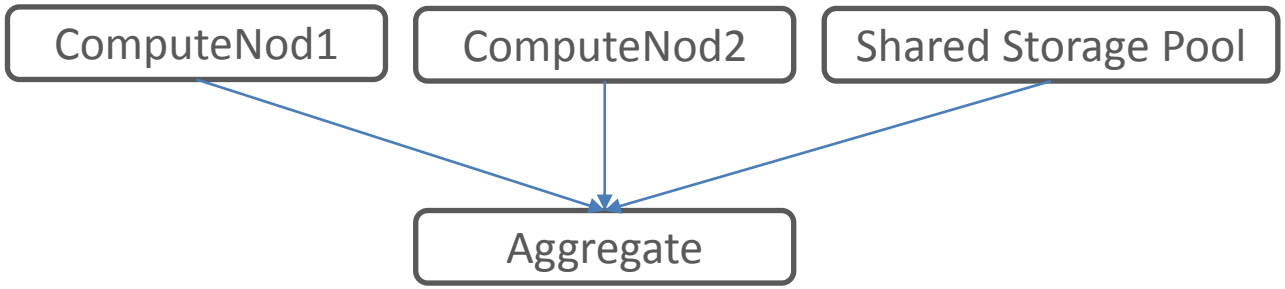
- Inventory :
  - RP: ResourceProvider1
  - ResourceClass: MEMORY_MB
  - Total: 262,144
  - StepSize: 128
  - MinUnit: 256
  - MaxUnit: 8192
  - Reserved: 16,384
  - AllocationRatio: 1

- Allocation:
  - RP: ResourceProvider1
  - ResourceClass: VCPU
  - used: 8

- Allocation:
  - RP: ResourceProvider1
  - ResourceClass: MEMORY_MB
  - used: 4096

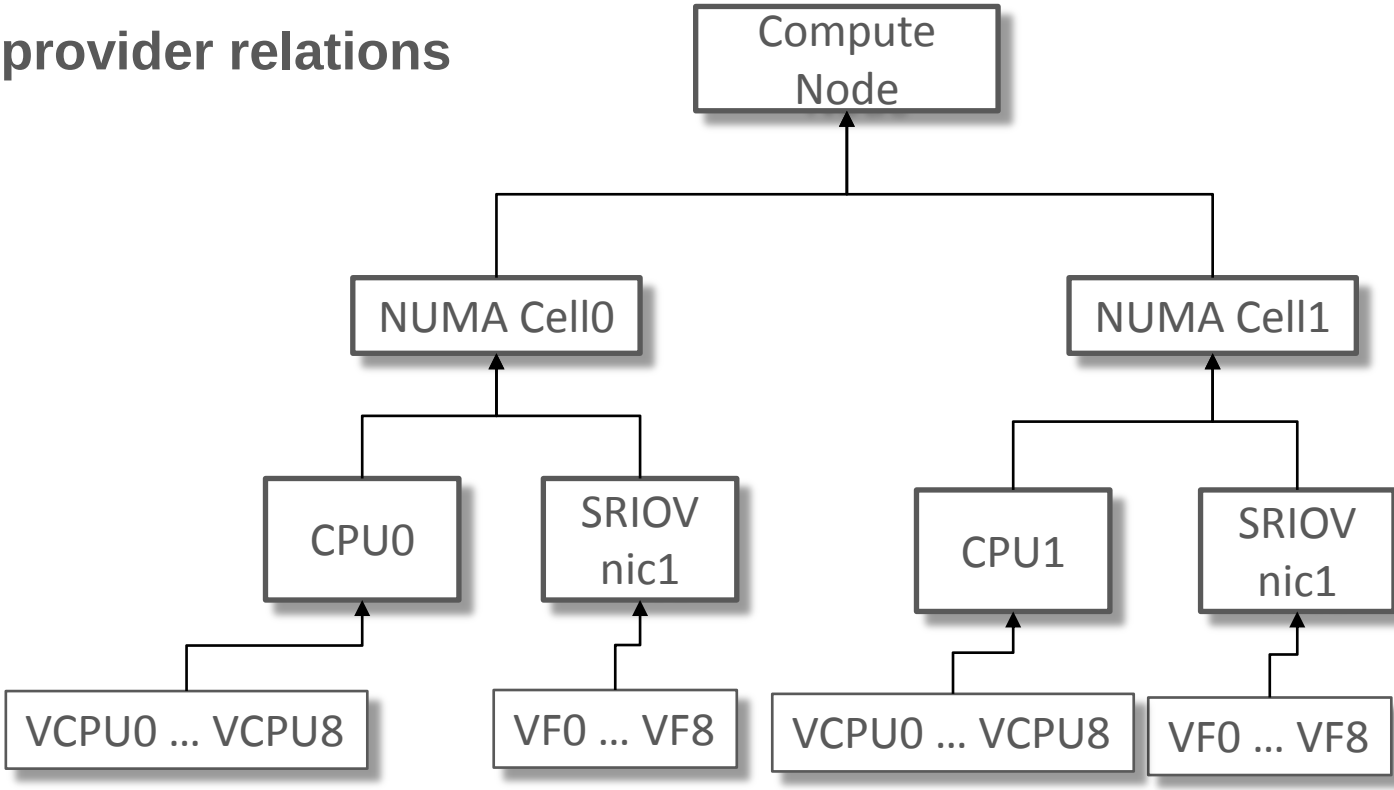
The New Data Model: Shared Resource: Aggregate

- A collection of resource provider
- The ResourceProvider provides shared resource should tag with trait 'MISC_SHARED_VIA_AGGREGATE'
- Examples:
  - Shared Storage Pool
  - Routed Network



The New Data Model: Topology: Nested Resource Provider

- Parent-Child provider relations
- NUMA
- SRIOV
- GPU
- FPGA



The status of placement development

- Traits API, Done in Pike
- The Shared Resource, Done in Pike
- The new Claim API, Done in Pike
- Claim in the scheduler, Done in Pike almostly
- Support Traits with Claim API
- Request Traits with Flavor
- Separate from Nova, in early Queens
- Nested Resource Provider, in Queens
- Notification
- Affinity/Anti-Affinity

The expected features…

- CPU features
- GPU support
- FPGA
- RDT
- Other OpenStack Services use Placement service
  - Neutron
  - Cinder
  - Ironic

THANK YOU