November 2018
Berlin

How can placement help you achieve advanced instance scheduling by integrate with different services

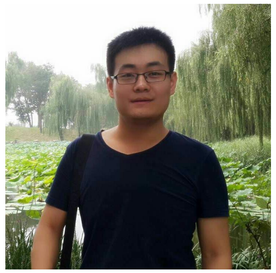
Zhenyu Zheng irc: Kevin_Zheng Huawei
Yikun Jiang irc: Yikun Huawei
Sheng Hu irc: Tommylikehu Huawei

About us

- Zhenyu Zheng

Huawei Technologies Co., Ltd.

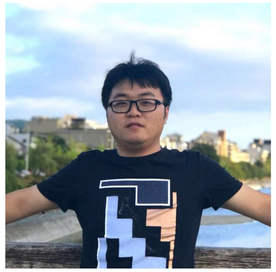
OpenStack Nova Contributor, Upstream Developer.



- Yikun Jiang

Huawei Technologies Co., Ltd.

OpenStack Nova & Cinder Contributor, Upstream Developer.



- Sheng Hu

Huawei Technologies Co., Ltd.

OpenStack Cinder Core Reviewer, Upstream Developer.

Contents

- Placement in a Nutshell
- How Nova uses Placement and what problems did it solve
- How other services interact with Nova through Placement and what feature did we achieve
- What can users expect for Stein?

Placement in a nutshell

Placement in a Nutshell - History

- Introduced in Newton(14.0.0) release as a part of Nova.
- Goal: Enable more effective accounting of resources in an OpenStack deployment and better scheduling of various entities in the cloud[1].
- A separate RESTful API stack and data model used to track resource provider inventories and usages, along with different classes of resources[1].

Placement in a Nutshell - Now

- Added a lot of features in the past 4 releases, made it more powerful and easier to use [1]:

- - Allowing define custom resource classes (Ocata)
- - Added Traits APIs in Pike and allow query RPs by Traits in Queens
- - Allowing query Allocation Candidates that are members of aggregates (Rocky)

- More and more services are starting to consider to adopt Placement.

- Placement is now going through extractions.

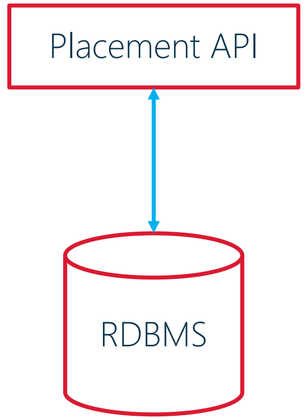
Placement in a Nutshell - Contents

- This session will focus on high-level resource abstractions and workflows.
- **Eric Fried** and **Ed Leafe** have given a very good session in Vancouver about implementation details with an very interesting example:
- https://www.openstack.org/videos/vancouver-2018/placement-present-and-future-in-nova-and-beyond

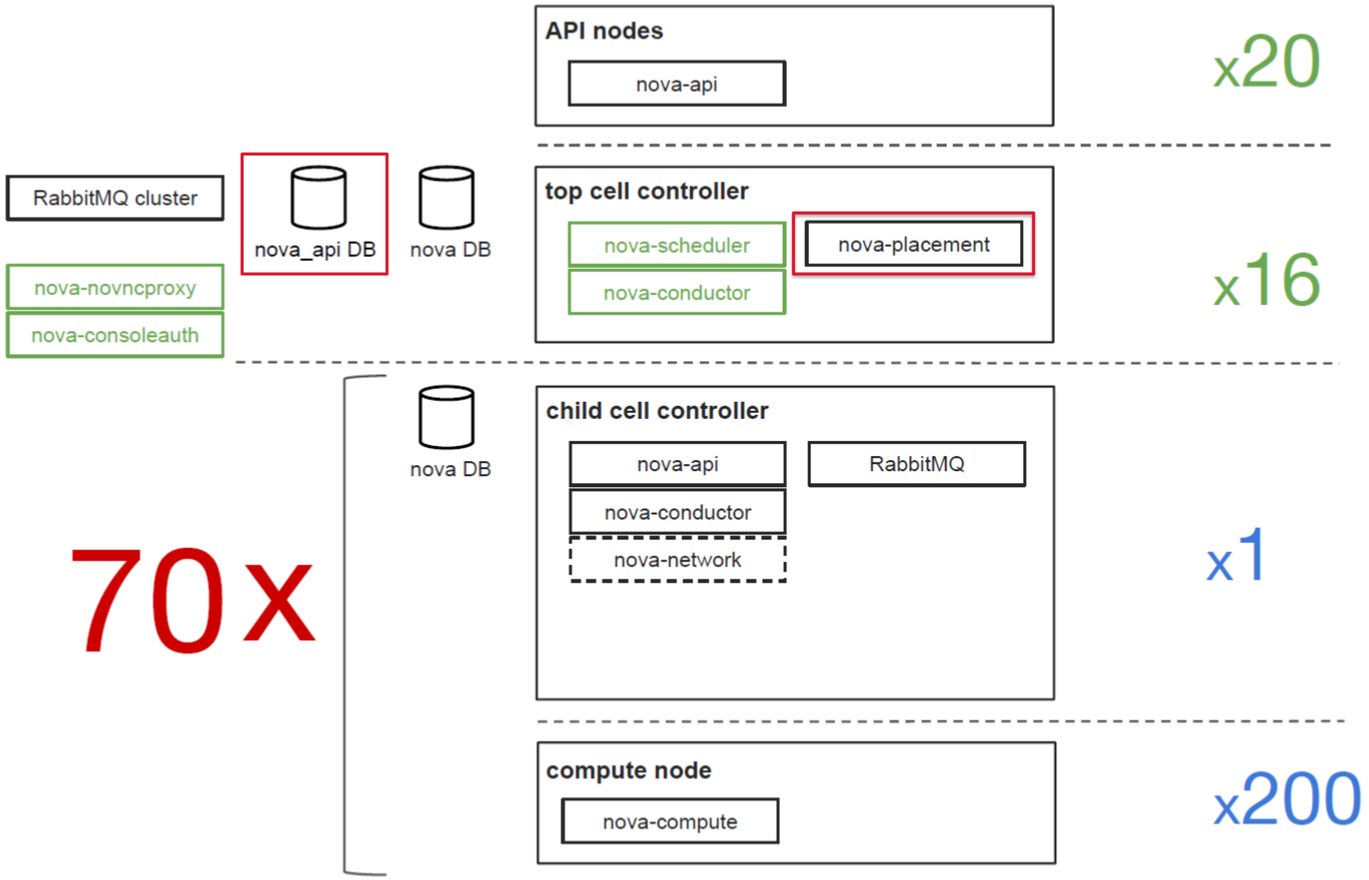
Placement in a Nutshell - From 1000 Feet

- Placement service is straightforward:
  - A WSGI application send/receive JSON requests;
  - A RDBMS for data persistence.

- State is managed solely in the DB. Thus scaling the placement service could be done by increasing the number of WSGI app instances and scaling the RDBMS using traditional database scaling techniques.



Placement in a Nutshell - Deployment



- A sample deployment at CERN[2].
- Placement deployed together with Cells V2.
- Overall 16 Placement services & 70 cells(200 compute nodes in each cell) made a successful deployment of 14,000 compute nodes in one region.

[2] https://www.openstack.org/videos/vancouver-2018/moving-from-cellsv1-to-cellsv2-at-cern

Placement in a Nutshell - Concepts

- **Resource Providers**: An abstraction data model representing the object that provides certain type/number of resources tracked by placement service, such as **compute node** & **storage pool**.

- **Resource Class**: Types of resources, there are standard resource classes (for example DISK_GB, MEMORY_MB, and VCPU) and custom resource classes (prefixed with CUSTOM_*)

- **Inventories: Quantity** of different resource classes that each resource provider can provide, for example, RP_1 has the inventory of **100 DISK_GB**, **2048 MEMORY_MB** and **8 VCPU**.

- **Traits:** Describe **qualitative** aspects of the resource provider, for example, the **DISK_GB** provided by RP_1 might be **solid state drives (SSD),** so we can set a **``is_SSD`` traits for RP_1**.

Placement in a Nutshell – Concepts

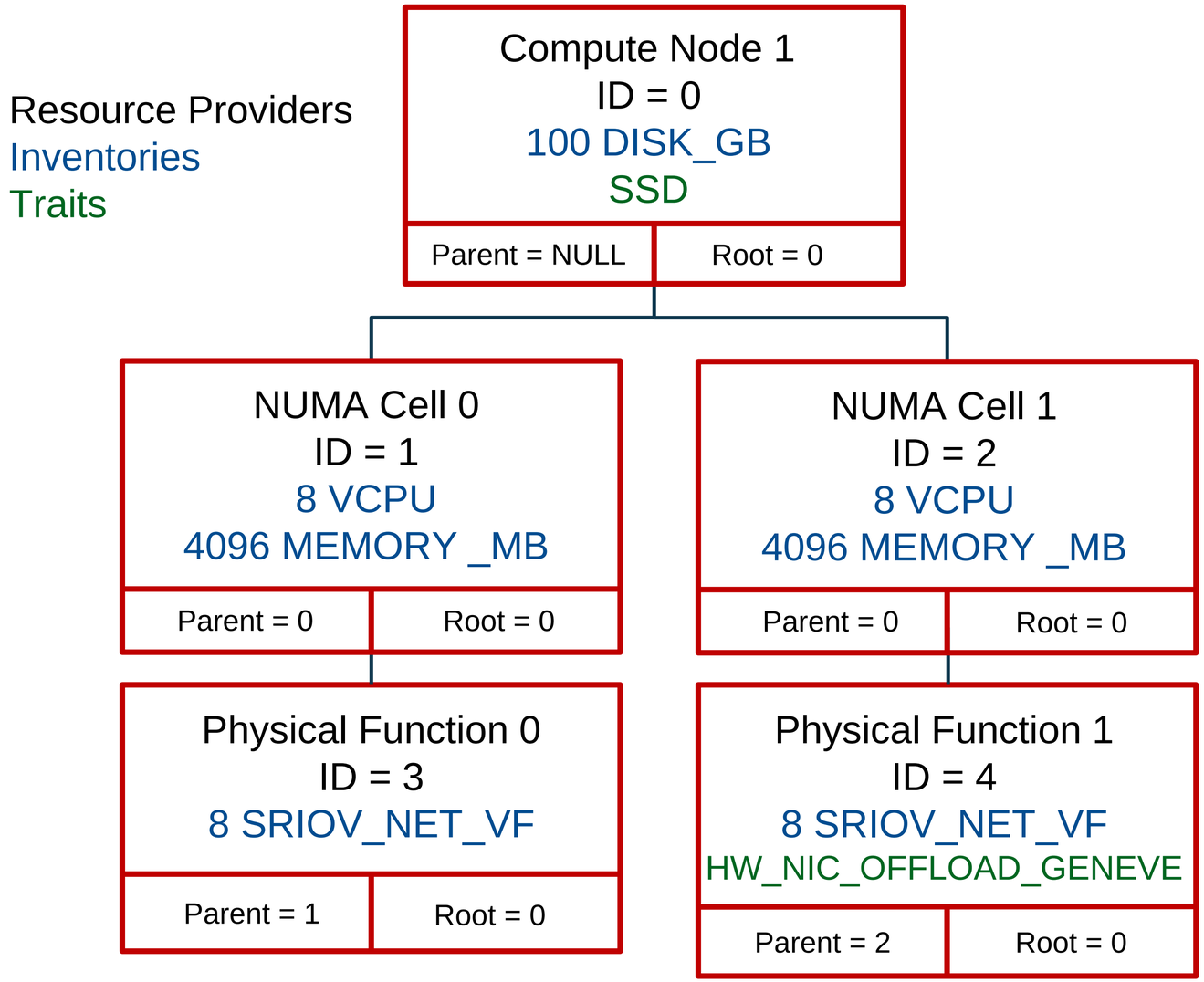
- **Consumers:** The user who occupied resources from the resource providers, for example, an instance is a consumer for RP_1(a compute node), who consumed 10 DISK_GB, 1024 MEMORY_MB and 4 VCPU.

- **Allocations:** The data model used to store the resource occupation relationship between resource providers and consumers, a typical allocation record could be consumer_1 occupied 4 unit of resource_1 from resource_provider_1.

- **Allocation Candidates:** Placement will provide a group of resource providers that are suitable for the requests, they are called the allocation candidates, callers can then use these candidates as an input for their own filter and sort process to select the best candidate.

Placement in a Nutshell – Concepts

- **Nested Resource Providers:** In Queens release, placement introduced the ability to allow hierarchical relationship between different resource providers to be represented. This is very useful for users to represent resources like NUMA nodes and SRIOV_NET_VFs.



How Nova uses Placement & what problems did it solve

Nova workflow - Problems

- **Incorrect resource usage reporting:** Due to legacy reasons, Nova considers resources are only being reported by a compute node, when reporting, Nova naively calculates resource usage and availability by simply summing amounts across all compute nodes in its database, causing number of problems.

- **Large Scale Problem:** When scheduling, Nova scheduler retrieves a list of all compute node in the entire deployment and loops through them across all the filters enabled in the deployment, it is extremely wasteful and this inefficiency gets worse the larger the deployment is.

Nova workflow - Problems

- **Cross-project scheduling:** It was very hard to leverage Nova scheduling and advanced scheduling features by other service(like routed network functionality provided by Neutron) at the same time to achieve an more advanced scheduling process. Using a more generic resource management service makes it much more easier.

Nova workflow - Report

- Compute Node reports to placement:
- Logic added in nova-compute(resource_tracker) to report its' available resources to placement as a resource provider and related inventories.
- Currently only reports VCPU, DISK_GB, MEMORY_MB and VGPU, and also starts to report some CPU features as traits[3].
- Syncing periodically and during initialize, done by request to placement with:

PUT /resource_providers/{rp_uuid}/inventories:

{

'resource_provider_generation': 66,

'inventories': {

'VCPU': 16,

...

}}

Nova workflow – Scheduling

- Nova-Scheduler gather all scheduling related parameters(such as VCPU, MEMORY_MB, DISK_GB, etc. ``Flavor Extra_Specs`` and ``image_properties`` will be translated to ``Traits`` requests.
- Nova-Scheduler will call Placement as:

- *GET*
- */allocation_candidates?*
- *resources=VCPU:1,MEMORY_MB:1024,DISK_GB:100&required=SSD*
- *OpenStack-API-Version: placement 1.10 (Maximum in Pike)*

Nova workflow – Scheduling

- A typical response will be:


```
{
   "provider_summaries": {
      "0bd25bea-5adc-4b39-ac4d-acd6e98d2439": {
         "traits": [
            "HW_CPU_X86_SSE2"...
         ],
         "resources": {
            "VCPU": {
               "used": 6,
               "capacity": 256
            },
            "MEMORY_MB": {
               "used": 5120,
               "capacity": 59568
            },
            "DISK_GB": {
               "used": 22,
               "capacity": 837
            }
         }
      }
   ...
   },
   "allocation_requests": [
      {
         "allocations": {
            "0bd25bea-5adc-4b39-ac4d-acd6e98d2439": {
               "resources": {
                  "VCPU": 1,
                  "MEMORY_MB": 512,
                  "DISK_GB": 1
               }
            }
         }
      }
   ]
}
```

Nova workflow – Scheduling

- Nova-Scheduler will then go through the enabled filters and weighers as before to select the best hosts for instance creation.

- Resource claim has been shifted to an earlier state to scheduler from nova-compute, by calling *PUT /allocations/{consumer_uuid}* of the placement API, together with all claiming resources as body. In this way we maintain a only source of the resources and avoid conflicts and reschedules.

- In current Nova, as cells v2 is used and rescheduling can only work within one cell, scheduler will select a best host and then select alternatives from the same cell.

Nova workflow – Scheduling

- When we are in a large and sparse cloud(e.g. 10,000 empty compute nodes), a request of very small instance(or very normal instance) can return 10,000 candidates. This has implication for memory and performance on both placement service and Nova-Scheduler.

- Thus in Queens we added two config options[4]:
  [scheduler]/max_placement_results, default=1000
  [placement]/randomize_allocation_candidates, default=False

- In Rocky, pre-placement filters functionality is added in the early phase of the scheduling process, users can get improved overall scheduling process. Currently support filter resource provider aggregate by project_id and/or availability zone.

How other services interact with Nova through Placement and what feature did we achieve

Workflow – Nova & Neutron Interaction

- Goal:

  Enabling instance scheduling based on the network bandwidth available in the hosts.

- Workflow:

  1. Nova creates the root RP of the compute node RP tree;

  2. Neutron creates the networking RP tree of a compute node under the compute node root RP and reports bandwidth inventories;

  3. Neutron provides the resource_request of a port in Neutron API;

Workflow – Nova & Neutron Interaction

- Workflow – cond':

4. Nova takes the ports' resource_request and includes it in the GET /allocation_candidates request;

5. Nova selects the best host from the allocation candidates and claims the resources in Placement;

6. Nova will also pass the allocation information to Neutron during port binding.

Workflow – Nova & Cyborg Interaction

- Cyborg:
- Cyborg is an OpenStack project that aims to provide a general management framework for acceleration resources, such as FPGA, ASIC, GPU etc [5].
- Launched in Pike and growing fast.
- The acceleration resources will be attached to instances as devices, thus we need Nova/Cyborg joint scheduling to make it work. This could be done with the help of Placement.

Workflow – Nova & Cyborg Interaction

- Workflow:
- Cyborg will discover, manage the accelerator resources and abstract them as resource providers in Placement;
- Like Neutron, Cyborg will report the accelerator resources as child resource providers of a compute node root resource providers;
- User will have to specify accelerator request in flavor_extra_specs, image_properties or scheduler_hints when boot instance;
- Nova scheduler will use these information during the scheduling process;
- A new ``os-acc`` lib will be used for attaching/detach process.

What can users expect for Stein ?

Stein Features

- Placement in a separate project
- NUMA Topology with Resource Providers[6]:
  - Using resource providers tree for explaning the relationship between a root RP(compute node) and one or more NUMA nodes, each of them having separate resources(Memory, PCI devices)
- Network Bandwidth RPs[7]:
  - Make network bandwidth as a child resource provider of a parent RP(compute node) to achieve bandwidth based scheduling
- Support ANY traits in allocation_candidates query[8]:
  - Able to query RPs with ANY of the required tratis
- Support filtering by forbidden aggregate[9]:
  - Able to use ``!member_of`` syntax to query RPs that are not member of the specified group

References

- [1] https://docs.openstack.org/nova/latest/user/placement.html
- [2] https://www.openstack.org/videos/vancouver-2018/moving-from-cellsv1-to-cellsv2-at-cern
- [3] https://specs.openstack.org/openstack/nova-specs/specs/rocky/implemented/report-cpu-features-as-traits.html
- [4] https://specs.openstack.org/openstack/nova-specs/specs/queens/implemented/allocation-candidates-limit.html
- [5] wiki.openstack.org/wiki/Cyborg
- [6] https://specs.openstack.org/openstack/nova-specs/specs/stein/approved/numa-topology-with-rps.html
- [7] https://specs.openstack.org/openstack/nova-specs/specs/stein/approved/bandwidth-resource-provider.html
- [8] https://specs.openstack.org/openstack/nova-specs/specs/stein/approved/placement-any-traits-in-allocation_candidates-query.html
- [9] https://specs.openstack.org/openstack/nova-specs/specs/stein/approved/negative-aggregate-membership.html

Q & A ?

openstack. HUAWEI

Thank You !