Deploying Instances On Heterogeneous Hardware Using Availability Zones in OpenStack® Compute

William Kelly, Deployment Engineer, Rackspace Hosting, Inc. (william.kelly@rackspace.com)
Joseph W. Breu, Deployment Engineer, Rackspace Hosting, Inc. (joseph.breu@rackspace.com)
1. Overview

The purpose of this technical paper is to provide an overview of OpenStack Compute scheduling options available in the Diablo (2011.3) release. We also want to demonstrate the flexibility of availability zones used in conjunction with these schedulers to provide solutions to the common scheduling concern of maintaining separate pools of hosts to provide level of service for instances with differing service requirements. Specifically, we provide examples around heterogeneous hardware pools, but the same techniques can be used to address per-project hardware guarantees, segmentation of production from development environment, and similar segregation concerns. We hope to present this information with significant technical depth to aid system administrators and system architects in their OpenStack endeavors. We will answer the following questions:

1. What is OpenStack’s definition of an availability zone?
2. How does OpenStack differentiate between a zone and an availability zone?
3. What schedulers does OpenStack support with the Diablo release?
4. How do availability zones work with the schedulers?
5. How can we use existing OpenStack capabilities to effectively schedule instances across pools of disparate hardware?

2 http://www.rackspace.com/cloud/who_uses_cloud_computing/
2. Background

In our work on the Rackspace® Cloud Builders engagements, we found our clients frequently desired to repurpose servers from other projects or from a pool of unused servers to provide the basis of their OpenStack installations. The hardware components of these servers varied significantly, causing instance performance variances based on the capabilities of the host compute node for a particular instance.

Similarly, in our clients’ production environments, some instances require specialized hardware to run optimally. For example, database instances may need to be scheduled on hosts with a greater ratio of disks per core than general purpose VMs, or a research cluster may have instances that must be scheduled on hosts that can provide GPU capabilities. Other clients require separate development and production hardware without incurring the overhead of a full OpenStack environment dedicated to each environment’s concerns.

While OpenStack provides a pluggable instance scheduling system that allows arbitrary algorithms to be used for placement of a new instance on existing compute nodes, many users prefer not to deviate from the community code when preventable. Creating a custom scheduler to resolve issues of scheduling instances appropriately across groups of disparate hardware may have ramifications for upgradability and the effectiveness of community or professional support systems.

OpenStack does provide a solution to these problems through the use of availability zones.
3. What is an availability zone?

An OpenStack availability zone is a logical partition of hosts or volume services within a single OpenStack deployment. Compute service availability zones are defined at the host configuration level, providing a method to segment compute nodes by arbitrary criteria, such as hardware characteristics, physical location, or task designation. The Simple and Zone schedulers provided by OpenStack take availability zones into account when scheduling, but the Chance and Least Cost schedulers ignore availability zones.

WHAT IS THE DIFFERENCE BETWEEN A ZONE AND AN AVAILABILITY ZONE?

In addition to availability zones, Nova has an additional concept of a more general zone, referring to a full OpenStack deployment. These zones are currently a work in progress and may see significant changes in the Essex release. The definitions we are providing are accurate with respect to the Diablo release. Within a zone, several availability zones may be defined. Zones allow logical partitioning of full Compute deployments into logical groups for load balancing and instance distribution purposes. At a minimum, a zone requires an API node, a Scheduler node, a database and RabbitMQ, but a production zone will typically contain many API, Scheduler, Volume, Network and Compute nodes in conjunction with a cluster of databases and RabbitMQ servers.

Zones are structured in parent-child hierarchies, with each zone aware of zero or more children. Zones are only aware of their direct descendents and do not keep track of their parents or their childrens’ children. Through these relationships, zones can act as request brokers: if a particular zone is not capable of servicing a request, the request may be forwarded to (child) zones for processing. Zones may be nested in a tree fashion, and a request may be propagated through an entire branch of a tree when appropriate.

Zones share nothing. They communicate via the public OpenStack API only. No database, queue, user or project definition is shared between zones. In contrast, an availability zone is a logical segregation of compute nodes within a zone that shares resources with the hosting zone. For those familiar with Amazon Web Services (AWS), OpenStack zones are roughly comparable to a Region within AWS, while OpenStack availability zones are a more general construct than Amazon’s that can be used to express similar isolation constraints or for other purposes within a zone.

Though availability zones do not directly contain knowledge of other availability zones, two of the currently available Compute schedulers are capable of accessing this information and using it as part of their scheduling process. It is important to choose an appropriate scheduler when working with availability zones, as not all provided schedulers are availability zone aware.
4. Brief overview of current schedulers

The OpenStack Diablo release supports the following scheduling algorithms:

- **Chance**
  - The Chance algorithm randomly schedules an instance without regard to zone, availability zone, or compute node utilization states. This scheduler builds a list of all compute nodes and randomly selects one for scheduling. As the Chance scheduler does not take compute node utilization into account when selecting a viable host, compute nodes that are at capacity can be selected through this random process.

- **LeastCost**
  - The LeastCost algorithm schedules instance creation on available compute nodes in the order of the weight assigned to each compute node. Weighting occurs through calling an arbitrary list of functions, specified via the 'least_cost_scheduler_cost_functions' configuration flag in nova.conf. This list should be composed of the full python paths to single arity functions (functions that take one argument), each expecting an argument of a tuple pair containing the OpenStack host and the OpenStack service and returning an integer value representing the suitability of the host for handling the given request. Two example cost functions are implemented within nova.scheduler.least_cost.

  - The default weighting function, noop_cost_fn, always returns 1.

```python
def noop_cost_fn(host):
    return 1
```

  - This results in a behavior similar to the Chance algorithm with one caveat: The LeastCost scheduler filters out nodes that are unsuitable for handling the request due to lack of resources.

  - The other provided example, compute_fill_first_cost_fn, gives higher weights to hosts with less available RAM than other hosts, allowing a scheduler that attempts to fill compute nodes prior to scheduling on unutilized compute nodes. This example provides essentially the opposite effect of the Simple scheduler.
• **Simple**
  - The Simple algorithm schedules an instance on available compute nodes ordered by the number of resources available. If an availability zone is specified, the Simple scheduler schedules the instance within the given availability zone. This process is described in detail in the next section.

• **Zone**
  - Randomly schedules an instance on a compute node within an availability zone. Similar to the Chance algorithm, the zone scheduler does not filter hosts that are incapable of running an instance due to current resource utilization out of the list of candidates.

In addition to the provided schedulers, Compute provides base classes to allow for the creation of other schedulers.
5. How do availability zones work with schedulers?

The behavior of the scheduler varies based on the schedule driver in use; however, the logic utilized to determine the compute nodes in an availability zone is consistent across all scheduling algorithms. The basic logic is:

- If the request to create an instance supplies a desired availability zone then the instance is scheduled across all compute nodes that are members of the availability zone utilizing the logic of the scheduler driver defined in nova.conf.

- If the request to create an instance does not supply a desired availability zone then the scheduler creates a list of available compute nodes within the default ‘nova’ availability zone and uses the logic of the scheduler driver to determine the host to schedule the instance on.

If the configured scheduler driver does not support availability zones (i.e., Chance and LeastCost) then the desired availability zone is ignored and the instance is scheduled for creation on a compute node per the logic of the defined driver.

To configure availability zones you must configure a scheduler driver that supports availability zones and also define a default scheduler zone on the API node as well as set:

- On the compute node modify nova.conf to set node_availability_zone and scheduler_driver flags:
  - Set --node_availability_zone to a string that represents the availability zone. i.e., --node_availability_zone=megazone
  - Set --scheduler_driver to your selected scheduler. i.e., --scheduler_driver=nova.schedulers.simple.SimpleScheduler
- restart nova-compute and nova-network.
- launching instances into a zone with euca2ools
  - euca-run-instances -z <availability_zone name> <image-id>
- launching instances into a zone using curl and the RESTful API
- Use the Identity Service, Keystone, to get an authentication token. Replace KEYSTONEHOST with the IP address used by the identity service as well as change USERNAME and PASSWORD to the username and password that you have configured in the identity service for authentication. For the purpose of this example we will be using the tenant named openstack.
The authentication token for future requests will be stored in [access][token][id] and the admin endpoints for glance, nova, and keystone are found in [access][serviceCatalog]. Additionally you can find the tenant id and name in [access][token][tenant].

- Get the id of the image that you would like to create the instance from. Replace GLANCEHOST with the IP address used by your image service and replace AUTHTOKEN with the token returned from the identity service.

For our example we will use an ubuntu 11.04 image (id=3) to create the new instance.
• Get the id of the flavor that you would like to launch. Replace NOVA-API with the IP address of your NOVA API host and replace AUTH_TOKEN with the token returned from the identity service.

[CURL EXAMPLE]

{
    "flavor": {
        "id": 1,
        "links": [
            {
                "href": "http://NOVA-API:8774/v1.1/v1/flavors/1",
                "rel": "self"
            },
            {
                "href": "http://NOVA-API:8774/v1/flavors/1",
                "rel": "bookmark"
            }
        ],
        "name": "m1.tiny",
    }
}

For our example we will use flavor m1.tiny (id=1) which will build an instance with 512MB of RAM. If you wish to get further details about the configuration of an individual flavor you can use the following:

[CURL EXAMPLE]

curl -s -H 'X-Auth-Token: AUTH_TOKEN' http://NOVA-API:8774/v1.1/1/flavors/1 | python -mjson.tool
{
    "flavor": {
        "disk": 0,
        "id": 1,
        "links": [
            {
                "href": "http://NOVA-API:8774/v1.1/v1/flavors/1",
                "rel": "self"
            },
            {
                "href": "http://NOVA-API:8774/1/flavors/1",
                "rel": "bookmark"
            }
        ],
        "name": "m1.tiny",
        "ram": 512,
        "rxtx_cap": 0,
        "rxtx_quota": 0,
        "swap": 0,
        "vcpus": 1
    }
}

• Create an instance in the ‘megazone’ availability zone. Change NOVAAPI to the IP Address of your Nova API host and replace AUTHTOKEN with the token returned from the identity service.

[CURL EXAMPLE]

curl -X POST -s -H 'X-Auth-Token: AUTHTOKEN' -H 'Content-type: application/json' \
-d '{"server":{"name":"test-server","imageRef":"3","flavorRef":"1","availability_zone": "megazone","key_name":"mykey"}}' \nhttp://NOVAAPI:8774/v1.1/servers | python -mjson.tool

{
    "server":{
        "accessIPv4": "", 
        "accessIPv6": "", 
        "addresses":{}, 
        "adminPass": "<Random Password>", 
        "config_drive": "", 
        "created": "2011-11-30T16:28:28Z", 
        "flavor": {
            "id": "1", 
            "links": [
                {
                    "href": "http://NOVAAPI:8774/v1.1/flavors/1", 
                    "rel": "bookmark"
                }
            ], 
            "hostId": "", 
            "id": "166", 
            "image": {
                "id": "3", 
                "links": [
                    {
                        "href": "http://NOVAAPI:8774/v1.1/images/3", 
                        "rel": "bookmark"
                    }
                ], 
                "key_name": "mykey", 
                "links": [
                    {
                        "href": "http://NOVAAPI:8774/v1.1/servers/166", 
                        "rel": "self"
                    },
                    {
                        "href": "http://NOVAAPI:8774/v1.1/servers/166", 
                        "rel": "bookmark"
                    }
                ], 
                "metadata": {}, 
                "name": "test-server", 
                "progress": 0, 
                "status": "BUILD", 
                "tenant_id": "openstack", 
                "updated": "2011-11-30T16:28Z", 
                "user_id": "admin", 
                "uuid": "2499e2de-a249-4f46-9a66-8a9c6e7b6d0e"
            }
        }
    }
}
• Launching instances into a zone using Fog (github.com/fog)

[RUBY CODE EXAMPLE]

```ruby
#!/usr/bin/ruby
require 'rubygems'
$: << 'fog/lib'
require 'fog'

# set the following to match your environment
keystone = "http://KEYSTONEHOST:5000/v2.0/tokens"
user = "admin_user"
pass = "admin_password"
tenant = "tenant_name"
zone = "zone_name"
flavor_id = "flavor_id"
image_id = "image_id"

# create the connection
compute = Fog::Compute.new(:provider => 'OpenStack',
                           :openstack_auth_url => keystone,
                           :openstack_username => user,
                           :openstack_api_key => pass,
                           :openstack_tenant => tenant)

# launch an instance
server = compute.servers.new(:flavor_ref => flavor_id,
                              :image_ref => image_id,
                              :name => "Test Image",
                              :availability_zone => zone)
server.save
```
6. One solution to the heterogeneous hardware problem

Hosts can be categorized into availability modes according to their performance characteristics. In the case where there are two tiers of hardware—a general tier that is lower powered, and a special tier that is higher powered and reserved for instances that require higher performance—we can configure Compute nodes in the general tier in an availability zone ‘nova’ and Compute nodes in the higher tier to the availability zone ‘high’.

When you use an availability zone aware scheduler such as SimpleScheduler, API users are guaranteed that instances that require access to the higher tier of hardware can be scheduled appropriately by specifying the ‘high’ availability zone. At present, if SimpleScheduler is used, an availability zone should be specified with each request to prevent instances from being created inside of protected availability zones. If ZoneScheduler is used, default_schedule_zone should be set to ‘nova’, and the availability zone can be omitted from requests for instances intended for the general tier.

CAVEATS

The default behavior of the schedulers provided by Compute is to schedule requests for instances with unspecified availability zones across all compute nodes, regardless of their availability zone membership. This behavior can be changed in the ZoneScheduler by specifying a default_schedule_zone on all API nodes within a zone. With this flag set, instances will be scheduled within the default_schedule_zone when requests do not explicitly specify an availability zone. The SimpleScheduler does not provide configuration options for the default schedule zone at this time (patch pending).

If all nodes in an OpenStack zone are set to an availability zone other than the default schedule zone, the ZoneScheduler fails to find an available compute node for instance creation on requests that do not explicitly set availability zones as there are no nodes left with unallocated resources.
7. Conclusion

Availability zones can be a useful tool to segregate heterogeneous hardware within an OpenStack zone. Utilizing availability zones allows the user to define a group of compute nodes that will be used for specialized purposes (such as database servers, web servers, or pure computation nodes) and separate those functions from other instances running in the zone.

REFERENCES

Zone Scheduling
http://nova.openstack.org/devref/distributed_scheduler.html

The Chance Scheduler

The Simple Scheduler

Pending patch for Simple Scheduler in Diablo: https://review.openstack.org/#change,1986
DISCLAIMER

This document is for informational purposes only and is provided “AS IS.” The information set forth in document is intended as a guide and not as a step-by-step process. We strongly recommend that you engage additional expertise in order to further evaluate applicable requirements for your specific environment.

RACKSPACE MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND, EXPRESS OR IMPLIED, AS TO THE ACCURACY OR COMPLETENESS OF THE THIS DOCUMENT AND RESERVES THE RIGHT TO MAKE CHANGES TO SPECIFICATIONS AND PRODUCT/SERVICES DESCRIPTION AT ANY TIME WITHOUT NOTICE. RACKSPACE RESERVES THE RIGHT TO DISCONTINUE OR MAKE CHANGES TO ITS SERVICES OFFERINGS AT ANY TIME WITHOUT NOTICE. USERS MUST TAKE FULL RESPONSIBILITY FOR APPLICATION OF ANY SERVICES AND/OR PROCESSES MENTIONED HEREIN. EXCEPT AS SET FORTH IN RACKSPACE GENERAL TERMS AND CONDITIONS, CLOUD TERMS OF SERVICE AND/OR OTHER AGREEMENT YOU SIGN WITH RACKSPACE, RACKSPACE ASSUMES NO LIABILITY WHATSOEVER, AND DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO ITS SERVICES INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NONINFRINGEMENT.

ALTHOUGH PART OF THE DOCUMENT EXPLAINS HOW RACKSPACE SERVICES MAY WORK WITH THIRD PARTY PRODUCTS, THE INFORMATION CONTAINED IN THE DOCUMENT NOT DESIGNED TO WORK WITH ALL SCENARIOS. ANY USE OR CHANGES TO THIRD PARTY PRODUCTS AND/OR CONFIGURATIONS SHOULD BE MADE AT THE DISCRETION OF YOUR ADMINISTRATORS AND SUBJECT TO THE APPLICABLE TERMS AND CONDITIONS OF SUCH THIRD PARTY. RACKSPACE DOES NOT PROVIDE TECHNICAL SUPPORT FOR THIRD PARTY PRODUCTS, OTHER THAN SPECIFIED IN YOUR HOSTING SERVICES OR OTHER AGREEMENT YOU HAVE WITH RACKSPACE AND RACKSPACE ACCEPTS NO RESPONSIBILITY FOR THIRD-PARTY PRODUCTS.

Except as expressly provided in any written license agreement from Rackspace, the furnishing of this document does not give you any license to patents, trademarks, copyrights, or other intellectual property.

Rackspace, Rackspace logo, Fanatical Support, and/or other Rackspace marks mentioned in this document are either registered service marks or service marks of Rackspace US, Inc. in the United States and/or other countries. OpenStack™ and OpenStack logo are either registered trademarks or trademarks of OpenStack, LLC in the United States and/or other countries.

All other product names and trademarks used in this document are for identification purposes only to refer to either the entities claiming the marks and names or their products, and are property of their respective owners. We do not intend our use or display of other companies’ tradenames, trademarks, or service marks to imply a relationship with, or endorsement or sponsorship of us by, these other companies.

© 2012 Rackspace US, Inc. All rights reserved.