VM Ensembles

A unit or group of complementary parts that contribute to a single effect

Gary Kotton[1](VMware) and Gilad Zlotkin[2](Radware)

Introduction

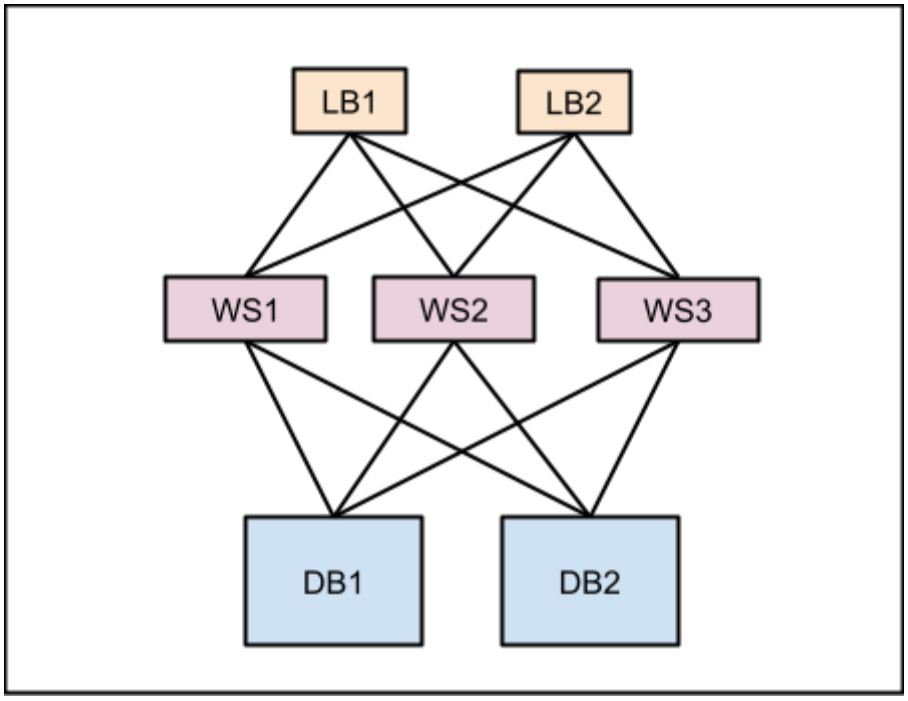
This document introduces the concept of a VM ensemble or VM group into Nova. An ensemble will provide the tenant the ability to group together VMs that provide a certain service or part of the same application. More specifically it enables configuring scheduling policies per group. This will in turn allow for a more robust and resilient service. Specifically, it will allow a tenant to deploy a multi-VM application that is designed for VM fault tolerance in a way that application availability is actually resilient to physical host failure. This is described in the example below.

A tenant is deploying a WEB service. This service has the following components in a fault resilient layout:

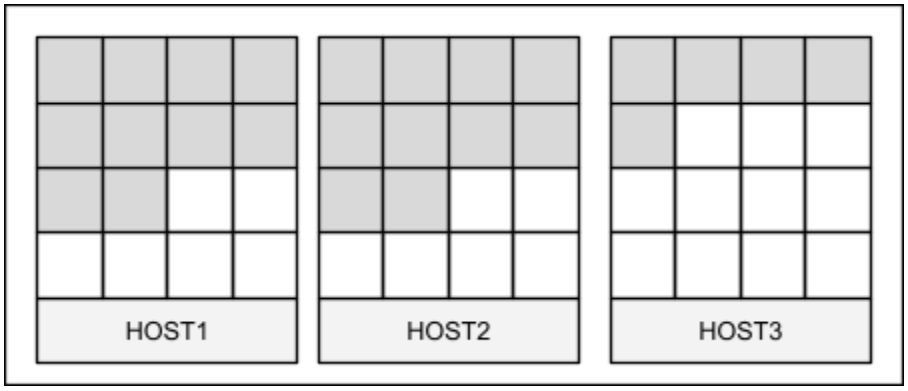
- Load balancers - these VM's will be used to distribute the load amongst the WEB servers. This provides fault resiliency for the WEB servers. (LB***i***). Each load balancer requires 1 capacity unit.
- WEB servers (WS***i***). Each web server requires 2 capacity units.
- Back end databases (DB***i***). Each database requires 4 capacity units.

[1] Gary Kotton is a Staff Engineer at VMware - gkotton@vmware.com

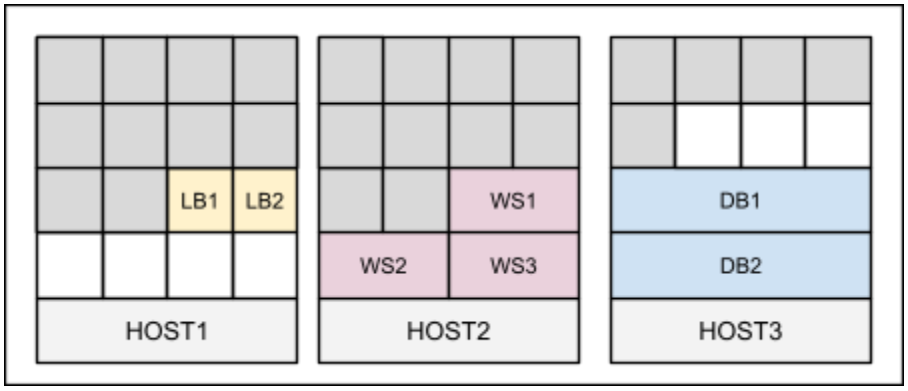
[2] Gilad Zlotkin is a VP Virtualization and Management at Radware - giladz@radware.com



The tenant is able to make use of host aggregates, zones or availability zones for different placement strategies for the VM's. Lets take for example a target host aggregate with 3 Hosts: HOST1 has 6 free compute units, HOST2 has 6 free compute units and HOST3 has 11 free compute units.



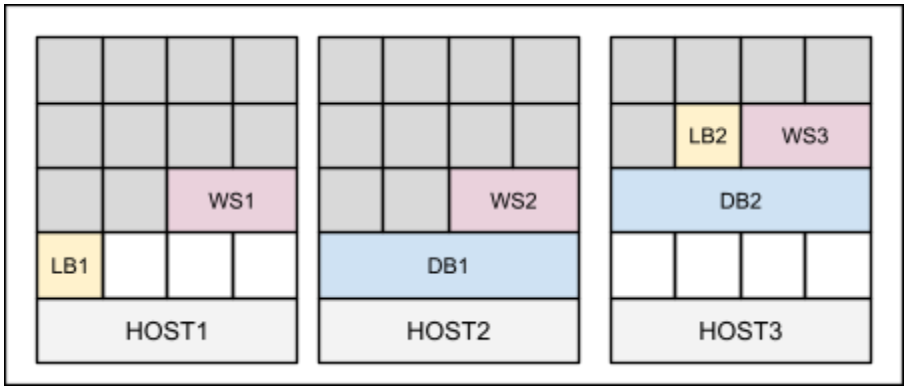
The placement of the VMs could be as follows:



The problem is that if any of the hosts go down, then the tenant's service will no longer be up and accessible.
This is where the notion of the ensembles comes into play. The tenant will be able to define the placement strategies for the groups, for example:

- LBG = Anti Affinity
- WSG = Anti Affinity
- DBG = Anti Affinity

In the example above the placements of the VM's will now be:



This means that the service is actually fault tolerant as designed, i.e. if one of the HOSTs goes down then the service will still be up and running. The ensembles will enable a service to be tolerant to host failures in an availability zone or a host aggregate.

Host server failure tolerance is different from site failure tolerance. Availability zone is a mechanism for managing site failure tolerance which typically requires twice of the required compute capacity - one in each site. Host failure resiliency typically requires only N+1 compute resources instead of 1+1 (or N+N). Host failure is a more common event in the cloud than an availability-zone/site failure. Mission critical application deployment in the cloud can be designed to be resilient for either or both types of failures.

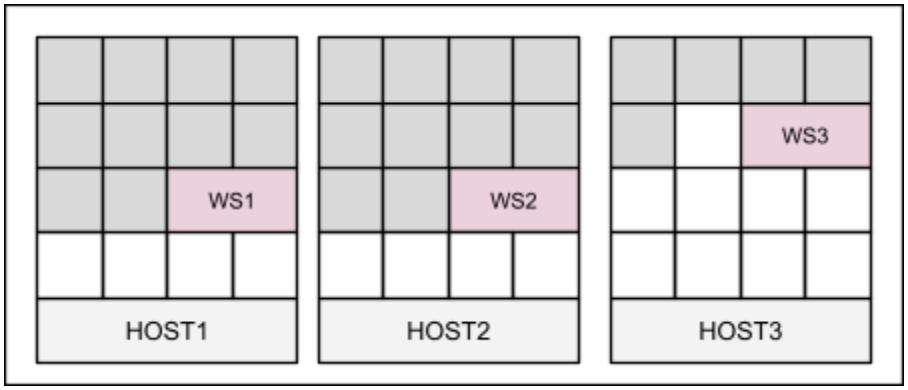
Implementation

The current Nova scheduler is admitting one VM. There is an option for passing scheduler hints, for example DifferentHost. In addition to this one can deploy same VM N times, this does not have the ability to use the scheduler hint for the DifferentHost. VM ensembles will need to be scheduled as a group. Otherwise, if we try to schedule them one VM at a time, then we are risking a false negative admission decision.

To demonstrated this, let us schedule VMs from the group one at the time.

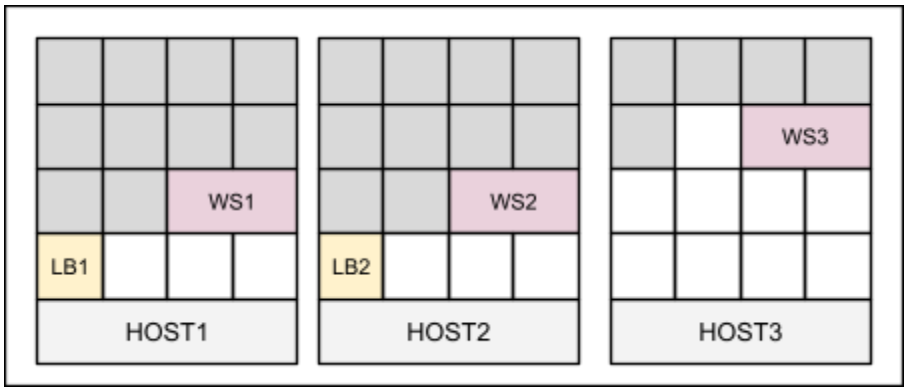
Starting by scheduling the 3 WS VMs, the Nova scheduler will take the anti-affinity property into account and will place them one after the other as follows:

nova boot --image ws.img --flavor 2 WS1
nova boot --image ws.img --hint different_host=[WS1] --flavor 2 WS2
nova boot --image ws.img --hint different_host=[WS1,WS2] --flavor 2 WS3



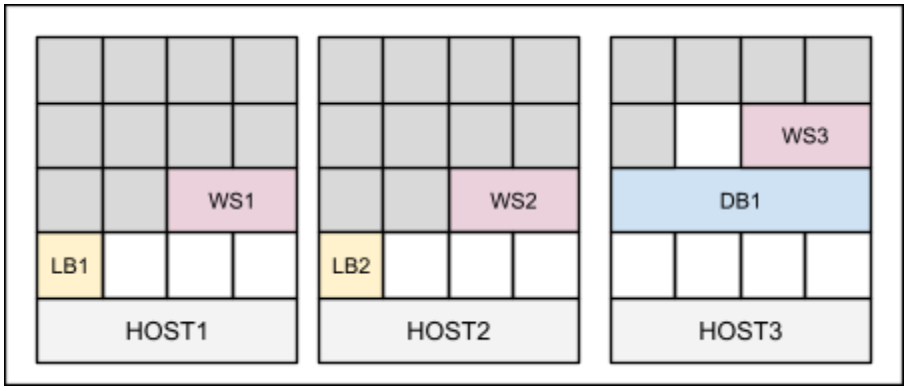
Scheduling the two LB VMs next, will get the Nova Scheduler to do the following placements:

nova boot --image lb.img --flavor 1 LB1
nova boot --image lb.img --hint different_host=[LB1] --flavor 1 LB2



Trying to schedule the DB1 VM next will succeed:

nova boot --image db.img --flavor 3 DB1

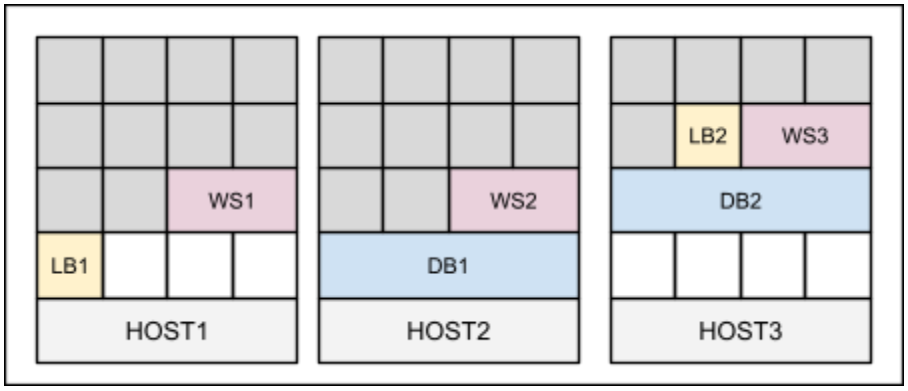


However, DB2 cannot be admitted, as the only HOST3 has 4 free slots. But, that would violate the anti-affinity constraints with DB1.

nova boot --image db.img --hint different_host=[DB1] --flavor 3 DB2

Therefore, the above call will fail and the Nova scheduler will reject admitting the DB2 VM, which practically means rejecting the admission of the entire multi-VM application.

This would be a false negative admission decision, as the application can be admitted as follows:



As the cloud compute capacity allocation percentage is higher, more likely we will suffer from false admission control decisions. Unless, we schedule the entire VM ensemble as a group. Cloud service providers are striving to maximize compute capacity percentage allocated to cloud tenants. For example, AWS is offering a special EC2 Spot Instances (http://aws.amazon.com/ec2/spot-instances/) to utilize any available capacity unit, such that capacity allocation will be maximized. Please note that even when capacity allocation on a given

host is maximized, it does not mean that CPU utilization is maximized, since not all VMs on that host are likely to utilize their vCPU quota at the maximum level at the same time. Therefore, it is safe to drive capacity allocation percentage to the 90%-100% level.

Conclusion: The Nova scheduler will need to schedule VM ensembles as a group.

Interface

In order for the scheduling to take place after all of the group members are known a new flag will be added to nova boot.

usage: nova boot [--flavor <flavor>] [--image <image>]
[--num-instances <number>] [--meta <key=value>]
[--file <dst-path=src-path>] [--key-name <key-name>]
[--user-data <user-data>]
[--availability-zone <availability-zone>]
[--security-groups <security-groups>]
[--block-device-mapping <dev-name=mapping>]
[--hint <key=value>]
[--nic <net-id=net-uuid,v4-fixed-ip=ip-addr,port-id=port-uuid>]
[--config-drive <value>]
[--poll]
[--ensemble <ensemble-name>]

The ensemble name is unique per user/tenant. This enables the scheduler to perform the group scheduling.

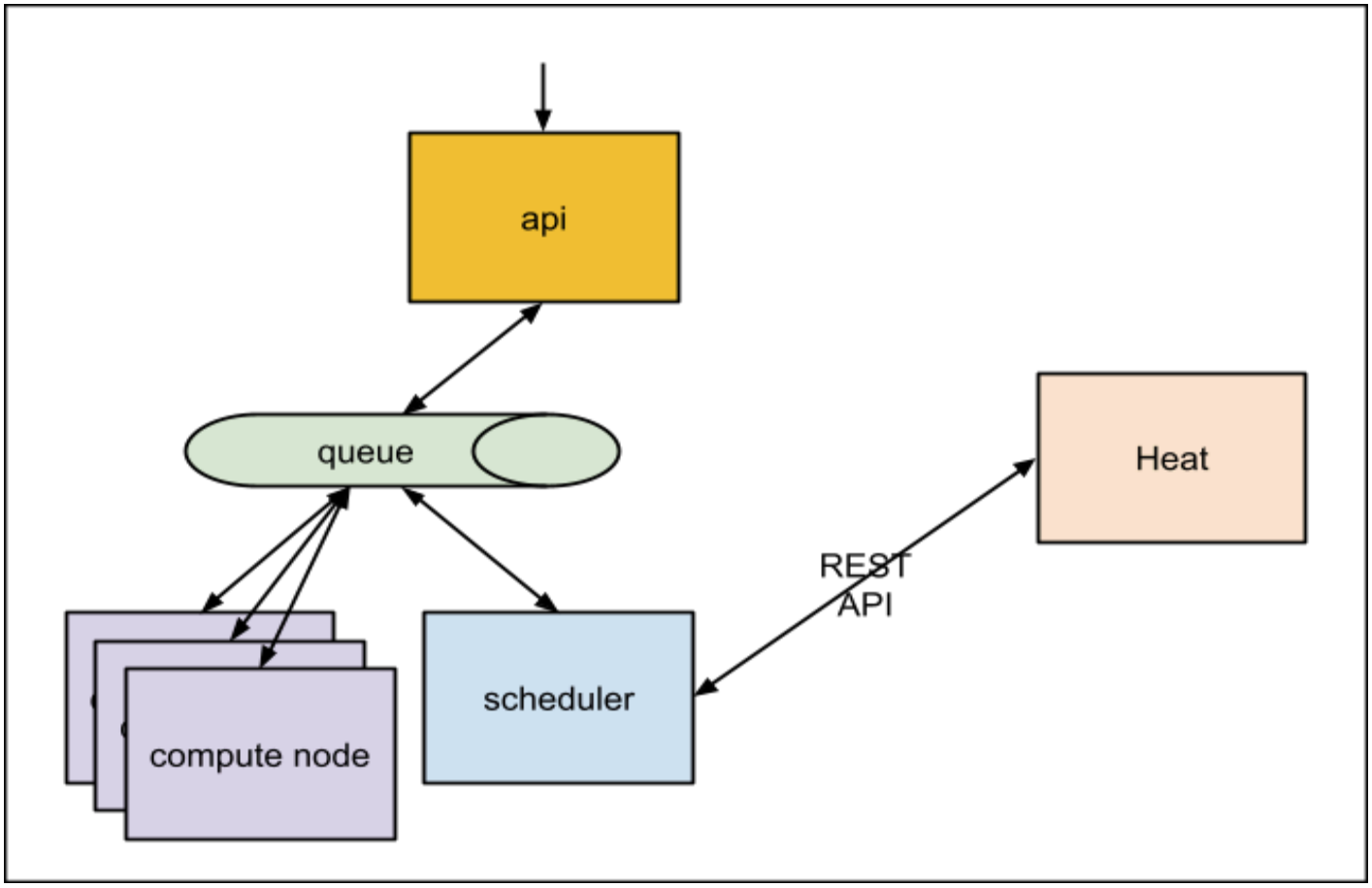
In order for the scheduling groups to be supported we suggest the support:

- Scheduler receives the group from a ensemble endpoint
- The tenant is able to pass the ensemble to Nova

Ensemble Endpoint

A ensemble endpoint provides the group information. This is done as follows:

1. The scheduler will contain a driver that will interface with the ensemble endpoint, for example Heat (http://wiki.openstack.org/Heat). This will require authentication and authorization information so that the scheduler is able to retrieve the relevant information from Heat.
2. When the scheduler receives a new scheduling request and the request contains a ensemble reference then the scheduler will interface with the third party to receive the group data. This is illustrated in the diagram below:



3. The scheduler will send a REST request to Heat to return the VM's in the ensemble that are to be scheduled. The API will be defined as follows:

GET /v1.0/ensembles/{name}.json

This will return a dictionary of the ensemble (the format is below).

Open Issues

1. At the moment Heat cannot work with a bulk deploy. This can be addressed by Heat deploying multiple ensembles. Using the example above there may be two calls:

nova boot --ensemble=DBG --ensemble=WSG

nova boot --ensemble=LBG

2. At the moment a considerable amount of data is passed from Heat to the nova boot. It may be worthwhile considering expanding the ensemble API to retrieve specific VM data prior to deploying the VM. For example:

GET /v1.0/ensembles/{name}/VM/{id}.json

Nova Ensemble Support

An API will be added to Nova to enable the user to pass a ensemble to be booted. This will be achieved by adding in a extension call Ensemble_create similar to Multiple_create. This will

enable the tenant to pass in the ensemble information. The API will enable the user to pass the following information through to the scheduler. This will be a dictionary (format defined below). The Nova CLI will need to be updated to enable the user to create a Ensemble.

Ensemble Data Format

The ensemble data format, either received from a endpoint or passed to the scheduler by the tenant will be of the following format:

{‘ensemble’: <name/id>,
‘type’: <scheduling type - ‘anti-affinity’>,
‘sub_groups’: [list of subgroups]}

Each **subgroup** is a dictionary that has the following format:

{‘name’: <unique name>,
‘type’: <scheduling type - ‘anti-affinity’>,
‘vms’: [list of VM’s to be scheduled]}

Where the list of VM’s to be scheduled has the following format (this is essentially the data that can be passed by a “nova boot” for a specific VM):

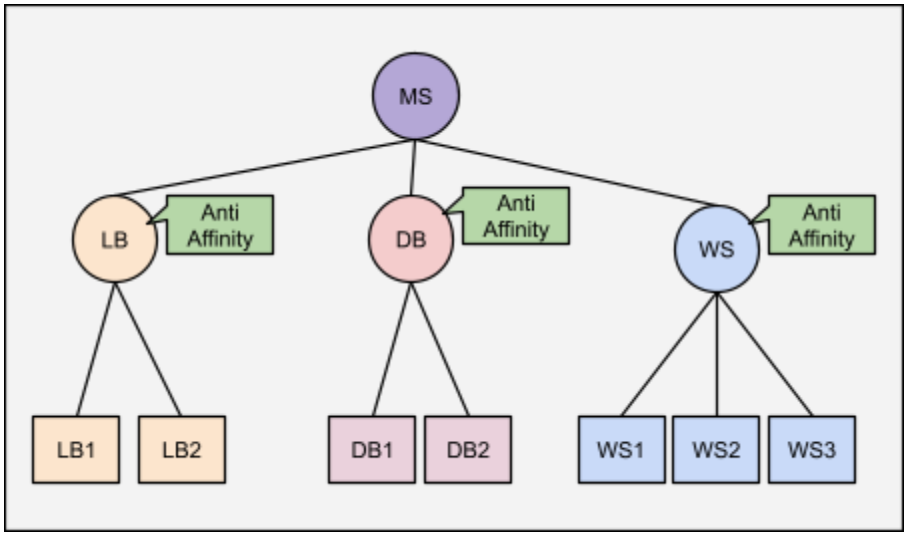
{‘name’: <name>,
‘flavor’: <flavor>,
‘image’: <image> etc.}

Example

The example below will demonstrate the usage of the API for the service discussed above. More specifically it will be deploying a ensemble, called MS, consisting of:

- 2 load balancers, called LBG, using image lb.img
- 3 web servers, called WSG, using image ws.img
- 2 database servers, called DBG, using image db.img

The diagram below demonstrates the ensemble and the subgroups.



Boot the ensemble
nova boot --ensemble=MS