Provisioning Large-Memory VMs on Large-Memory Hosts in Cadence Cloud

Solution Approach :

1. Objective

The goal of this solution is to ensure that virtual machines requiring high memory, such as 128 GB RAM or more, are provisioned only on physical servers that are purpose-built to handle such workloads. This improves application performance, avoids resource contention, and ensures proper placement of memory-intensive workloads in OpenStack.

2. Responsibility Division

The Infrastructure Team is responsible for identifying large-memory compute hosts, creating host aggregates, tagging those aggregates with appropriate metadata, and defining large-memory flavors with corresponding metadata. The Application or DevOps team is responsible for provisioning VMs using the correct flavor. The Nova scheduler will automatically apply matching filters based on metadata to ensure correct placement.

3. Solution Overview

The proposed approach leverages OpenStack's host aggregate mechanism and flavor metadata matching. The Infrastructure Team first groups the physical compute hosts that have large memory capacity into a host aggregate. Metadata is applied to this aggregate, for example, large\_mem=true.

Next, a VM flavor is created for large-memory VMs. This flavor includes a matching metadata field in the extra\_specs section, such as aggregate\_instance\_extra\_specs:large\_mem=true.

When the Application team launches a VM using this flavor, the Nova scheduler automatically filters out any compute nodes that do not belong to the large-memory host aggregate. The scheduler only considers nodes where the metadata matches and that have sufficient available memory and CPU capacity.

This guarantees that large-memory VMs will not be scheduled on standard or low-capacity compute hosts.

4. Example Configuration

Host aggregate name: agg-large-memory  
Host aggregate metadata: large\_mem=true  
Compute nodes assigned: compute-node-03 and compute-node-04

Flavor name: m2.large-mem  
Flavor specs: 128 GB RAM, 16 vCPUs, 100 GB disk  
Flavor extra specs: aggregate\_instance\_extra\_specs:large\_mem=true

When a VM is launched using the m2.large-mem flavor, the Nova scheduler will only consider compute-node-03 and compute-node-04. If either has enough available resources, the VM is placed there.

5. Affinity and Anti-Affinity Options

This approach supports further enhancement through affinity or anti-affinity rules. If multiple high-memory VMs must reside on the same host for performance reasons, the user can enable affinity using a server group. If VMs need to be distributed across hosts for high availability, anti-affinity rules can be applied.

Anti-affinity policies require that more than one eligible host be available in the aggregate. Otherwise, the scheduler may fail to place the VM.

6. Terraform Flavor Example

resource "openstack\_compute\_instance\_v2" "large\_memory\_vm" {  
name = "vm-large-memory-01"  
image\_name = "rhel9-latest"  
flavor\_name = "m2.large-mem"  
key\_pair = "default-key"

network {  
name = "private-network"  
}

security\_groups = ["default"]

metadata = {  
usage\_type = "large-memory"  
}

tags = ["large-mem"]  
}

This Terraform block provisions a VM using the pre-defined large-memory flavor that matches the host aggregate.

7. Benefits and Trade-Offs

This approach ensures that large-memory workloads are placed only on qualified servers, providing predictable performance and avoiding oversubscription of general-purpose hosts. It leverages native OpenStack features without custom development.

However, this approach requires upfront configuration of aggregates and flavors, and relies on correct metadata tagging. If no matching host is available, VM creation may fail. Automation and monitoring are important to ensure consistent behavior.

8. Conclusion

This solution provides a scalable, controlled method for handling high-memory VM placement using OpenStack's built-in features. It separates responsibilities between infrastructure and application teams, ensures intelligent scheduling via metadata, and supports advanced policies such as affinity and anti-affinity. It is production-ready and supports automation via Terraform or CI/CD pipelines.