VIRTUAL MACHINE PLACEMENT STRATEGIES FOR VIRTUAL NETWORK FUNCTIONS

Adam Grochowski
Juniper Networks

Abstract

Current Virtual Machine (VM) scheduling services, such as Openstack's Nova only have awareness of the CPU, RAM, and storage utilization from a hypervisor perspective. While this has proven to be sufficient for traditional cloud applications and their associated workloads, as they are rarely limited by network bandwidth, Virtual Networks Functions (VNFs) by comparison require fairly static and known quantities of CPU, memory and storage. However, properly placing their workloads depends upon a knowledge of hypervisor network resources and external network topology.

This paper will compare and contrast traditional cloud workloads and their placement with VNF workloads and their respective placement. We will discuss why theoretical extensions to cloud management software are needed to improve placement, quality of service and utilization of commodity hardware.

VNF/NFV BACKGROUND

Network Functions Virtualization (NFV) is an effort originally proposed by the European Telecommunications Standard Institute (ETSI) at the SDN and OpenFlow World Congress in 2012. [1]. The objective or focus of the NFV initiative, is to use standard virtualization technology to consolidate various network equipment types onto a common platform that can be easily distributed throughout the network. The NFV platform comprises high capacity servers, switches and storage, which will utilize software applications running on virtual machines to perform functions like routing, switching and security. These platforms can then be located in the datacenter, head end, or on the end users' premises.

Service Providers will benefit from NFV in the following ways. First and foremost, NFV provides for the ability to increase agility and the velocity of new service deployment, improving time to market. Virtualizing network equipment improves the ability to automate, which allows for faster service instantiation. NFV can help operators by reducing the required initial investment in equipment cost and power consumption as many network functions can be combined onto standard servers. As in traditional cloud application environments, services can be more quickly and even automatically scaled to meet increased demand.

With NFV service providers now have the ability to create virtual devices providing network services on demand that can be dynamically inserted into the end-user's path.

CURRENT WORKLOAD SCHEDULING

The industry has been moving toward Openstack as the standard for managing cloud computing platforms for private and many public cloud installations. Therefore, because Openstack is becoming the de-facto standard and it shares many things in common with other cloud management packages, this paper will use Openstack and its components as a representative example of cloud management software.

Before we dive into how cloud workload scheduling works, it is important to understand how cloud management platforms are structured. Openstack, as an example, comprises several sub projects, which represent the required functions or systems that manage the different aspects of a cloud installation [2]. These projects and their codenames are listed below:

Network – Neutron
Compute – Nova
Authentication – Keystone
UI – Horizon
Storage – Swift, Cinder
Image Service – Glance
Telemetry – Ceilometer, Gnocchi

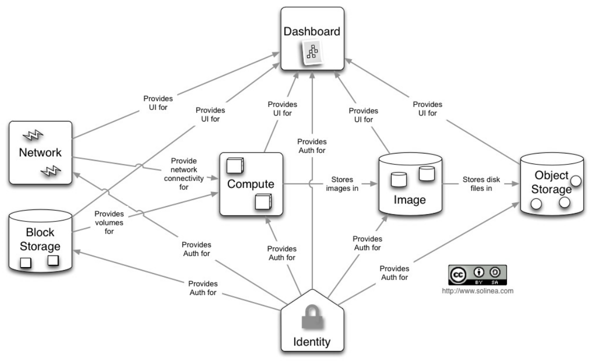

Figure 1: Openstack components. Source: openstack.org

All of these subprojects work in concert to orchestrate the placement and instantiation of a virtual machine as well as its associated network, storage, and other required resources. This placement is broadly based on the following: Availability zone, available RAM, storage, compute availability, custom filters (NB: additional items exist but for the purposes of this paper the most common metrics are referenced). When a user issues a request for a new virtual machine be created, the scheduler will check for available resources, select the appropriate hypervisor to service the request, and trigger the VM creation/boot on the selected host from a virtual image.

TRADITIONAL CLOUD WORKLOADS

Servicing VM requests based on the above metrics has proven adequate for traditional cloud workloads, since memory has in general been the primary point of contention for web-based apps and services, followed by the availability of CPU and storage resources.

In the case of traditional cloud workloads, per-node bandwidth utilization is an infrequent issue, and on the occasion that it is, 'noisy neighbors' are usually prevented from affecting other workloads with virtual switch based QoS, or by over-provisioning hypervisor resources.

NFV Requirements

With the entrance of NFV into the equation, requirements for scheduling need to be re-considered. Providers are starting to move network functions into the virtual world. These functions include, but are not limited to:

Firewalls
Intrusion detection and prevention systems
Route reflector
WAN acceleration appliances

In general, purpose built VNFs have fixed/known requirements for CPU, storage, and memory that can be scheduled by VM orchestrators such as Openstack Nova. It is important to note that in the case of NFV oversubscription does need to be considered. In the world of networking, sharing CPU cores or RAM will create sub-optimal network performance, so this does need to be taken into account when making scaling decisions.

Additionally, the network requirements for each subscriber can vary greatly. As an example, the requirements for a small business, branch office, or residential customer might require a virtual firewall capable of passing an average of 25 Mbps (IMIX), while a business providing free Wi-Fi to customers would potentially need far more bandwidth. At the time of this writing, Nova does not have the ability to schedule based on network utilization or for VNF workloads, this is a gap that needs to be addressed.

SCHEDULING OPTIONS

Since we have established different scheduling needs for traditional cloud workloads vs. VNF, we can consider some options to make the cloud orchestration software take available network bandwidth into consideration.

Turning again to Openstack as our example, we can look at the workflow of the filter scheduler, which makes the decision on workload placement. When a new instance is created a series of filters are applied in order to choose the appropriate hypervisor upon which to place that instance. Filters are binary, either a host passes filtering or it's rejected from consideration.
Current filters include [5]:

AggregateCoreFilter
AggregateDiskFilter
AggregateImagePropertiesIsolation
AggregateInstanceExtraSpecsFilter
AggregateIoOpsFilter
AggregateMultiTenancyIsolation
AggregateNumInstancesFilter
AggregateRamFilter
AggregateTypeAffinityFilter
AllHostsFilter
AvailabilityZoneFilter
ComputeCapabilitiesFilter
ComputeFilter
CoreFilter
NUMATopologyFilter
DifferentHostFilter
DiskFilter
GroupAffinityFilter
GroupAntiAffinityFilter
ImagePropertiesFilter
IsolatedHostsFilter
IoOpsFilter
JsonFilter
MetricsFilter
NumInstancesFilter
PciPassthroughFilter
RamFilter
RetryFilter
SameHostFilter
ServerGroupAffinityFilter
ServerGroupAntiAffinityFilter
SimpleCIDRAffinityFilter
TrustedFilter
TypeAffinityFilter

During the scheduling process, these filters are applied based on resources requested. Additional weights are applied, and finally workloads are created on the appropriate compute nodes.

The following diagram shows the filtering workflow. [5]:

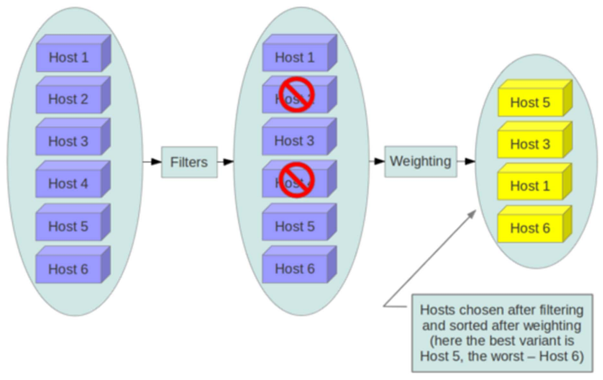

Figure 2: Filtering Workflow. Source: openstack.org

NetworkBWFilter

The contention of this paper is Network awareness is a requirement of NFV, and is absent from the metrics currently used to schedule virtual workloads.

One proposed solution is to create a a new filter that would allow the scheduler to take network requirements into account during into the decision making process. Taking the operative example from the functionality of the IoOpsFilter, a filter such as NetworkBwFilter would be created. As part of the NetworkBwFilter, a max_bw_per_host would be set to specify a high water mark allowable on a particular compute host.

When the request for a new VM instantiation is made, a required amount of bandwidth for that workload would be specified. As with existing filters, each time a host is selected for placement of VNFs, those resources are consumed virtually, and subsequent host selections can be adjusted accordingly. This would guarantee that the amount of requested bandwidth would never exceed that which is available.

TelemetryFilter

Another possibility is to create a filter that would take into account usage data from a telemetry package (e.g. ceilometer) and use actual usage data to make host filtering decisions. This would allow for even more efficient use of network bandwidth.

This filter could even make decisions based on the results of API calls to external systems. An example of this type of operation would be that during the workload creation process, the scheduler makes API calls to a network analytics server.
Utilization information gathered from the network analytics system could be applied as either a weight or a filter during the scheduling process.

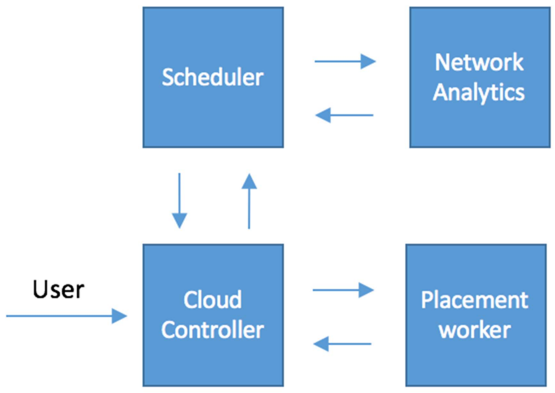

Figure 3: External Interaction with Network Analytics

Since reducing contention between workloads is a primary concern, continual monitoring would be required, with the potential need to re-spawn workloads elsewhere and redirect traffic to them when utilization changes.

Note: There exists a blueprint within Openstack for 'Utilization aware scheduling'. Which is meant to take into account transient resources, allowing decisions to be made based upon gathered usage statistics. [3] Work on the network monitor portion of Utilization aware scheduling appears to have been abandoned[4]. Were this work to continue, it would be a positive step in the goals laid out by this paper, at least where the Openstack project is concerned.

SUMMARY AND CONCLUSION

This paper has compared and contrasted traditional VM workload scheduling with the requirements of NFV applications and included a high level discussion of potential solutions. A gap exists in current cloud management software which needs to be addressed in order to make it more suitable for NFV applications. Taking network utilization into account during workload scheduling decisions can improve ROI for standard hardware by providing the ability to more densely pack virtual workloads and avoid stranded resources.

REFERENCES

[1] Chiosi, Margaret, et. al, (2012). *Network Functions Virtualisation, An Introduction, Benefits, Enablers, Challenges & Call for Action*, SDN and OpenFlow World Congress, Darmstadt-Germany
[2] http://docs.openstack.org/icehouse/training-guides/content/operator-getting-started.html
[3] https://wiki.openstack.org/wiki/UtilizationAwareScheduling
[4] https://review.openstack.org/#/c/44007/
[5] http://docs.openstack.org/liberty/config-reference/content/section_compute-scheduler.html