TECHNISCHE UNIVERSITÄT CHEMNITZ

Fakultät für Elektrotechnik und Informationstechnik
Professur Kommunikationsnetze

Masterarbeit

Design and implementation of an SDN based authentication and separation mechanism for WiFi users

Nishant Ravi

Chemnitz, February 24, 2017

Autor:	**Nishant Ravi**
Email:	**nisr@hrz.tu-chemnitz.de**
Matrikelnummer:	**355151**
Prüfer:	**Prof. Dr.-Ing. Thomas Bauschert**
Betreuer:	**Dipl.-Ing. Florian Schlegel**
Ausgabedatum:	July 11, 2016
Abgabedatum:	February 24, 2017

Abstract

The ever-increasing use of data services on mobile devices, places increased demands on existing networks. Especially in busy areas, such as shopping centers, office buildings or in event centers, the existing network coverage by UMTS and LTE is no longer sufficient. It is, therefore, obvious to direct some traffic through other radio standards. In this case, WLAN is particularly suitable because those frequencies are free to use without any license restrictions and since most mobile devices have long since supported this. However, an uncontrolled number of WLAN access points can interfere with each other. It is, therefore, desirable to install only one set of access points at these locations and manage them centrally. The research project BIC-IRAP (Business Indoor Coverage Integrated Radio Access Points) is a project aimed at providing a seamless coupling between LTE and WLAN.

The separation of data traffic is an important aspect when using shared hardware. No direct data exchange between the networks of different mobile radio providers should be possible. Likewise, the networks of different businesses or companies should be kept strictly separate from each other. Classic VLANs would be used for this purpose. Within the scope of the BIC-IRAP project, however, there were considerations to control parts of the network using SDN. Therefore, the goal of this master thesis is to operate an access point (AP) on an Open Flow-controlled switch. Users can be authenticated against a RADIUS server. The AP should supply at least two separate networks. If possible, the separation of data traffic should already take place in the AP. Optionally the AP should provide Hotspot 2.0 functionality.

The conceptualization and implementation must be documented in detail. The optional components are carried out in consultation with the supervisor. The successful completion of the work is a test set-up. The achievable performance characteristics must be recorded.

Contents

List of Figures

List of Tables

1 Introduction

The penetration of mobile internet users has increased many fold from a few thousands to millions over a short span of time. Due to the increasing demand for data among subscribers, mobile operators are pushed to go beyond boundaries to provide efficient and reliable data service to their customers. Although, the existing network services such as UMTS and LTE can handle larger data capacity, their coverage is not always sufficient in crowded places such as office buildings, convention centers, shopping malls etc. There is an urgent need to find a solution on how to offload the mobile data traffic over to other radio standards.

In such case, WLAN is an existing radio standard that has already been deployed in large numbers and has been supported by millions of devices lately. One unique advantage of using WLAN over other radio standards would be its license free usage of its radio frequency for commercial purposes. This WLAN standard, when deployed in a controlled manner can support data traffic routed from the mobile services. The IEEE 802.11 WLAN has already been widely used for commercial enterprises ranging from office networks, shopping malls to educational institutions etc. The deployment rages from a few dozens to hundreds of access points (APs), which serve many users through multitude of devices ranging from mobile devices, laptops to printers and other connected hardware. These networks also provide varied set of services that includes authentication, authorization and accounting (AAA), dynamic channel reconfiguration, interference management, security such as intrusion detection and prevention and providing quality of service.

These enterprise WLAN AP's are usually centrally managed through a controller. The task now is to find a solution to seamlessly direct traffic between LTE and WLAN. The research project BIC-IRAP (Business Indoor Coverage Integrated Radio Access Point) is currently aimed at providing a solution for the seamless coupling between LTE and WLAN.

The growing adoption of Software Defined Networking in the recent years has given rise to providing unique solutions without depending too much on hardware. The advantage of using SDN is that, it separates the network control pane from the physical network topology and uses software control flow to define how traffic is forwarded in the network. For example, the routing table and the flow control of a switch can be easily controlled remotely through a software controller. The capabilities of SDN is possible due to the use of OpenFlow protocol which is a standardized protocol that is used by the SDN based controller to manipulate the flow tables of network switches. This provides more flexibility to programmatically control the behavior of network switches by building network applications that talk to the network controller. Any OpenFlow enabled switch from any vendor provides a common interface to be manipulated via a controller, thus providing flexibility and simplified network management.

The cloud is a big deal for given reasons:

- It does not need any effort on the consumer or the end user's part to maintain or manage it.
- Any authenticated user can access the cloud-based applications and services from anywhere. All that the user needs is a device with an Internet connection.
- It is effectively infinite in size, so that the consumer or the end user doesn't need to worry about it running out of capacity.
- The resources can be scaled up or down quickly and easily to meet the desired changing demands of the consumer.
- The services offered in the cloud are metered services, so the end user pay only for what they use.

1.1 Where did the cloud come from?[1]

The Internet has its roots in the 1960's. But not until the early 1990's that it had any relevance for businesses. The World Wide Web (WWW) was born in 1991, and in 1993 a web browser called Mosaic was released that allowed users to view web pages that included graphics as well as text. This heralded the first company web sites – and not surprisingly, most of these belonged to companies involved in computing and technology.

As Internet connections got faster and more reliable, a new type of company called an "Application Service Provision" or ASP started to appear. ASPs took the existing business applications and ran them for their customers. The ASP would buy the computing hardware and keeping the application running, and the customer would pay a monthly fee to access it over the Internet.

But it wasn't until right at the end of the 1990's that cloud computing as it is known to-day did appear. That is when a company by name "salesforce.com" in 1999 introduced its own multi-tenant application which was specifically designed:

- to run "in the cloud";
- to be accessed over the Internet from a web browser;
- to be used by large numbers of customers simultaneously at low cost.

Since then the cloud has enormously grown and is still growing.

1.2 Services of "The Cloud"[2]

The Cloud is a very broad concept, and it covers just about every possible sort of online service. Anything as a service normally abbreviated as XaaS, provided remotely can be generalised as a cloud service. But when businesses refer to **"cloud procurement"**, there are usually three models of cloud services under consideration:

- **Software as a Service (SaaS)**
- **Platform as a Service (PaaS)**
- **and Infrastructure as a Service (IaaS)**

The cloud infrastructure as a service model can also be segregated based on the type of cloud network desired by the business which can be:

- Private cloud: with a cloud network accessible by limited and restricted permission to only particular organisation.
- Public cloud: with a cloud network accessible to the general public.
- and Hybrid cloud: which is partly private and secured, and partly public and accessible.

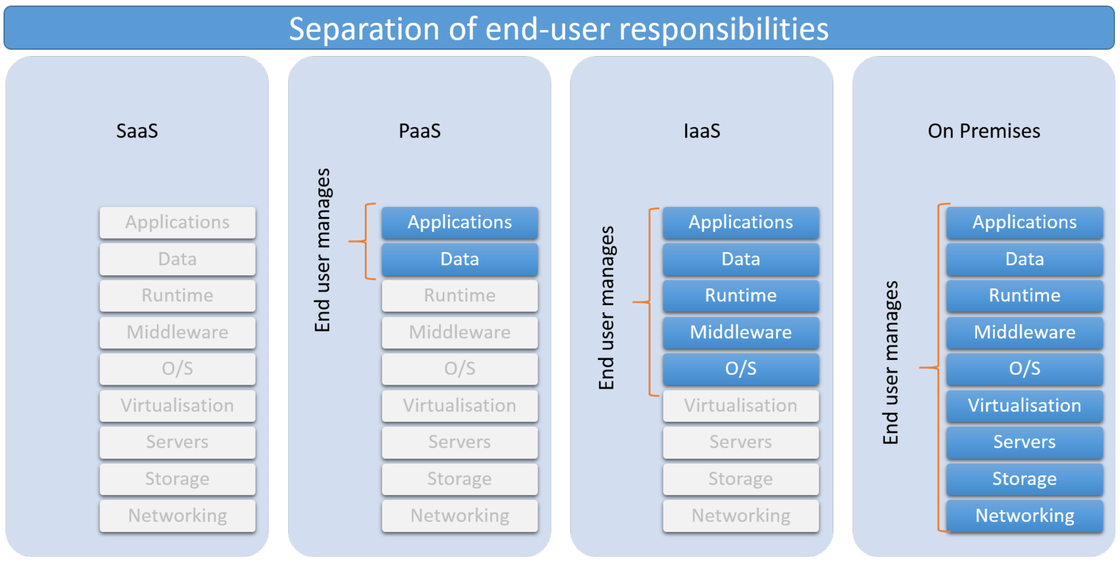


Figure 1.1: Separation of end user responsibilities based on Internet cloud service model

The figure 1.1 about the cloud service model describes what would be the roles that needs to be played by an end user to utilise and manage each type of service model compared to working on on-premise systems.

1.2.1 Software as a Service (SaaS)

Software as a service (SaaS) is a software distribution model in which a third-party provider hosts the applications and makes them available to customers over the Internet. Cloud-based applications or software as a service, run on distant computers "in the cloud" that are owned and operated by others and that connect to the user's computer via the Internet and, usually, a web browser.

With SaaS, you no longer have to purchase, install, update and maintain the software.

Figure 1.2: Software as a service model[4]

SaaS is the most familiar form of cloud service for consumers. SaaS moves the task of managing software and its deployment to third-party services. Among the most familiar SaaS applications for business are customer relationship management applications like Salesforce, productivity software suites like Google Apps, and storage solutions like Box and Dropbox.

Use of SaaS applications tends to reduce the cost of software ownership by removing the need for technical staff to manage the installation, maintenance, and upgrade software, as well as reduce the cost of licensing software. SaaS applications are usually provided on a subscription model.

To summarise the benefits of SaaS:

- The end user can sign up and rapidly start using innovative business apps

- Apps and data are accessible from any device connected to Internet
- No data is lost if the user's computer or a device breaks, as the data is in the cloud
- The service is able to dynamically scale to the usage needs

1.2.2 Platform as a Service (PaaS)

Platform as a service (PaaS) is a cloud computing model that delivers development and management tools or applications over the Internet. In a PaaS model, a cloud provider delivers hardware and software tools, "usually those needed for application development", to its users as a service. The provider of the PaaS hosts the hardware and software on their own infrastructure. As a result, PaaS frees the users from having to install in-house hardware and software to develop and run a new application.

It provides a cloud-based environment with everything required to support the complete lifecycle of building and delivering web-based (cloud) applications, without the cost and complexity of buying and managing the underlying hardware, software, provisioning, and hosting. The PaaS provider on request can provide a web development tool or platform which can be customised for the end user.

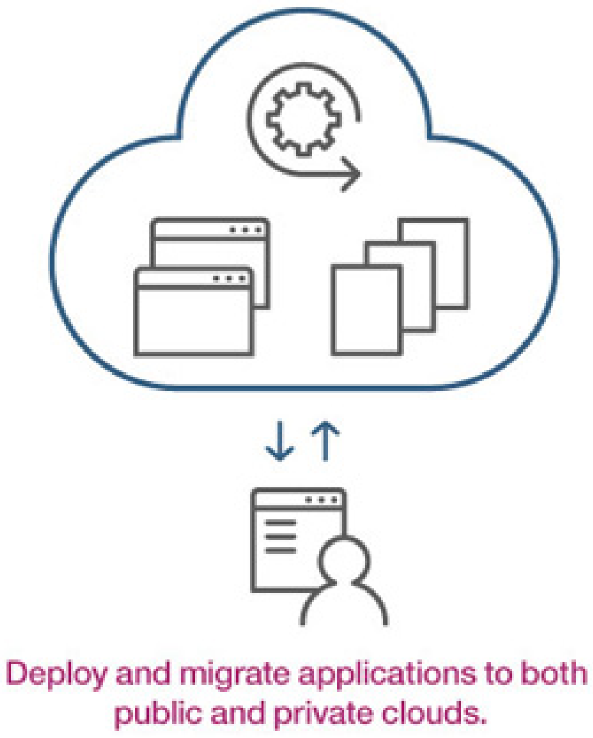

Deploy and migrate applications to both
public and private clouds.

Figure 1.3: Platform as a service model[5]

For example, deploying a typical business tool locally might require an IT team to buy and install hardware, operating systems, middleware (such as databases, Web servers and so on) the actual application, define user access or security, and then add the application to existing systems management or application performance monitoring (APM) tools. IT teams must then maintain all of these resources over time. A PaaS provider, however, supports all the underlying computing and software; users only need to log in and start using the platform – usually through a Web browser interface.

Common PaaS vendors include Salesforce.com's Force.com, which provides an enterprise customer relationship management (CRM) platform. PaaS platforms for software development and management include Appear IQ, Mendix, Amazon Web Services (AWS) Elastic Beanstalk, Google App Engine and Heroku.

To summarise the benefits of PaaS:

- Develop applications and get to market faster
- Deploy new web applications to the cloud in minutes
- Reduce complexity with middleware as a service

1.2.3 Infrastructure as a Service (IaaS)

Infrastructure as a Service (IaaS) is a form of cloud computing that provides virtualized computing resources over the Internet. Infrastructure as a service provides consumers with computing resources including servers, networking, storage, and data center space on a pay-per-use basis.

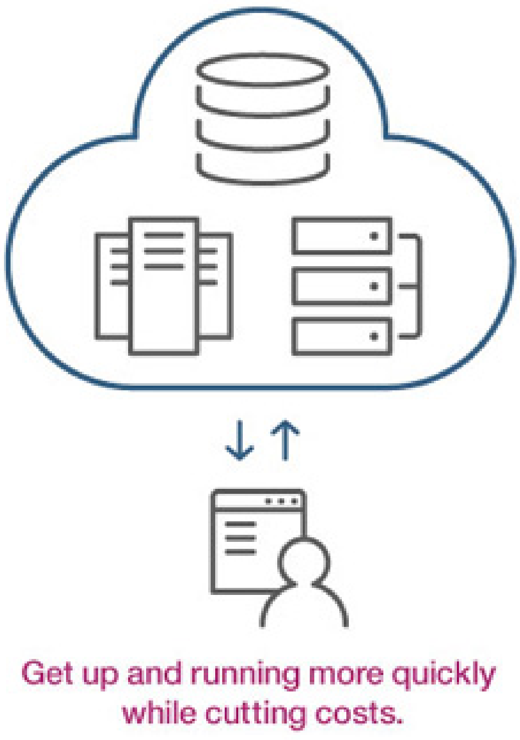


Figure 1.4: Infrastructure as a service model[6]

In an IaaS model, a third-party provider hosts hardware, software, servers, storage and other infrastructure components on behalf of its users. IaaS providers also host users' applications and handle tasks including system maintenance, backup and resiliency planning.

IaaS platforms offer highly scalable resources that can be adjusted on-demand. This makes IaaS well-suited for workloads that are temporary, experimental or change unexpectedly.

Other characteristics of IaaS environments include the automation of administrative tasks, dynamic scaling, desktop virtualization and policy-based services.

For example, if a business is developing a new software product, it might be more cost-effective to host and test the application through an IaaS provider. Once the new software is tested and refined, it can be removed from the IaaS environment for a more traditional in-house deployment or to save money or free the resources for other projects.

Leading IaaS providers include Amazon Web Services (AWS), Windows Azure, Google Compute Engine, Rackspace Open Cloud, IBM SmartCloud Enterprise and Profit-Bricks. To summarise the benefits of IaaS:

- No need to invest in having or owning the hardware
- Infrastructure scales on demand to support dynamic workloads
- Flexible, innovative services available on demand

1.3 Summary about "The Cloud"

The cloud has become a key enabler for today's small or medium sized business to unlock creativity, drive unrivaled innovation and level the competition with larger enterprise competitors.

Prior to the advent of cloud-based products software solutions delivered over the Internet companies were often forced to invest in servers and other products to run software and store data. The advent of cloud services as well as their steady improvement in such areas as security and reliability make these solutions a logical choice for business owners and principals who want the latest innovations, functionality, and efficiency as well as cost effectiveness.

Many businesses garner considerable cost savings by migrating their software systems to the cloud. In addition to reducing reliance on the purchase and maintenance of servers, companies often lower their information technology costs in such areas as dedicated personnel and software upgrades. Most cloud services upgrade and update software via the Internet with little or no downtime for end users, decreasing the wait time associated with installing and testing software on an on-site network. Moreover, cloud-based products are scalable: unlike conventional software, cloud services can be expanded as needed to encompass as many end users as required without additional overhead of servers and upgrades to handle added workloads.

When more number of small and medium sized business begin to realize that the cloud can do more than just reduce the cost of IT, the journey to the cloud becomes inevitable and the question shifts from whether to adopt cloud technologies to how to do so sensibly.

It is fascinating to understand the working of cloud as it has created lot of new opportunities for research, jobs and investment in this domain.

2 Introduction to OpenStack Cloud[3]

An introduction about cloud computing and its different offering of service modelling was introduced in the chapter ??. This chapter Introduction to OpenStack Cloud[3] introduces about one of the open source IaaS cloud operating system called OpenStack.

OpenStack is a cloud operating system that controls large pools of compute, storage, and networking resources throughout a datacenter, all managed through a dashboard that gives administrators control while empowering their users to provision resources through a web interface.

OpenStack is the open source cloud computing platform that meets the needs of public and private cloud providers regardless of size. OpenStack services control large pools of compute, storage, and networking resources throughout a data center. The figure 2.1 shows the basic overview of core services in OpenStack.

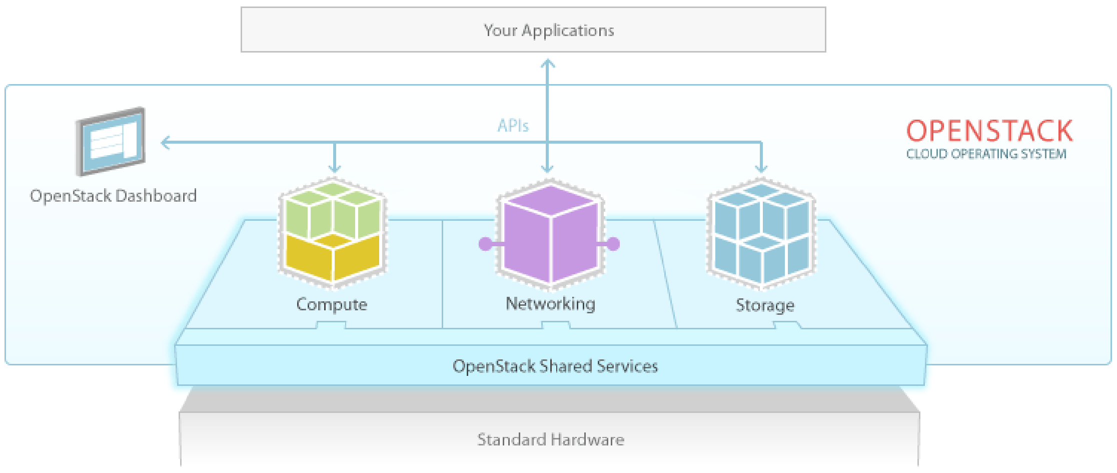


Figure 2.1: OpenStack software diagram[7]

The technology behind OpenStack consists of a series of interrelated projects delivering various components for a cloud infrastructure solution. Each service provides an open API so that all of these resources can be managed through a dashboard that gives administrators control while empowering users to provision resources through a web interface, a command-line client, or software development kits that support the API.

OpenStack is designed for horizontal scalability, so the user can easily add new compute, network, and storage resources to grow their cloud over time. In addition to the pervasiveness of massive OpenStack public clouds, many organizations, such as PayPal, Intel, and Comcast, build large-scale private clouds. OpenStack offers much more than a typical software package because it lets the user to integrate a number of different technologies to construct a cloud. This approach provides great flexibility, but the number of options might be daunting at first.

OpenStack clouds are powered by various OpenStack projects. The selection of the components in openstack depends on how does the IaaS provider wants to use the OpenStack:

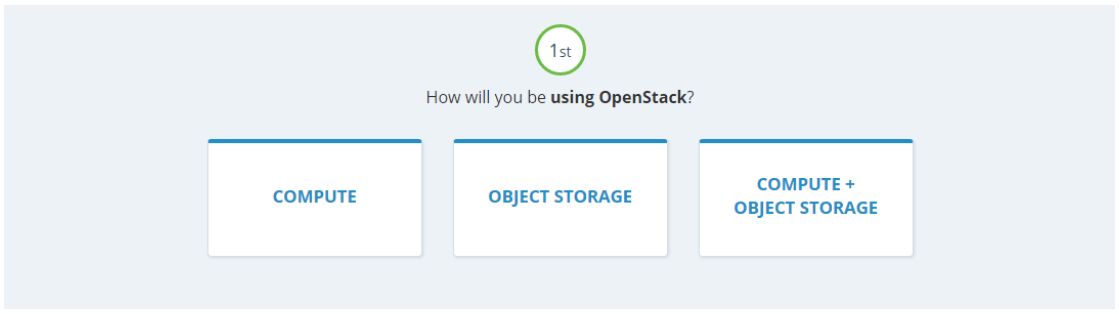


Figure 2.2: Primary objective to decide the purpose of having an OpenStack Cloud[3]

The figure 2.2 actually specifies if the primary objective of the usage of OpenStack is, only with Compute - which defines the management of virtual machines on cloud, or only with Object Storage - which defines the allocation of end user based storage space on cloud, or the mix of both of Compute and Object Storage. Based on the objective of setting up the cloud, the core components can be selected as per the need from the wide range of available components.

2.1 Architecture of OpenStack

The OpenStack project is an open source cloud computing platform that aims for simple implementation, massive scalability, and a rich set of features. Cloud computing experts from around the world contribute to the project.

OpenStack provides an Infrastructure-as-a-Service (IaaS) solution through a variety of complemental services. Each service offers an application programming interface (API) that facilitates this integration.

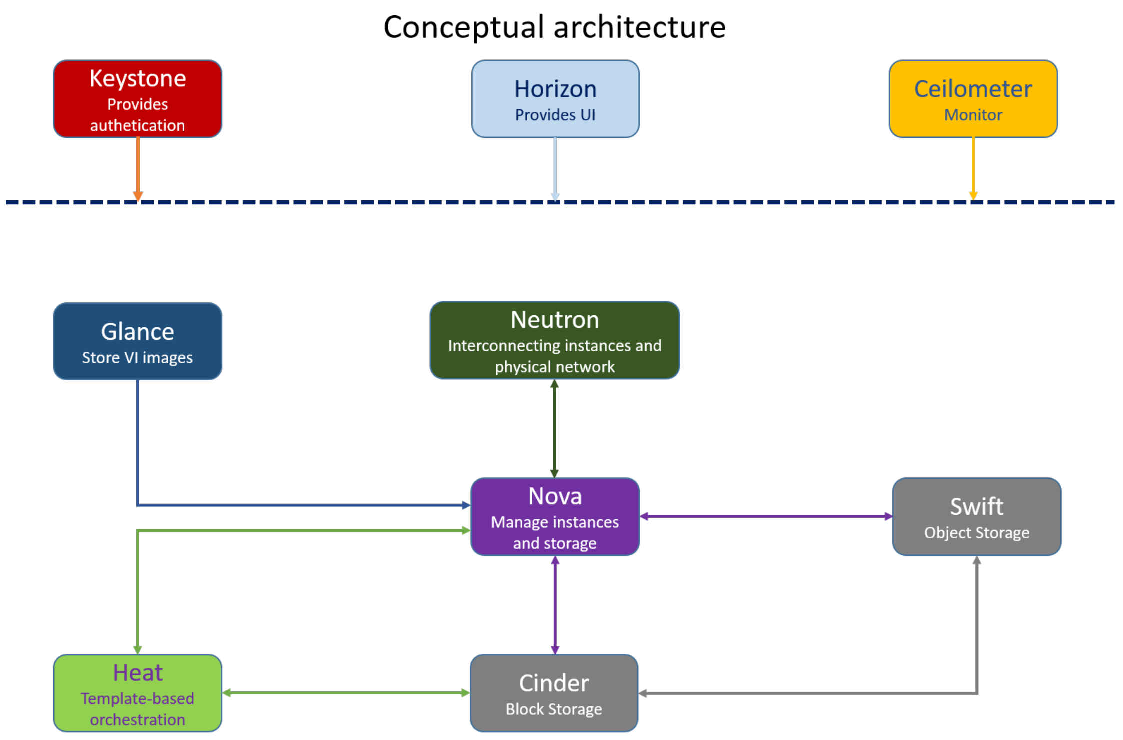


Figure 2.3: The relationships among the OpenStack services

The figure 2.3 shows the architecture of the services and how each service communicates with the other services and provide the desired cloud solution.

The implementation of the services are based on the objective of having the OpenStack cloud. Each purpose of cloud solution have their own set of service requirements that needs to be installed on the nodes and enabled.

2.1.1 OpenStack for compute

To set up the OpenStack for the purpose of Compute, it would require the implementation of the following services:

- Identity service (Keystone) – core service

- Compute service (Nova) – core service
- Networking service (Neutron) – core service
- Block storage service (Cinder) – core service
- Image service (Glance) – core service
- Dashboard service (Horizon) – optional
- Telemetry service (Ceilometer) – optional

2.1.2 OpenStack for storage

To set up the OpenStack for the purpose of storage, it would require the implementation of the following services:

- Object storage service (Swift) – core service
- Identity service (Keystone) – core service
- Dashboard service (Horizon) – optional
- Telemetry service (Ceilometer) – optional

2.1.3 Description of the core services in OpenStack

- Compute service (Nova) - Manages the lifecycle of compute instances in an OpenStack environment. Responsibilities include spawning, scheduling and decomissioning of machines on demand.
- Object storage service (Swift) - Stores and retrieves arbitrary unstructured data objects via a RESTful, HTTP based API. It is highly fault tolerant with its data replication and scale out architecture. Its implementation is not like a file server with mountable directories.
- Networking service (Neutron) - Enables network connectivity as a service for other OpenStack services, such as OpenStack Compute. Provides an API for users to define networks and the attachments into them. Has a pluggable architecture that supports many popular networking vendors and technologies.
- Image service (Glance) - Stores and retrieves virtual machine disk images. OpenStack Compute makes use of this during instance provisioning.

- Identity service (Keystone) - Provides an authentication and authorization service for other OpenStack services. Provides a catalog of endpoints for all OpenStack services.
- Dashboard service (Horizon) - Provides a web-based self-service portal to interact with underlying OpenStack services, such as launching an instance, assigning IP addresses and configuring access controls.
- Telemetry service (Ceilometer) - Monitors and meters the OpenStack cloud for billing, benchmarking, scalability, and statistical purposes.
- Block storage service (Cinder) - Provides persistent block storage to running instances. Its pluggable driver architecture facilitates the creation and management of block storage devices.
- Orchestration service (Heat) - Orchestrates multiple composite cloud applications by using either the native HOT template format or the AWS CloudFormation template format, through both an OpenStack-native REST API and a CloudFormation-compatible Query API.

The thesis is more focussed on the understanding of the nova compute service and the nova filter scheduler, and this would be documented further in the next sections the OpenStack Nova Compute and the Standard Nova Filter Scheduler.

2.2 OpenStack Nova Compute

Nova is an OpenStack project designed to provide power massively scalable, on demand, self service access to compute resources. Nova enables the lifecycle management of the virtual machines by provisioning or managing the resources for creation, management or deletion of virtual machines.

Nova is further segregated into many sub modules based on the defined purpose like segregation of nodes, scheduling mechanism, management of virtual machines.

2.2.1 Host Aggregates

Nova enables the segregation of nova nodes by categorizing them into "Availability Zones" by the use of Host Aggregates mechanism. This mechanism helps to further divide resources based on Availability Zones. Host aggregates are visible only to the administrators whereas Availability zones are visible to the users. This information is used by the nova scheduler as to enable advanced scheduling or to define any logical groups for creation or migration of virtual machines.

2.2.2 Threading model

All OpenStack services use the green thread model of threading which reduces the likelihood of race conditions. In case of any code taking longer execution time and blocking any other threads for execution, an added eventlet code from the greenthread library will hlp to switch to process any pending threads.

2.2.3 Virtual Machine States and Transitions

Nova manages the state transitions of virtual machine throughtout the lifecycle from creation to deletion or handle any error states. When the creation of an instance is requested, the nova allocates the compute node for hosting the instance and enters the initial state of "building" and spawns the virtual instance unless there are any errors which would lead to the transition of the state to "error". As shown in *Figure* 2.4, it provides the information about different transitional states along with the possible change of state instance state.

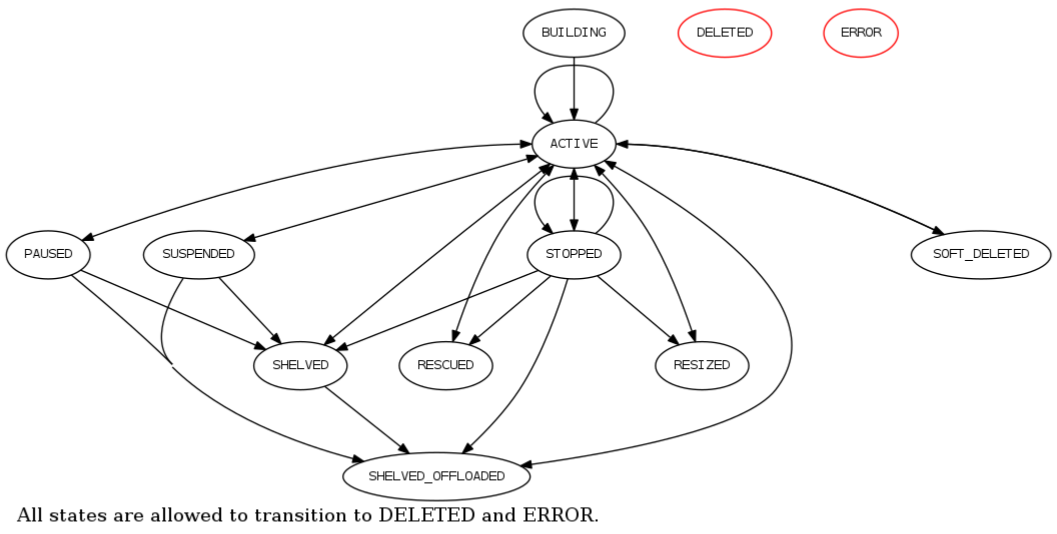


Figure 2.4: The diagram shows the allowed different state transistions of a virtual machine[8]

2.2.4 Filter Scheduler

Nova has its own scheduling mechanism called "Filter Scheduler" which computes the placement decision of creation or migration of virtual instances. The scheduler is further explained in detail in the section Standard Nova Filter Scheduler.

2.2.5 Advanced Message Queue Protocol and Nova

The Advanced Message Queue Protocol (AMQP) is the messaging technology chosen by OpenStack to communicate between different nodes using a message queue enabler like RabbitMQ or Qpid. The nova components use the Remote Procedure Calls(RPC hereinafter) to communicate to one another. The RPC calls are authenticated for communication using message queue enabler like rabbitMQ, which validates the authentication using Keystone.

2.2.6 Block Device Mapping

Nova has a concept of block devices that can be exposed to cloud instances. Block device mapping is a way to organize and keep data about all of the block devices an instance has.

2.2.7 Nova OpenStack RESTful API

The Nova provides the RESTful API which is exposed as an HTTP request service for routing, controllers and actions, serialization and trigering faults.

2.2.8 Conductor

Conductor serves as a database proxy, object backporter and also as a centralized place to manage the execution of workflows which involve the scheduler.

2.2.9 Notifications in Nova

Similarly to other OpenStack services Nova emits notifications to the message bus with the Notifier class provided by oslo.messaging.

2.3 Standard Nova Filter Scheduler

The **Filter Scheduler** supports filtering and weighting to make informed decisions on where a new instance should be created. This Scheduler supports working with Compute Nodes only.

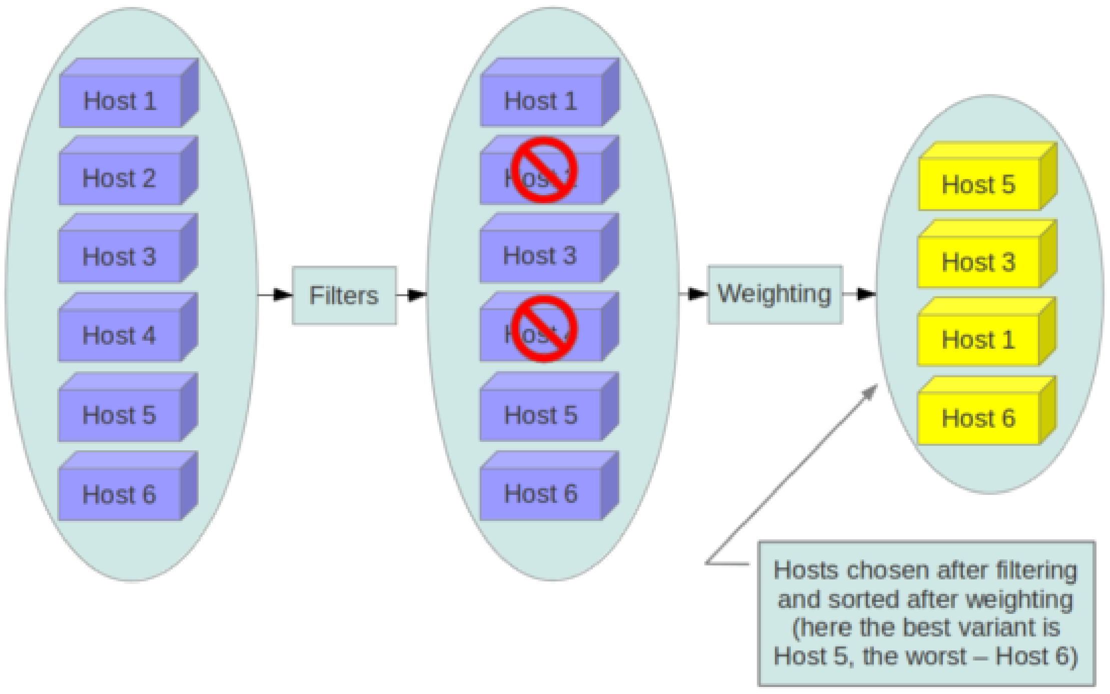

Host 1
Host 2
Host 3
Host 4
Host 5
Host 6
Filters
Host 1
Host 3
Host 5
Host 6
Weighting
Host 5
Host 3
Host 1
Host 6
Hosts chosen after filtering and sorted after weighting (here the best variant is Host 5, the worst – Host 6)

Figure 2.5: Filtering the compute hosts and sorting them based on weights[9]

The filter scheduler iterates over all the available compute nodes, evaluates each compute node against the set of filters, and a list of the possible nodes for instance creation is generated sorted based on the weights. The scheduler then chooses the best host for the instance by choosing the most weighted host. For any specific filter to pass a specific host, the filter matches the users request against the state of the host as defined by each filter definition.

If the scheduler does not find any hosts and returns an empty set of list of hosts for any requested instance, it means that there is no possible host where the instance can be placed to complete the user request.

The filter scheduler has to be flexible to support the required variety of filtering and weighting strategies. The nova also permits to implement the user's own filtering algorithm.

2.3.1 Filtering

There are many standard filter classes which may be used (nova.scheduler.filters):

- **AllHostsFilter** - The initial hosts filter to pass all the available hosts.
- **ImagePropertiesFilter** - This filters the possible hosts based on properties defined by the instance's image. It passes hosts that can support the properties specified on the image used by the instance.
- **AvailabilityZoneFilter** - This filters the hosts which are grouped by availability zone. It passes only those hosts which match the availability zone specified in the instance properties.
- **ComputeCapabilitiesFilter** - This filter checks that if the capabilities provided by the host compute service satisfy any extra specifications associated with the instance type. It passes hosts that can create the specified instance type.
- **ComputeFilter** - This passes all the hosts that are operational and enabled.
- **CoreFilter** - This filters the hosts based on available VCPU cores. It passes hosts with sufficient number of CPU cores.
- **AggregateCoreFilter** - This filters the hosts by CPU core number with per-aggregate `cpu_allocation_ratio` setting specified in the nova configuration. If no per-aggregate value is found, it will fall back to the global default `cpu_allocation_ratio`.
- **IsolatedHostsFilter** - This filter is based on `image_isolated`, `host_isolated` and `restrict_isolated_hosts_to_isolated_images` flags.
- **JsonFilter** - This filter allows simple JSON-based grammar for selecting hosts.
- **RamFilter** - This filters the hosts by their available RAM capacity. Only those hosts with sufficient RAM capacity to host the instance are passed.
- **AggregateRamFilter** - This filters the hosts by RAM with per-aggregate `ram_allocation_ratio` setting specified in the nova configuration. If no per-aggregate value is found, it will fall back to the global default `ram_allocation_ratio`.

- **DiskFilter** - This filters the hosts by their storage space. Only hosts with sufficient storage space to host the instance are passed.
- **AggregateDiskFilter** - This filters the hosts by disk allocation with per-aggregate `disk_allocation_ratio` setting. If no per-aggregate value is found, it will fall back to the global default `disk_allocation_ratio`.
- **NumInstancesFilter** - This filters the compute nodes by number of running instances. Nodes with too many instances will be filtered out.
- **AggregateNumInstancesFilter** - This filters the hosts by number of instances with per-aggregate `max_instances_per_host` setting.
- **IoOpsFilter** - This filters the hosts by concurrent I/O operations on it. The hosts with too many concurrent I/O operations will be filtered out.
- **AggregateIoOpsFilter** - filters hosts by I/O operations with per-aggregate `max_io_ops_per_host` setting. If no per-aggregate value is found, it will fall back to the global default `max_io_ops_per_host`.
- **PciPassthroughFilter** - This filter schedules instances on a host if the host has devices to meet the device requests in the `'extra_specs'` for the flavor.
- **SimpleCIDRAffinityFilter** - This filter allows a new instance on a host within the same IP block.
- **DifferentHostFilter** - This filter allows the instance on a different host from a set of instances.
- **SameHostFilter** - This filter puts the instance on the same host as another instance in a set of instances.
- **RetryFilter** - This filters the hosts that have been attempted for scheduling. Only passes hosts that have not been previously attempted.
- **TrustedFilter** (EXPERIMENTAL) - This filters the hosts based on their trust. Only passes hosts that meet the trust requirements specified in the instance properties.
- **TypeAffinityFilter** - Only the hosts that are not already running an instance of the requested type are passed.
- **AggregateTypeAffinityFilter** - This limits `instance_type` by aggregate.
- **ServerGroupAntiAffinityFilter** - This filter implements anti-affinity for a server group.

- **ServerGroupAffinityFilter** - This filter works the same way as ServerGroupAntiAffinityFilter. The difference is that when you create the server group, you should specify a policy of 'affinity'.
- **AggregateMultiTenancyIsolation** - This isolate the tenants in specific aggregates.
- **AggregateImagePropertiesIsolation** - This isolates the hosts based on image properties and aggregate metadata.
- **MetricsFilter** - This filters the hosts based on metrics `weight_setting`. Only those hosts with the available metrics are passed.
- **NUMATopologyFilter** - This filters the hosts based on the NUMA topology requested by the instance, if any.

2.3.2 Weights

Filter Scheduler uses the weight based approach during its selection of a host for the requested instance. A weigher is a way to select the best suitable host from a group of valid hosts by giving weights to all the hosts in the list.

In order to prioritize one weigher against another, all the weighers have to define a multiplier that will be applied before computing the weight for a node. All the weights are normalized beforehand so that the multiplier can be applied easily. Therefore the final weight for the object will be:

```
weight = w1_multiplier * norm(w1) + w2_multiplier * norm(w2) + ...
```

The Filter Scheduler weighs hosts based on the config option `scheduler_weight_-classes`, this defaults to `nova.scheduler.weights.all_weighers`, which selects the following weighers:

- **RAMWeigher** - Compute weight based on available RAM on the compute node.
- **DiskWeigher** - Hosts are weighted and sorted by free disk space with the largest weight winning.
- **MetricsWeigher** - This weigher can compute the weight based on the compute node host's various metrics. The to-be weighed metrics and their weighing ratio are specified in the configuration file as the followings:

  ```
  metrics_weight_setting = name1=1.0, name2=-1.0
  ```

- **IoOpsWeigher** - The weigher can compute the weight based on the compute node host's workload. The default is to preferably choose light workload compute hosts.
- **ServerGroupSoftAffinityWeigher** - The weigher can compute the weight based on the number of instances that run on the same server group. The largest weight defines the preferred host for the new instance.
- **ServerGroupSoftAntiAffinityWeigher** - The weigher can compute the weight based on the number of instances that run on the same server group as a negative value.
- **IoOpsWeigher** - Hosts are weighted and sorted by free disk space with the largest weight winning.

Filter Scheduler makes a local list of acceptable hosts by repeated filtering and weighing. Each time it chooses a host, it virtually consumes resources on it, so subsequent selections can adjust accordingly. It is useful if the customer asks for a large block of instances, because weight is computed for each instance requested.

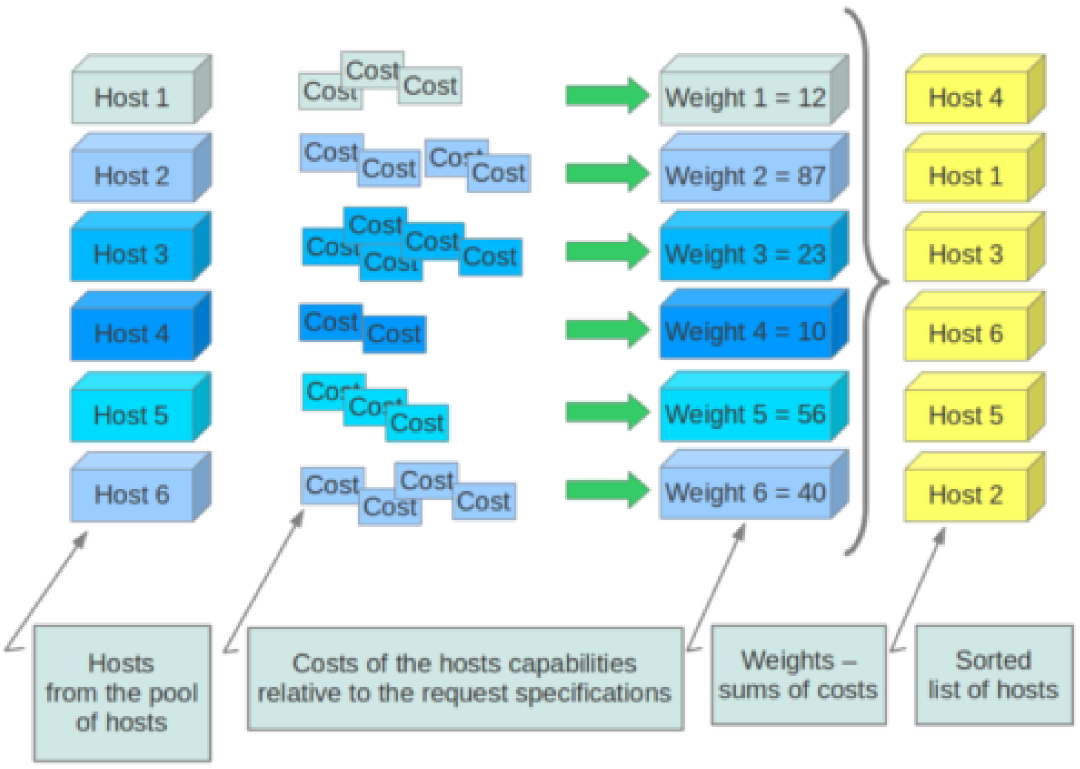


Figure 2.6: Compute hosts sorted based on weights[10]

At the end, the Filter Scheduler sorts selected hosts by their weight and attempts to provision instances on the chosen hosts.

2.4 Opening for any improvements in the filter scheduler?

The filter scheduler executes the instances one instance at a time. If an user requests for bulk creation of instances, for example: consider a request for creation of n instances, the filter scheduler would need to run for n number of times to execute the placement decision for each requested instance.

If the user's request for the creation of large number of instances could be modelled into a Linear programming model with the parameters that are required to filter the hosts and then computing them into a Linear programming solver like cPlex, the resultant solution could achieve better results with lesser execution time for the creation of large number of instances.

There could be many possibilities of modelling the linear programming model to have a placement decision of the host like the Virtual Network Embedding problem. Such problems are also known as knapsack problem in combinatorial optimisation.

Given the current availability of input data, there is also no possibility of network aware scheduling which would consider the amount of network load on the host machines before placing the instance.

With the learning curve of OpenStack, there could be many possibilities to optimise and solve the instance scheduling problem based on the purpose of the cloud provider and the availability of the inputs to model the Linear program with a better scheduling approach.

2.5 Structure of the thesis

The first chapter ?? provides the basic concept of cloud and its offerings. This chapter Introduction to OpenStack Cloud[3] gives an idea about OpenStack as an IaaS model and the brief understanding about the services offered by OpenStack.

The next chapter Installation and Configuration of OpenStack will explain the implementation of the services to set up the OpenStack for the purpose of computing. The chapter Standard Nova Scheduling Algorithm would explain about the Nova Filter Scheduler and the flow of code as how the filter scheduler algorithm is invoked during creation of instances. Then the chapter Implementation of Cplex based Nova Scheduler would explain the new mathematical formulation used to schedule the instances and its implementation. In the chapter Comparision of performance evaluation one could see and evaluate the performance between the existing filter scheduler and the new modelled scheduler algorithm followed by the chapters Further possible extensions in OpenStack and Conclusion.

3 Installation and Configuration of OpenStack

The OpenStack consists of several key service projects which needs to be installed seperately. These projects work together depending on the user's cloud needs. The projects include Compute, Identity Service, Networking, Image Service, Block Storage, Object Storage, Telemetry, Orchestration, and Database. Any services can be installed seperately and configure them stand-alone or as connected entities.

The Liberty version of OpenStack has been installed for the cloud test setup which has been documented in this thesis. The OpenStack installation guide can also be found at OpenStack Installation Guide for Ubuntu.

The OpenStack project is an open source cloud computing platform that supports all types of cloud environments. The project aims for simple implementation, massive scalability, and a rich set of features. Cloud computing experts from around the world contribute to the project.

OpenStack provides an Infrastructure-as-a-Service (IaaS) solution through a variety of complemental services. Each service offers an application programming interface (API) that facilitates this integration.

3.1 Minimum hardware requirements for the compute purpose of the cloud

It requires at the least two nodes(hosts) to set up the OpenStack and launch a virtual machine or instance. Optional services such as Block Storage and Object Storage require additional nodes.

The hardware architecture explained is the minimal hardware requirements for each type of node. The figure 3.1 gives a general idea about the required nodes and optional nodes to start with an OpenStack for compute.

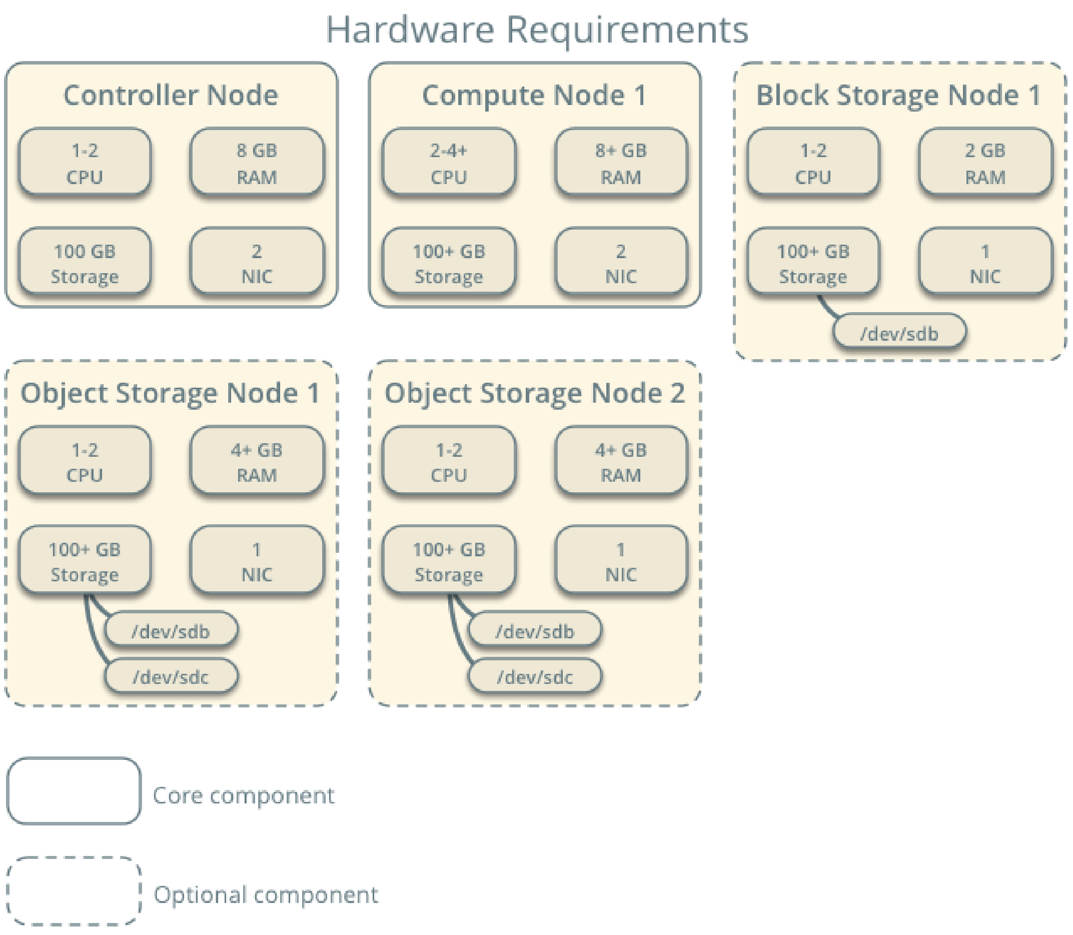


Figure 3.1: Minimum hardware requirements to set up the OpenStack test bed[11]

3.1.1 Controller

The `controller` node runs the Identity service, Image service, management portions of Compute, management portion of Networking, various Networking agents, and the dashboard. It also includes supporting services such as an SQL database, message queue, and Network Time Protocol.

Optionally, the `controller` node runs portions of Block Storage, Object Storage, Orchestration, and Telemetry services.

The `controller` node requires a minimum of two network interfaces, 8 GB of RAM, 100 GB of storage space.

3.1.2 Compute

The `compute` node runs the hypervisor portion of Compute that operates instances. By default, Compute uses the KVM hypervisor. The `compute` node also runs a Networking service agent that connects instances to virtual networks and provides firewalling services to instances via security groups.

You can deploy more than one `compute` node. Each node requires a minimum of two network interfaces, more than 8 GB of RAM, more than 100 GB of storage space and more than 2 CPU cores.

3.1.3 Block Storage

This is an optional implementation and has not been set up for the test bed for this thesis. The optional Block Storage node contains the disks that the Block Storage service provisions for instances.

You can deploy more than one block storage node. Each node requires a minimum of one network interface.

3.1.4 Object Storage

This is also an optional implementation and has not been set up for the test bed for this thesis. The optional Object Storage node contain the disks that the Object Storage service uses for storing accounts, containers, and objects.

This service requires two nodes. Each node requires a minimum of one network interface. You can deploy more than two object storage nodes.

3.2 Networking Options

OpenStack provides two diferent networking options:

- Networking Option 1: Provider networks
  The provider networks option deploys the OpenStack Networking service in the simplest way possible with primarily layer-2 (bridging/switching) services and VLAN segmentation of networks. Essentially, it bridges virtual networks to physical networks and relies on physical network infrastructure for layer-3 (routing)

services. Additionally, a DHCP service provides IP address information to instances.

- Networking Option 2: Self-service networks
  The self-service networks option augments the provider networks option with layer-3 (routing) services that enable self-service networks using overlay segmentation methods such as VXLAN. Essentially, it routes virtual networks to physical networks using NAT. Additionally, this option provides the foundation for advanced services such as LBaaS and FWaaS.

Considering the network options, the networking option 2 of self service networks have been implemented for the test bed.

3.3 Configuration of Services and config files

For the test bed of this thesis, there is one `controller` node and four `compute` nodes which have been configured. All the nodes have been installed with the `Ubuntu 14.04 LTS` Operating system. For the purpose of having the GUI, the desktop version of the Operating system has been set up on each node.

This thesis does not explain the installation in detail as it is already available in OpenStack Installation Guide for Ubuntu. This thesis covers the basics of the topics which were covered during the setup of the test bed are documented. And also, the problems that were faced due to some of the missing configurations as they were not mentioned in the OpenStack Installation Guide for Ubuntu are documented in this section.

To keep the configuration of the testbed simple, the password used for convenience on all the host machines and all the OpenStack services is `user`.

3.3.1 Host networking

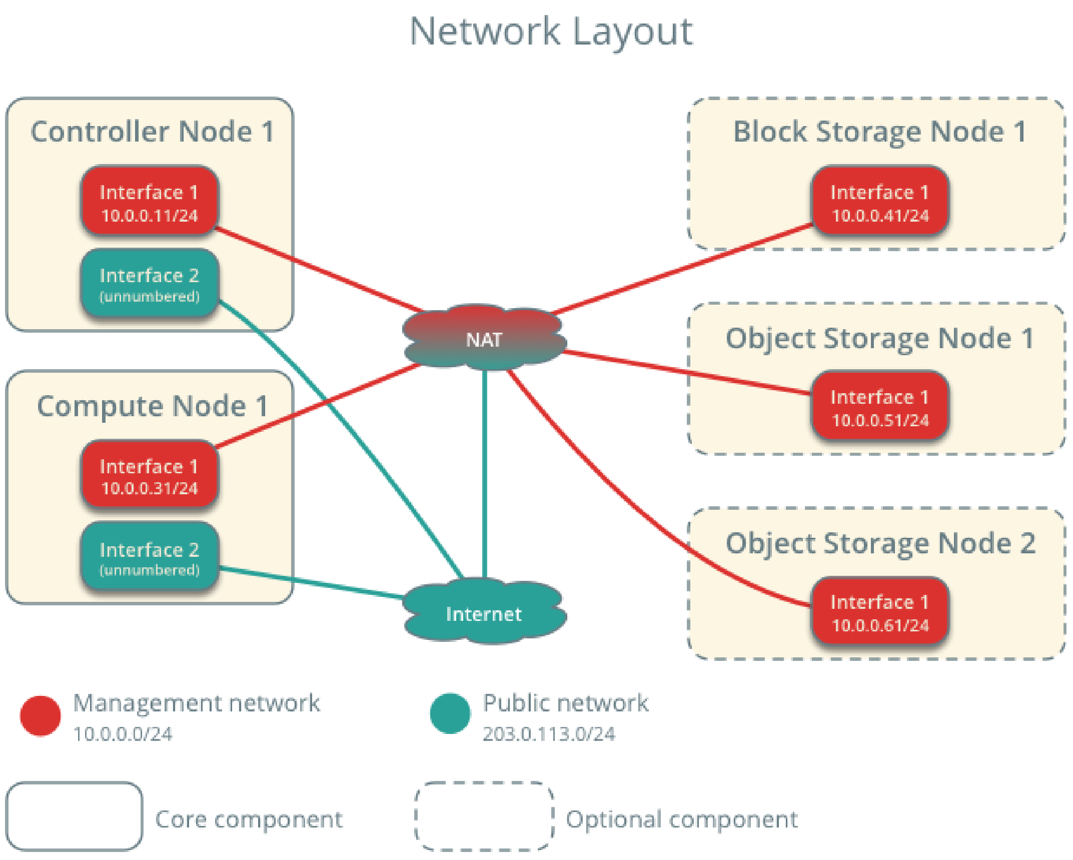


Figure 3.2: Network layout among all the host machines[12]

The figure 3.2 shows how the connections are established between all the node machines. A private management network and a public network provided with internet connection are set up for the `controller` and `compute` nodes. The management network's IP for the network interface of `controller` node is set to static IP of `10.0.0.11`. Similarly, the management network's IP for one of the network interfaces on the five of the `compute` nodes have the static IPs ranging from `10.0.0.31` to `10.0.0.35`.

In this thesis, the storage nodes have not been implemented.

For the testbed setup, one `controller` node and five computes nodes have been set up.

3.3.2 Network Time Protocol (NTP)

The Chrony application is installed on all the nodes to synchronise the time on all the installed nodes. The `controller` node is set to synchronise with the network time server provided by "TU-Chemnitz" given by the server host name as `ntphost1.hrz.tu-chemnitz.de`. The other `compute` nodes are set to have the synchronised time with the `controller` node by setting the `controller` node as network time server.

3.3.3 OpenStack packages

OpenStack packages are installed on all the nodes by enabling the OpenStack repository and installing all required basic packages.

The MariaDB SQL database server is installed on the `controller` node to store the information related to OpenStack services for authentication and other related activities. The MongoDB NoSQL database is installed on the `controller` for the purpose of Telemetry service (Ceilometer).

OpenStack uses the message queue to co-ordinate operations and services among the installed services. The RabbitMQ server is installed on the `controller` node to provide the message queue service among the services.

3.3.4 Add the Identity service

The OpenStack Identity service provides a single point of integration for managing authentication, authorization, and service catalog services. Other OpenStack services use the Identity service as a common unified API.

When an OpenStack service receives a request from a user, it checks with the Identity service whether the user is authorized to make the request.

In the setup, the Identity service has been installed on the `controller`.

The keystone service entity and API endpoints are created to enable and access the Identity service by internal service calls and from the external requests using the REST based API. The public, the internal and the admin endpoints are created to access the Identity service. These three endpoints are created across all further services installed.

Each service that you add to your OpenStack environment requires one or more service entities and three API endpoint variants in the Identity service.

While configuring the endpoints for the Identity service, the `region` is set to `TUChemnitz`. For example:

```
$ openstack endpoint create --region TUChemnitz identity public http
    ://controller:5000/v2.0
```

A point to remember is, do not duplicate the given endpoints for a given region. This has caused the issue with failure of the service.

Create the projects, users, and roles in the Identity service to access the services for `compute` users for user specific projects based on the access permissions.

The test bed has two projects (admin, demo) for test, 3 diferent users (admin, demo, service) to different access permissions defined by two roles (admin, user).

3.3.5 Add the Image service

The OpenStack Image service is central to Infrastructure-as-a-Service (IaaS) as shown in figure 2.3. It accepts API requests for disk or server images, and image metadata from end users or OpenStack Compute components. It also supports the storage of disk or server images on various repository types, including OpenStack Object Storage.

The SQL database is used to store the metadata information of the operating system image or the server image or the disk storage. There are templates available of specified requirements to create a virtual machine. These templates are called as flavors in OpenStack. The details of requirements to boot a virtual machine is defined in these flavors. For example, the hard disk capacity, the RAM capacity, the number of virtual cores, which would be required to satisfy the minimum requirements criteria of the desired operating system or to build a robust compute machine with a higher configurations a template is created to give as an input to the scheduler to create a virtual machine.

This service is installed on `controller` node. The glance service entity and the API endpoints are created. While configuring the endpoints for the Image service, the `region` is set to `TUChemnitz`. For example:

```
$ openstack endpoint create --region TUChemnitz image public http://
    controller:9292
```

The `cirros-0.3.4-x86_64-disk.img` disk image was downloaded and added to the image service as a bootable image of Cirros operating system to create and boot virtual machines.

3.3.6 Add the Compute service

OpenStack Compute is a major part of an Infrastructure-as-a-Service (IaaS) system. The main modules are implemented in Python.

OpenStack Compute interacts with OpenStack Identity for authentication, OpenStack Image service for disk and server images, and OpenStack dashboard for the user and administrative interface. Image access is limited by projects, and by users; quotas are limited per project (the number of instances, for example). OpenStack Compute can scale horizontally on standard hardware, and download images to launch instances.

The nova service entity and the API endpoints are created. While configuring the endpoints for the Compute service, the `region` is set to `TUChemnitz`. For example:

```
$ openstack endpoint create --region TUChemnitz compute public http
    ://controller:8774/v2/%\(tenant_id\)s
```

RabbitMQ which is the installed message queue server is used to pass the communication messages between the nodes and the other services. The MariaDB SQL database stores the build-time and run-time states of the whole cloud infrastructure like available instance, instances in use, available networks and different projects.

The OpenStack compute is installed with required packages on `controller` node which are different than the `compute` node. The important compute services like `nova-scheduler`, `nova-conductor`, `nova-api`, `nova-cert`, `nova-novncproxy` and `python-novaclient` are installed on the `controller` node. The configuration of RabbitMQ message service needs some additional parameters that needs to be added in the `/etc/nova/nova.conf` configuration file in the `controller` node under the `[oslo_messaging_rabbit]`.

```
[oslo_messaging_rabbit]
rabbit_host = controller
rabbit_port = 5672 #define ports
rabbit_hosts = controller:5672
rabbit_userid = openstack
rabbit_password = user
rabbit_use_ssl = false
```

The compute service failed to function when the `rabbit_port` was not defined in the configuration.

The OpenStack compute is installed with the `nova-compute` package on all the `compute` nodes. The configuration of RabbitMQ message service needs some additional parameters that needs to be added in the `/etc/nova/nova.conf` configuration

file in the compute nodes under the [oslo_messaging_rabbit].
Please refer to Appendix RabbitMQ configuration for the [oslo_messaging_rabbit] configuration parameters.

In the [DEFAULT] section of the /etc/nova/nova.conf file in compute nodes, configure the my_ip option with their respective management IP address of the compute nodes.

In the [vnc] section of the /etc/nova/nova.conf file in compute nodes, enable and configure remote console access by providing the public base url of the controller node to make it accessible from any machine outside the management network.

```
[vnc]
enabled = True
vncserver_listen = 0.0.0.0
vncserver_proxyclient_address = $my_ip
novncproxy_base_url = http://os-controller.etit.tu-chemnitz.de:6080/
    vnc_auto.html
```

Verify the operation of the Compute service by performing the commands as stated in the OpenStack Installation Guide for Ubuntu.

3.3.7 Add the Networking service

This section explains about installation and configuration of the OpenStack Networking service (neutron) using the self-service networks option. OpenStack Networking (neutron) allows you to create and attach interface devices managed by other OpenStack services to networks.

OpenStack Networking (neutron) manages all networking facets for the Virtual Networking Infrastructure (VNI) and the access layer aspects of the Physical Networking Infrastructure (PNI) in any given OpenStack environment. OpenStack Networking enables tenants to create advanced virtual network topologies which may include services such as a firewall, a load balancer, and a virtual private network (VPN).

Networking provides networks, subnets, and routers as object abstractions.

The neutron service entity and the API endpoints are created. While configuring the endpoints for the Networking service, the region is set to TUChemnitz. For example:

```
$ openstack endpoint create --region TUChemnitz network public http
    ://controller:9696
```

The Networking option 2: Self service networks is adapted for configuration of the test bed.

The required neutron networking components are installed on the controller node. The configuration of RabbitMQ message service needs some additional parameters that needs to be added in the /etc/neutron/neutron.conf configuration file in the controller node under the [oslo_messaging_rabbit].
Please refer to Appendix RabbitMQ configuration for the [oslo_messaging_rabbit] configuration parameters.

In the /etc/neutron/neutron.conf configuration file of the controller, under the [nova] section, the region_name is set to TUChemnitz.

```
[nova]
...
region_name = TUChemnitz
```

In the /etc/neutron/metadata_agent.ini file of the controller, under the [DEFAULT] section, the auth_region is set to TUChemnitz. The metadata_proxy_shared_-secret is set to user which will be used in /etc/nova/nova.conf configuration file.

```
[DEFAULT]
...
auth_region = TUChemnitz
...
metadata_proxy_shared_secret = user
```

Edit the /etc/nova/nova.conf configuration file to set the metadata_proxy_share-d_secret with the value of user.

```
[neutron]
...
region_name = TUChemnitz
...
metadata_proxy_shared_secret = user
```

Once the controller node has been set up, the common Networking components are installed on each of the compute nodes. The configuration of RabbitMQ message service needs some additional parameters that needs to be added in the /etc/neutron/n-eutron.conf configuration file in each of the compute node under the [oslo_messag-ing_rabbit].
Please refer to Appendix RabbitMQ configuration for the [oslo_messaging_rabbit] configuration parameters.

As the *Networking option 2: Self service networks* was set up in `controller` node, the Linux bridge agent, which builds layer-2 (bridging and switching) virtual networking infrastructure for instances including VXLAN tunnels for private networks and handles security groups, is configured on each `compute` node.

As in `controller` node, the `compute` nodes are configured to use the networking service by editing the `/etc/nova/nova.conf` configuration file on each `compute` node.

```
[neutron]
...
region_name = TUChemnitz
```

The configuration of networking service is verified to have all the services of networking to be active.

3.3.8 Add the dashboard

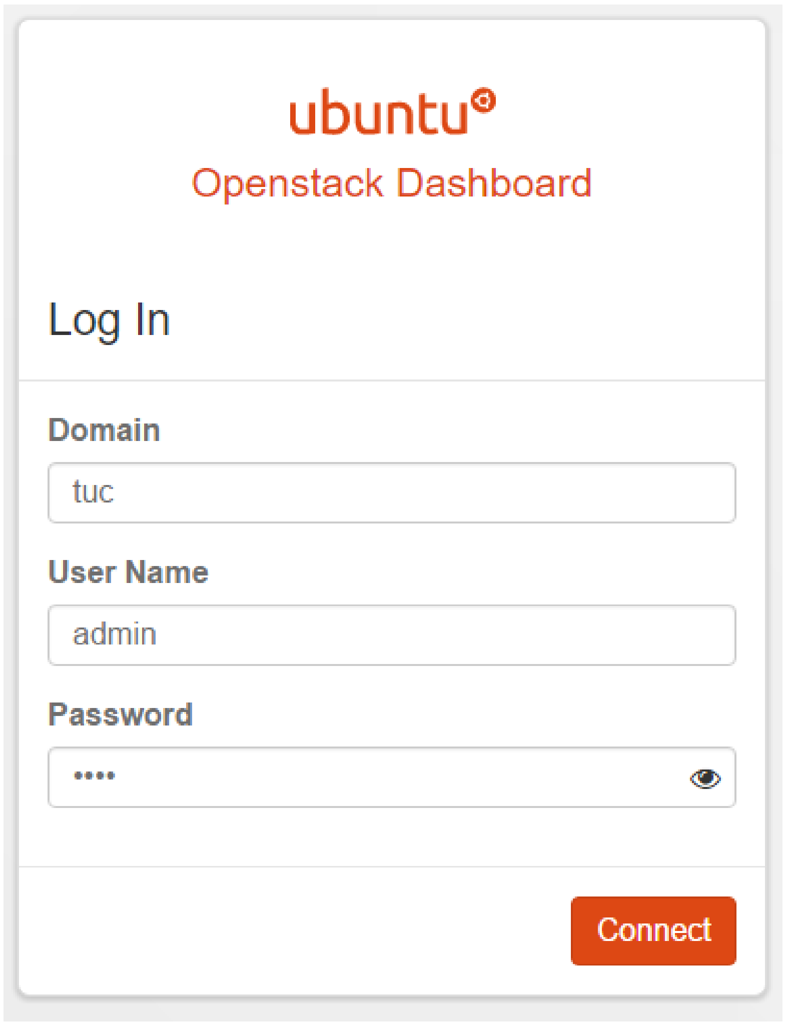
The OpenStack Dashboard, also known as horizon is a web interface that enables cloud administrators and users to manage various OpenStack resources and services.

The Dashboard enables web-based interactions with the OpenStack Compute cloud controller through the OpenStack APIs.

Horizon enables you to customize the brand of the dashboard.

The dashboard relies on functional core services including identity, image, compute, and neutron. The dashboard is installed on the `controller` node.

The figure 3.3 shows the login screen for the OpenStack dashboard.



Log In

Domain
tuc

User Name
admin

Password
••••

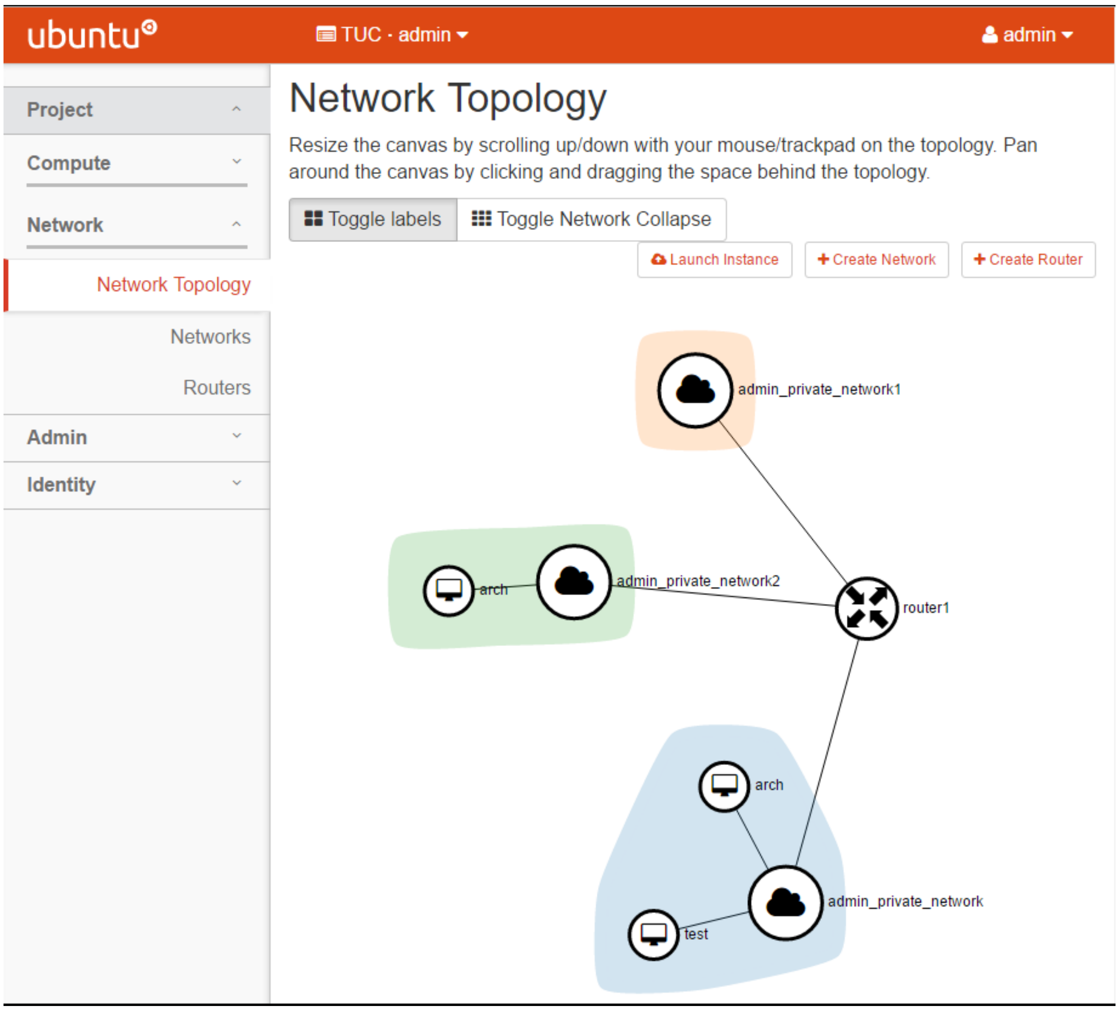
Figure 3.3: Login screen for the dashboard

The dashboard can be accessed by using the web browser at:
`http://os-controller.etit.tu-chemnitz.de/horizon`.

Provide the login credentials to authenticate the user against the Keystone to login into the Dashboard.

The login credentials are:
Domain: `TUC`
User Name: `admin` or `demo`
Password: user

The figure 3.4 shows the pictorial representation of the network and the instances created on the OpenStack under the link *Network Topology*.



Network Topology

Resize the canvas by scrolling up/down with your mouse/trackpad on the topology. Pan around the canvas by clicking and dragging the space behind the topology.

Figure 3.4: The dashboard screen showing the diagrammatic representation of network topology

3.3.9 Add the Telemetry service

The Object storage and Bloack storage services have not been deployed for the test bed. The next service deployed is the OpenStack Telemetry service.

Telemetry service is deployed to have an analysis of metering data with respect to Compute and Image services. Telemetry helps to efficiently poll the metering data related to OpenStack services.

Unlike other services, the Telemetry service uses a NoSQL database. The MongoDB NoSQL server has been installed on the controller to be utilised by the Telemetry service. The `password` for the MongoDB is set to `user`.

The ceilometer service entity and the API endpoints are created. While configuring the endpoints for the Telemetry service, the `region` is set to `TUChemnitz`. For example:

```
$ openstack endpoint create --region TUChemnitz metering public http
    ://controller:8777
```

The configuration of RabbitMQ message service needs some additional parameters that needs to be added in the `/etc/ceilometer/ceilometer.conf` configuration file on the `controller` node under the `[oslo_messaging_rabbit]`.
Please refer to Appendix RabbitMQ configuration for the `[oslo_messaging_rabbit]` configuration parameters.

In the `[service_credentials]` section, the `os_region_name` is set as `TUChemnitz`.

```
[service_credentials]
...
os_region_name = TUChemnitz
```

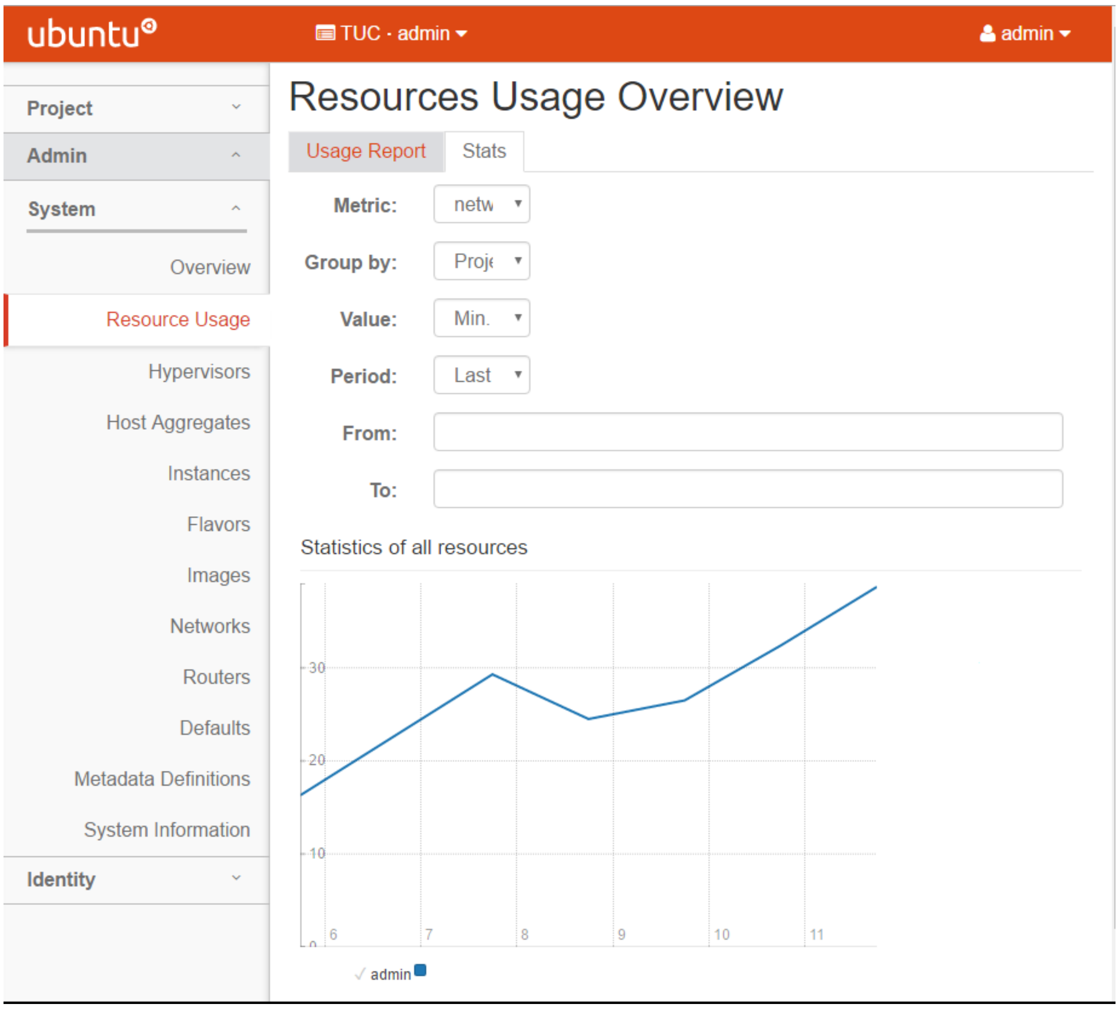
Once the Telemetry service has been installed on `controller`, it is configured to be consumed by Image service by editing the existing `/etc/glance/glance-api.conf` and `/etc/glance/glance-registry.conf` configuration files by adding the `[oslo_messaging_rabbit]`.
Please refer to Appendix RabbitMQ configuration for the `[oslo_messaging_rabbit]` configuration parameters.

Once the OpenStack installation procedure with additional configuration as mentioned above is configured, the Telemetry is for Image service.

To enable the Telemetry service for Compute service, the ceilometer should be installed on each `compute` node. Once the Telemetry service has been installed on all the `compute` nodes, it is configured to enable ceilometer by editing the `/etc/ceilometer-/ceilometer.conf` configuration file by adding the `[oslo_messaging_rabbit]`.
Please refer to Appendix RabbitMQ configuration for the `[oslo_messaging_rabbit]` configuration parameters.

In the `[service_credentials]` section, the `os_region_name` is set as `TUChemnitz`.

```
[service_credentials]
...
os_region_name = TUChemnitz
```



Resources Usage Overview

Metric: netw

Group by: Proje

Value: Min.

Period: Last

From:

To:

Statistics of all resources

Figure 3.5: An example of the metering with the number of incoming bytes on a network for a virtual machine interface.

All the above mentioned services have been installed and configured in the OpenStack test bed for this thesis.

4 Standard Nova Scheduling Algorithm

OpenStack Nova uses the `nova-scheduler` service to determine how to dispatch the compute requests which are requested by the user. It calculates and determines on which `compute` host node should the request for the virtual machine be processed. The scheduler can be configured through a variety of options provided by the OpenStack.

4.1 Theory of Nova Algorithm

As explained in Standard Nova Filter Scheduler, the nova scheduler has a **Filter Scheduler** which supports the filtering and weighting the hosts to make informed decisions about on which host a new instance should or can possibly be created.

The filter scheduler iterates over each host and validates if the available resources on the host can support the requested virtual instance. It creates a list of available hosts hosts which can serve the request to host an instance and proceeds with the filter mechanism.

The filter scheduler iterates over each available host, collects the resource information from the host and weighs according to the highest to lowest resource availability.

This filter scheduler in the Compute scheduler service is configured with the following default scheduler options in the `/etc/nova/nova.conf` file:

```
scheduler_driver_task_period = 60
scheduler_driver = nova.scheduler.filter_scheduler.FilterScheduler
scheduler_available_filters = nova.scheduler.filters.all_filters
scheduler_default_filters = RetryFilter, AvailabilityZoneFilter,
    RamFilter,DiskFilter, ComputeFilter, ComputeCapabilitiesFilter,
    ImagePropertiesFilter, ServerGroupAntiAffinityFilter,
    ServerGroupAffinityFilter
```

By default, the `scheduler_driver` is configured with a filter scheduler which can be found at `/usr/lib/python2.7/dist-packages/nova/scheduler/filter_scheduler.py` with the class name `FilterScheduler`.

In the default configuration, this scheduler considers hosts that meet all the following criteria:

- Any host that has not been attempted for scheduling purposes (RetryFilter).
- Any host that is available in the requested availability zone (AvailabilityZone-Filter).
- Any host that has sufficient availability of RAM to host an instance (RamFilter).
- Any host that has sufficient disk memory space available for root and ephemeral storage (DiskFilter).
- Any host that can and is available to service the request (ComputeFilter).
- Any host that can satisfy the extra specifications associated with the instance type (ComputeCapabilitiesFilter).
- Any host that can satisfy any architecture, hypervisor type, or virtual machine mode properties specified on the instance's image properties (ImageProperties-Filter).
- Any host that is on a different host than other instances of a group (if requested) (ServerGroupAntiAffinityFilter).
- Any host that is in a set of group hosts (if requested) (ServerGroupAffinityFilter).
- Any host that is visible in the refreshed scheduler cache list of available hosts; by the use of `scheduler_driver_task_period` option to specify how often the list is updated.

When the task of scheduling a virtual machine on a host is triggered by an user, the Filter Scheduler iterates over all found compute nodes, evaluating each against a set of filters. The resultant list of available hosts are ordered by the weighers.

At the end, the Filter Scheduler sorts selected hosts by their weight and attempts to provision instances on the chosen hosts.

4.2 Code Trace

In this section, the code trace to the nova scheduler can be observed.

When an instance is requested, it is received by the nova API listener for validation of the nova API URL and processes the requested tasks based on the requested input parameters. The Nova API reads the default scheduler file from the `/etc/nova/nova.conf` file. In the python source file `filter_scheduler`, the `FilterScheduler` class is initialized by the `driver.Scheduler` driver.

The scheduler driver object calls the `def select_destinations(self, context, request_spec, filter_properties)` function in the `FilterScheduler` class.

```
1  def select_destinations(self, context, request_spec,
       filter_properties):
2      """Selects a filtered set of hosts and nodes."""
3      self.notifier.info(context,
4          'scheduler.select_destinations.start',
5          dict(request_spec=request_spec))
6
7      # Count of requested virtual instances
8      num_instances = request_spec['num_instances']
9
10     # The FilterScheduler._schedule function is called,
11     # which schedules the possible hosts for the requested instances
12     selected_hosts = self._schedule(context, request_spec,
13         filter_properties)
14
15     ...
16     ...
```

Listing 4.1: The function select_destination

In the above code listing 4.1, the function `_schedule` called at line 12 initiates the scheduling algorithm by passing the `request_spec` as the parameter which holds all the specifications for creating the virtual instances like; requested number of instances, required RAM capacity per instance, required HDD capacity per instance, required VCPUs per instance, required OS image to boot the virtual instance with it, and other instance related parameters.

Stepping into the function `FilterScheduler._schedule` given in listing 4.2.

```
1  def _schedule(self, context, request_spec, filter_properties):
2      """Returns a list of hosts that meet the required specs,
3      ordered by their fitness.
```

```
 4      """
 5
 6      ...
 7      ...
 8
 9      # iterate each host and select the host to place an instance
10      for num in range(num_instances):
11          # Filter local hosts based on requirements ...
12          hosts = self.host_manager.get_filtered_hosts(hosts,
13                  filter_properties, index=num)
14          if not hosts:
15              # Can't get any more locally.
16              break
17
18          LOG.debug("Filtered %(hosts)s", {'hosts': hosts})
19
20          # weigh the hosts based on the weighing filter
21          weighed_hosts = self.host_manager.get_weighed_hosts(hosts,
22                  filter_properties)
23
24          ...
25          ...
26
27          chosen_host = random.choice(
28              weighed_hosts[0:scheduler_host_subset_size])
29          LOG.debug("Selected host: %(host)s", {'host': chosen_host})
30          selected_hosts.append(chosen_host)
31
32          ...
33          ...
34
35      LOG.info('%s number of instances scheduled with filter scheduler
       in %s seconds' % (num_instances, (rtime.time() - start_time)))
36      return selected_hosts
```

Listing 4.2: The function _schedule

In the listing 4.2, at the line number 12, the function `get_filtered_hosts` in the class `HostManager` in the python source file `host_manager` is called to filter the hosts based on the RAM filter, Disk Filter, Compute Filter, and other specified filters in the list.

Stepping into the `get_filtered_hosts` given in listing 4.3.

```
def get_filtered_hosts(self, hosts, filter_properties,
        filter_class_names=None, index=0):
    """Filter hosts and return only ones passing all filters."""
```

```
4
5      ...
6      ...
7
8      if filter_class_names is None:
9          filters = self.default_filters
10     else:
11         filters = self._choose_host_filters(filter_class_names)
12     ignore_hosts = filter_properties.get('ignore_hosts', [])
13     force_hosts = filter_properties.get('force_hosts', [])
14     force_nodes = filter_properties.get('force_nodes', [])
15
16     ...
17     ...
18     return self.filter_handler.get_filtered_objects(filters,
19             hosts, filter_properties, index)
```

Listing 4.3: The function get_filtered_hosts

In this listing 4.3, the filter classes are loaded either by default at line number 9 or by the specified list of filters at line number 11. These classes include the `nova.scheduler.filters.ram_filter`, `nova.scheduler.filters.disk_filter`, `nova.scheduler.filters.compute_filter` and other specified default filters.

In the listing 4.2, at the line number 21, the function `get_weighed_hosts` in the class `HostManager` in the python source file `host_manager` is called to get list of hosts sorted according to the weights calculated by the weights filter. At line 30 the chosen host is then set as selected host for the requested virtual instance.

The listing 4.4 calculates the weights of each host by comparing with the available properties of host and the requested properties of an instance as mentioned in the section 2.3.2

```
def get_weighed_hosts(self, hosts, weight_properties):
    """Weigh the hosts."""
    return self.weight_handler.get_weighed_objects(self.weighers,
            hosts, weight_properties)
```

Listing 4.4: The function _schedule

The `get_weighed_objects` looks for actual available resources on the hosts and calculates the weight of each objects.

The custom trace of the log can be observed in the Appendix Logs at FilterScheduler log trace.

5 Implementation of Cplex based Nova Scheduler

After the iterations of debugging the OpenStack Scheduler Python module, the key parameters which contribute in the calculation of the placement decisions of virtual instances are identified. The key parameters that would be required in computation of placements decision are:

- Available RAM capacity on each of the Compute node and
- Available Hard Disk capacity on each of the Compute node

The number of CPU cores are not taken into consideration. As the Compute nodes support the hardware acceleration for virtual machines using the KVM mode, the CPU cores can be over utilised. The default `cpu_allocation_ratio` is set to 16 which allows the over creation of virtual CPU cores, due to which the available CPU cores do not play a major role in determining the creation of virtual instances.

In this chapter, the formulation of the new cPlex based Mathematical model is explained, and also the implemented code is provided with an explanation.

5.1 Mathematical Formulation

The idea behind the mathematical formulation is to implement the energy efficient scheduler mechanism, where the placement decisions are made to utilise maximum capacity of the Compute node before a new placement decision is requested on the other Compute nodes. This was conceptualised with an idea to extend the algorithm with an ability to perform live migration of the existing virtual machines to re-order the placement decisions.

As the live migration is still a work in progress, the mathematical model is formulated with only the essential parameters into considerations.

Parameters for the mathematical model are defined as follows:
n_s is the host number in the set of nodes.
N_s is the set of all the host nodes.

n_v is the virtual instance number in the requested virtual instances list.
N_v is the set of all the requested virtual instances to be placed.
x_{n_s} is the each individual host node.
$x_{n_s}^{n_v}$ is the unique combination of requested virtual instance from the set of N_v and host node from the set of N_s.
$suit_{n_s}^{n_v}$ is the suitable host entry for the requested virtual instance.
$d_{n_v}^{RAM}$ is the required demand of RAM for the given virtual instance.
$c_{n_s}^{RAM}$ is the available capacity of RAM in the given host node.
$d_{n_v}^{HDD}$ is the required demand of hard disk for the given virtual instance.
$c_{n_s}^{HDD}$ is the available capacity of hard disk in the given host node.

The objective function is to minimize the host nodes for placing the requested virtual instances. This objective function can be given as:

$$\min \sum_{n_s \in N_s} x_{n_s} \tag{5.1}$$

For each virtual instance, the placement request of the virtual instance is done on one and only one host node. This can be given by the equation:

$$\sum_{n_s \in N_s} x_{n_s}^{n_v} suit_{n_s}^{n_v} = 1, \forall n_v \in N_v \tag{5.2}$$

The equation to determine if the host is suitable or not suitable for a given combination of virtual instance and host is given by:

$$x_{n_s}^{n_v} \leq suit_{n_s}^{n_v}, \forall n_s \in N_s, n_v \in N_v \tag{5.3}$$

The equation which evaluates if the placement decision can be made on the given host node or not for a given virtual instance is given by:

$$x_{n_s} \geq x_{n_s}^{n_v}, \forall n_s \in N_s, n_v \in N_v \tag{5.4}$$

The RAM capacity constraints are given as follows:

$$\sum_{n_v \in N_v} x_{ns}^{n_v} d_{n_v}^{RAM} \leq c_{n_s}^{RAM}, \forall n_s \in N_s \tag{5.5}$$

The hard disk(HDD) capacity constraints are given as follows:

$$\sum_{n_v \in N_v} x_{ns}^{n_v} d_{n_v}^{HDD} \leq c_{n_s}^{HDD}, \forall n_s \in N_s \tag{5.6}$$

These equations provide the boundary conditions to determine the possible placement decision of the virtual instances in the host nodes.

5.2 The overview of implementation

IBM ILOG CPLEX® Optimizer is a mathematical programming technology that enables decision of mathematical optimization for improving efficiency, reducing costs, and increasing profitability[15]. This software is used to provide the solution for placement decision of the virtual instances on the host nodes. For the purpose of this Thesis, the version 12.6 of the IBM Cplex is used.

To implement the mathematical formulation described in the section Mathematical Formulation, a new scheduler file is created with a name "`tuc_ccn_scheduler.py`".

The `tuc_ccn_scheduler.py` has the class and function definition similar to the default `FilterScheduler`. There are few changes in the implementation of the `_schedule` function and a new function named as `solve_TUC_Cplex` has been added to implement the new cPlex based mathematical scheduling model.

The `_schedule` function for the new `tuc_ccn_scheduler` is given as:

```python
def _schedule(self, context, request_spec, filter_properties):
    """Returns a list of hosts that meet the required specs,
    ordered by their fitness.
    """
    elevated = context.elevated()
    instance_properties = request_spec['instance_properties']

    # NOTE(danms): Instance here is still a dict, which is converted from
    # an object. The pci_requests are a dict as well. Convert this when
    # we get an object all the way to this path.
    # TODO(sbauza): Will be fixed later by the RequestSpec object
    pci_requests = instance_properties.get('pci_requests')
    if pci_requests:
        pci_requests = (
            objects.InstancePCIRequests.from_request_spec_instance_props(
                pci_requests))
        instance_properties['pci_requests'] = pci_requests

    instance_type = request_spec.get("instance_type", None)

    update_group_hosts = filter_properties.get('group_updated', False)

    config_options = self._get_configuration_options()

    filter_properties.update({'context': context,
                              'request_spec': request_spec,
                              'config_options': config_options,
                              'instance_type': instance_type})

    # Find our local list of acceptable hosts by repeatedly
    # filtering and weighing our options. Each time we choose a
    # host, we virtually consume resources on it so subsequent
    # selections can adjust accordingly.

    # Note: remember, we are using an iterator here. So only
    # traverse this list once. This can bite you if the hosts
    # are being scanned in a filter or weighing function.
    hosts = self._get_all_host_states(elevated)
    selected_hosts = []
    num_instances = request_spec.get('num_instances', 1)

    # the function get_filtered_hosts is called only once
    # before the for loop unlike the default scheduler's
    # _schedule function
    hosts = self.host_manager.get_filtered_hosts(hosts,
            filter_properties, index=0)

    weighed_hosts = self.weight_handler.get_weighed_objects(self.
weighers,
            hosts, filter_properties)

    vi_hosts = {}
    start_time = rtime.time()
    # implementation of cPlex based mathematical solver
    try:
        vi_hosts = self.solve_TUC_Cplex(hosts, filter_properties,
num_instances)
    except CplexError as exc:
        LOG.error('%s', exc)
        reason = _(exc)
        raise exception.NoValidHost(reason=reason)
    #end of the function call within try and exception

    for num in range(num_instances):
        weighed_hosts = vi_hosts[num]

        scheduler_host_subset_size = CONF.scheduler_host_subset_size

        if scheduler_host_subset_size > len(weighed_hosts):
            scheduler_host_subset_size = len(weighed_hosts)
        if scheduler_host_subset_size < 1:
            scheduler_host_subset_size = 1

```

```
72         chosen_host = weighed_hosts[0]
73         LOG.debug("Selected host: %(host)s", {'host': chosen_host})
74 
75         # append the selected hosts  to the array mapping
76         selected_hosts.append(chosen_host)
77 
78         # Now consume the resources so the filter/weights
79         # will change for the next instance.
80         chosen_host.obj.consume_from_instance(instance_properties)
81         if update_group_hosts is True:
82             # NOTE(sbauza): Group details are serialized into a list now
83             # that they are populated by the conductor, we need to
84             # deserialize them
85             if isinstance(filter_properties['group_hosts'], list):
86                 filter_properties['group_hosts'] = set(
87                     filter_properties['group_hosts'])
88             filter_properties['group_hosts'].add(chosen_host.obj.host)
89     LOG.info('%s number of instances scheduled with tuc scheduler in %s seconds' % (num_instances, (rtime.time() - start_time)))
90     return selected_hosts
```

Listing 5.1: The cPlex based TUC_scheduler's _schedule function

In the above code listing 5.1, the `get_filtered_hosts` at line number 45 is called before the for loop and called only once to provide an input to cPlex solver, unlike the default scheduler which filters the hosts for each request of an instance.

The cPlex based solver funtion is called at line number 55. The hosts, filter_properties and num_instances are passed as parameters to the function in the listing 5.2.

```
def solve_TUC_Cplex(self, hosts, filter_properties, num_instances):
    weighedHosts    = []
    #available RAM capacity on each compute (host) node
    ns_rams         = []
    #available HDD capacity on each compute (host) node
    ns_hdds         = []
    for host in hosts:
        weighedHost = []
        weighedHost.append(host)
        weighedHost = self.host_manager.get_weighed_hosts(weighedHost,
                filter_properties)
        weighedHosts.append(weighedHost)
        ns_rams.append(host.free_ram_mb)
        ns_hdds.append(host.free_disk_mb)

    instance_type = filter_properties['instance_type']
    root_gb      = instance_type['root_gb']
    memory_mb    = instance_type['memory_mb']
    nvram        = 1.0*memory_mb
    nvhdd        = 1024.0*root_gb
    my_prob      = cplex.Cplex()

    #Number of hosts available to cater the requested instances
    host_count   = len(hosts)
    vis          = num_instances

    # cPlex input parameters
    # objective function parameters
    my_obj       = []
    # upper bound values
    my_ub        = []
    # lower bound values
    my_lb        = []
    # parameter type
    my_ctype     = ''
    # names of all the mathematical parameters
    my_colnames = []
    # the values on the right hand side
    my_rhs       = []
    # unique name for each row
    my_rownames = []
    my_sense     = ''

    rows         =   []

    # translate the mathematical formulations into cPlex input
    for i in range(host_count):
        my_obj.append(1.0)
        my_colnames.append("x_"+str(i))
        my_ub.append(1.0)
        my_lb.append(0.0)
        my_ctype = my_ctype+'B'
        vvar  = []
        vvalr = []
        vvalh = []
        for j in range(vis):
            my_obj.append(0.0)
            my_colnames.append("x_"+str(i)+"_"+str(j))
            my_ub.append(1.0)
            my_rhs.append(0.0)
            my_ctype = my_ctype+'B'
            my_lb.append(0.0)
            row = []
            var = []
            val = []
            var.append("x_"+str(i))
            var.append("x_"+str(i)+"_"+str(j))
            val.append(1.0)
            val.append(-1.0)
            row.append(var)
            row.append(val)
            rows.append(row)
            my_rownames.append('rsv_'+str(i)+'_'+str(j))
            my_sense = my_sense+'G'
            vvar.append("x_"+str(i)+"_"+str(j))
            vvalr.append(nvram)
            vvalh.append(nvhdd)
        row = []
        row.append(vvar)
        row.append(vvalr)
        rows.append(row)
        my_rhs.append(ns_rams[i])
        my_sense = my_sense+'L'
        my_rownames.append('rsram_'+str(i))
        row = []
        row.append(vvar)
        row.append(vvalh)
        rows.append(row)
        my_rhs.append(ns_hdds[i])
        my_sense = my_sense+'L'
        my_rownames.append('rshdd_'+str(i))

    for i in range(vis):
        row = []
        var = []
        val = []
        for j in range(host_count):
            var.append("x_"+str(j)+"_"+str(i))
            val.append(1.0)
        my_sense = my_sense+'E'
        row.append(var)
        row.append(val)
        rows.append(row)
        my_rhs.append(1.0)
        my_rownames.append('rnv_'+str(i))

    my_prob.objective.set_sense(my_prob.objective.sense.minimize)

    my_prob.variables.add(obj=my_obj, lb=my_lb, ub=my_ub,
        types=my_ctype, names=my_colnames)

    # pass all parameters to cPlex to solve the equation
    my_prob.linear_constraints.add(lin_expr=rows, senses=my_sense,
        rhs=my_rhs, names=my_rownames)
    # solve the mathematical model
    my_prob.solve()

    numcols = my_prob.variables.get_num()
    numrows = my_prob.linear_constraints.get_num()

    slack = my_prob.solution.get_linear_slacks()
    # get the solution values
    x = my_prob.solution.get_values()

    vi_hosts = {}
    for i in range(vis):
        for j in range(host_count):
            vi_host = (j*vis+1+j)+i
            if x[vi_host] == 1.0:
                vi_hosts[i] = weighedHosts[j]

    #LOG.info('VI Hosts: %(vih)s', {'vih': vi_hosts})
    return vi_hosts
```

Listing 5.2: The cPlex based TUC_scheduler's solve_TUC_Cplex function

The above code listing 5.2 is a cPlex based approach to perform scheduling of virtual instances.

The comparisions of both the schedulers and their performances are provided in the chapter Comparision of performance evaluation.

6 Comparision of performance evaluation

In this chapter, observations are made based on both the schedulers. The data based on default nova scheduler driver and the data based on tuc_scheduler driver are evaluated for different iterations of scheduling of virtual instances. A data set of different number of virtual instance creation is performed on both of the scheduler drivers.

6.1 Observations of standard Nova scheduler

Observing the listing 4.2, at the line 12 it can be seen that the `get_filtered_hosts` is performed for each virtual instance scheduling rather than once for the whole of the request. This is also followed by line 21 to get weighed hosts for each virtual instance scheduling instead of once for the whole of the request.

In the log trace listing B.2 at the line number 185, it can be observed that, the placement decision of the 10 virtual instances is spread across all the host systems. This creates the need to keep the hosts powered up for all the time.

The log trace listing B.2 provided in the Appendix Logs in chapter FilterScheduler log trace for creation of 10 virtual instances shows the number of times the `get_filtered_hosts` and `get_weighed_hosts` is called to calculate the placement decision of 10 virtual instances.

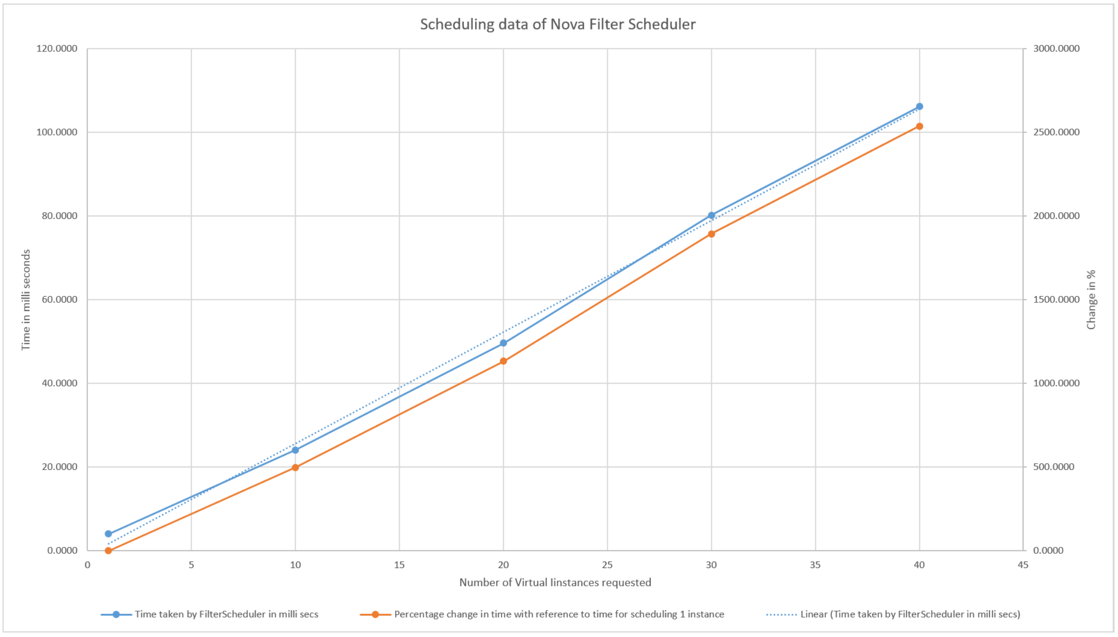


Figure 6.1: Placement decision time for Standard Nova Scheduler from the data C.1

The above table C.1 is the time logs recorded for different requested number of virtual instances. Here in the table, for providing a placement decision of 10 virtual instances, the standard nova scheduler takes around 24.10ms. The percentage change in time is given by the change in time with reference to time for scheduling 1 instance.

6.2 Observations of cPlex based scheduler

Observing the listing 5.1, at the line 45 it can be seen that the `get_filtered_hosts` is executed only once for all the bulk requests of virtual instance scheduling. This is followed by line 55 `solve_TUC_Cplex` to solve the scheduling problem for all the bulk requests.

In the log trace listing B.3 at the line number 39, it can be observed that, the placement decision of all the 10 virtual instances is concentrated on one host system "`compute03`".

The log trace listing B.3 provided in the Appendix Logs in chapter cPlex based Scheduler log trace for creation of 10 virtual instances shows that only one instance of the `get_filtered_hosts` is called to calculate the placement decision of 10 virtual instances.

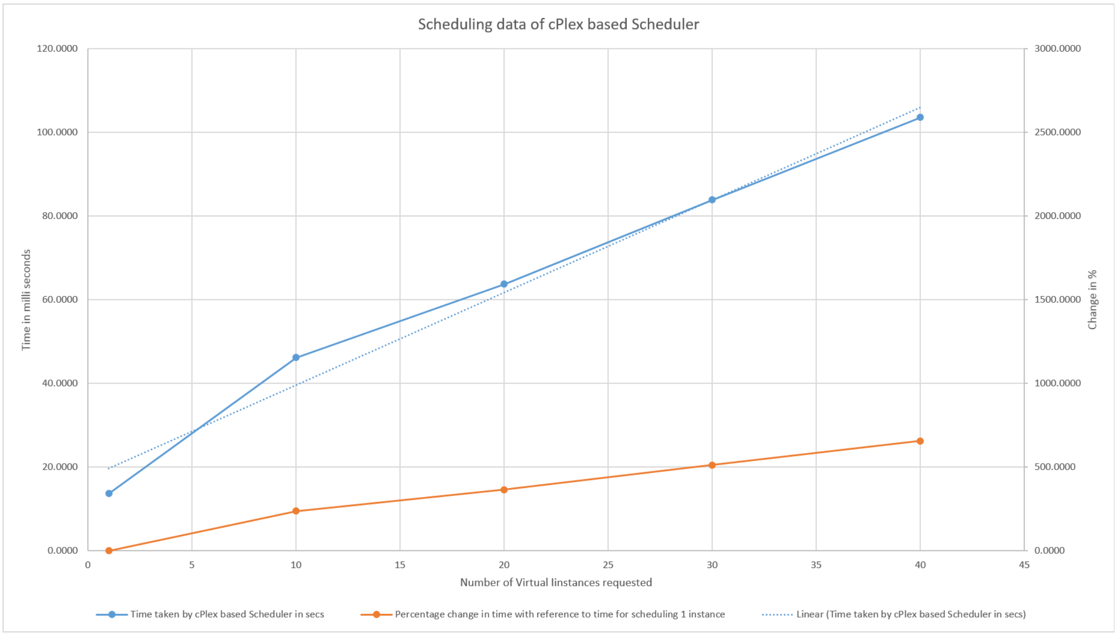


Figure 6.2: Placement decision time for cPlex based Scheduler from the data C.2

If the total requested capacity of RAM or HDD of virtual instances are more than the total available capacities, then the cPlex throws an exception as "unsolvable problem error". This exception is caught by the code and displayed as an error message on the screen.

6.3 Comparision

As stated in sections Observations of standard Nova scheduler and Observations of cPlex based scheduler, the default Filter scheduler places the virtual instances across the hosts, whereas the `tuc_ccn_scheduler` aggregates the creation of virtual instances on minimum possible host systems. As the virtual instances on the new `tuc_ccn_scheduler` are not spread across the multiple machines and aggregated on the few machines, the rest of the machines can be powered down which reduces the operating power costs.

On the other hand, combining the above two charts 6.1 and 6.2, it can be observed that the slope of the cplex based scheduler is lesser than the slope of the standard nova filter scheduler. Which means that, for a larger requests of virtual instances, the cPlex based scheduler would be more efficient in providing the placement decisions for all the requested virtual instances. The initial offset time for execution of cPlex to solve the mathematical problem is higher compared to the standard nova filter scheduler.

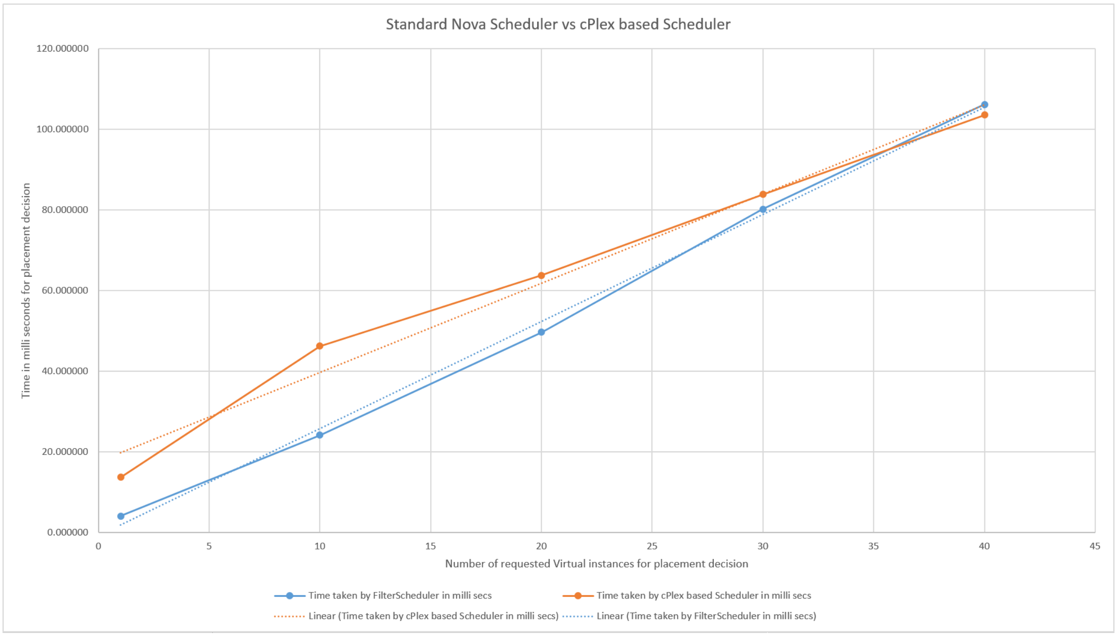


Figure 6.3: Comparision of placement decision time between Standard Nova FilterScheduler and cPlex based Scheduler

From the above chart 6.3, it can be concluded that, the higher the number of requests to schedule the virtual instances the lesser the solving time for scheduling compared to the nova filter scheduler.

6.4 Space for further improvements

As an example, let there be three host machines with a RAM capacity of 16GB each. Let there be 9 virtual machines which consume 4GB of RAM each and shared equally among three host machines. The available capacity of RAM is limited to 4GB on each machine whereas the total available capacity of RAM across all the machines is 12GB. When there is a request for a virtual machine which requires the RAM of 8GB, both the schedulers would fail to make a placement decision for the requested virtual instance as there is no host with an available capacity of 8GB.

"Live Migration" is a functionality which would move the running virtual machine from one physical host machine to another physical host machine with minimum or no downtime. Currently, the OpenStack Liberty has an ongoing issue with "Live Migration". The live migration functionality fails to migrate the virtual instance from the source host machine to requested destination host machine and turns off the virtual instance scheduled for migration. This "Live Migration" functionlity could mitigate the above mentioned issue with runtime reallocation of live(running) virtual machines to free the space by migrating it to any other possible hosts, freeing the space on the host and make a placement decision to allocate a new request of 8GB.

An idea for future could also be to have a shared resource pooling with a high speed dedicated networking bus on the hardware, to make the large clusters of compute servers into a single shared entity.

7 Further possible extensions in OpenStack

Being one of the key open source software platform for Cloud Computing in Infrastructure-as-a-Service model, it has various possible implementations and extensions. As far as networking is concerned, there are modules in and out of OpenStack which provides the Network Functions Virtualization (NFV) for global telecom providers.

Network Functions Virtualization (NFV) allows telecom and enterprise network operators to control their networking functions: physical, virtual and functional domains—using commercial off-the-shelf hardware, and open source software as a single control pane for management and orchestration. Here, OpenStack can provide a platform for the development and evolution of NFV components across the virtual systems. With robust system level integration and deployment a reference NFV platform can be created to accelerate the transformation of enterprise and service provider networks.

Early on, the telecommunications industries and the networking vendors have recognised the potential for OpenStack as a platform for NFV, which triggered investigations and work in development of OpenStack compatible modules to optimize for NFV.

"Network Functions Virtualization (NFV) is now synonymous with OpenStack. When people say NFV, there is an implication that they are talking about OpenStack."[13]

Both the European Telecommunications Standards Institute and Linux Foundation collaboration project OPNFV have defined specifications and released reference platforms for NFV that select OpenStack as the Virtualization Infrastructure Manager. Additionally, OpenStack is the dominant choice for additional management and orchestration functions.[16]

The interoperability between OpenStack and virtualized network functions is still an ongoing development.

So, what does Network Functions Virtualization (NFV) define? To answer in a simple way, it is a new way to define, create, and manage networks by replacing dedicated network appliances with software and automation. It is the idea of replacing the physical network devices which are dedicated and expensive by the virtual software devices.

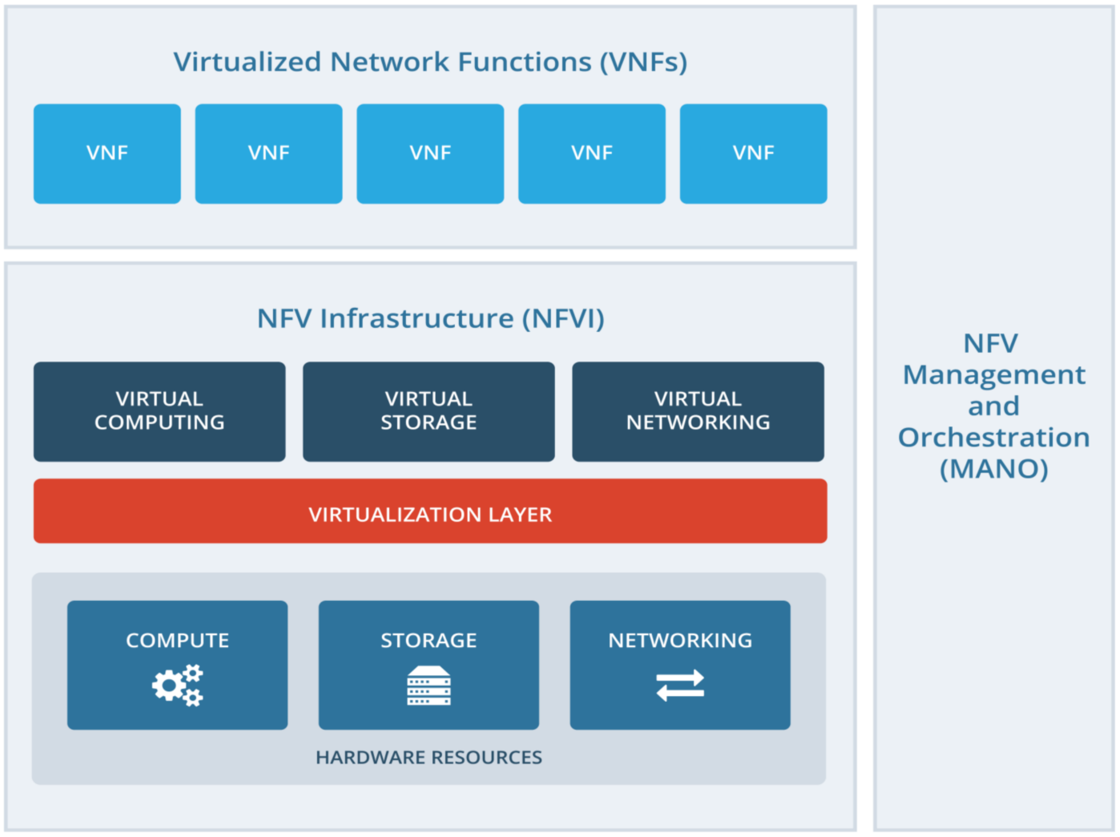


Figure 7.1: NFV functional overview[13]

In an NFV environment, a virtual network function (VNF) takes on the responsibility of handling specific network functions that run on one or more virtual machines (VMs), on bare metal, or in containers, on top of the physical networking infrastructure. A VNF can be an instance of any virtual hardware, for example: message router, CDN, DPI, Firewall, DNS...

The benefits of NFV stem from the fact that it runs on general purpose servers and switches in virtual machines or containers and is built with standard open APIs.

There are many ongoing NFV implementations. To keep it short, two of the NFV projects will be discussed.

- Open Baton
- Tacker

7.1 Open Baton - NFV Orchestrator

Open Baton is an European Telecommunications Standards Institute's (ETSI) NFV compliant Management and Orchestration (MANO) Framework. It enables virtual Network Services deployments on top of the NFV infrastructure.

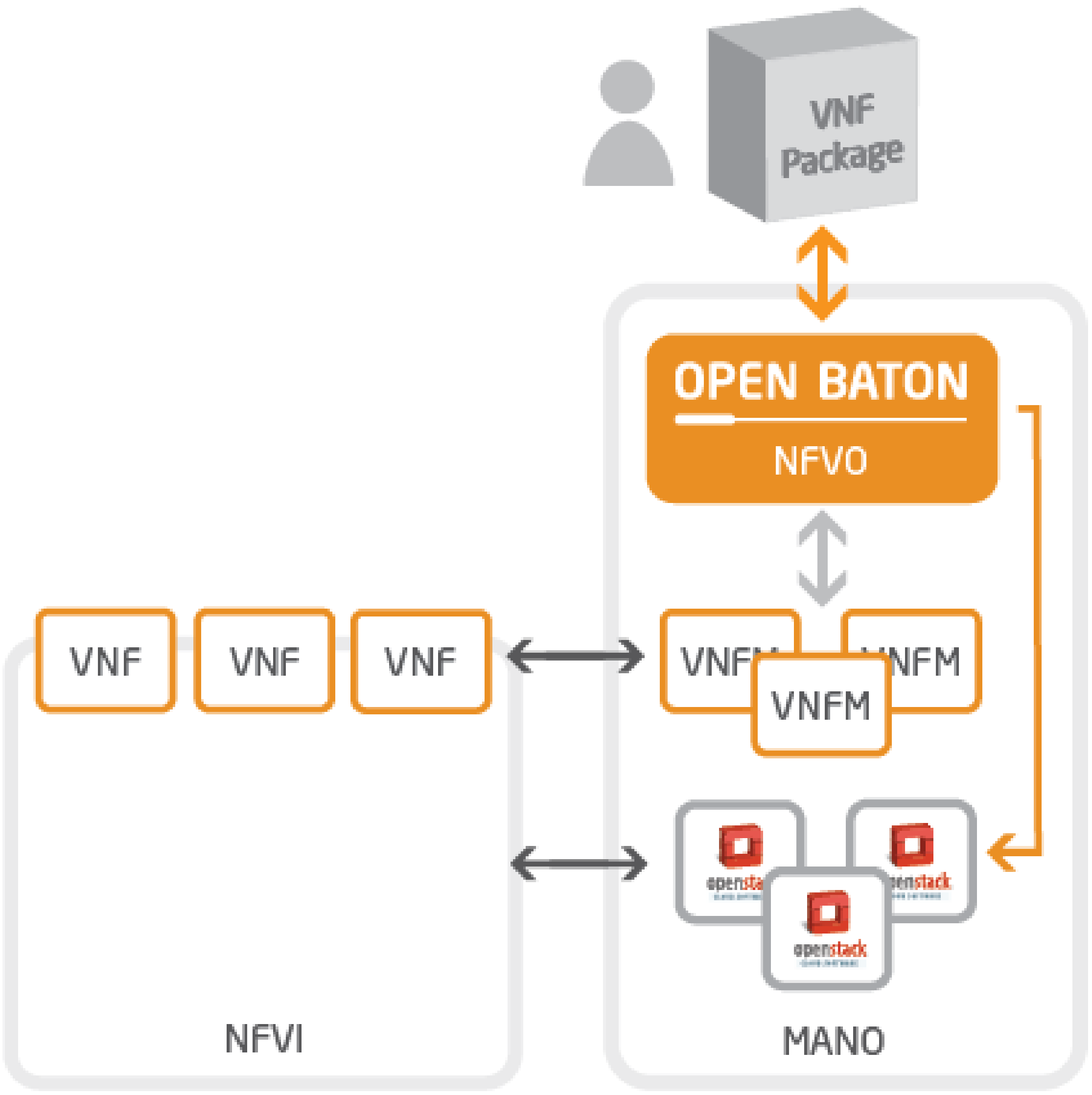


Figure 7.2: Placement of VNFM and VNF over the OpenStack Vitual Machines[14]

The Open Baton software is deployed over OpenStack with its own dashboard which would provide an easy to use web based VNF management console. The VNF is configured on the virtual machine in the OpenStack IaaS.

7.2 Tacker - OpenStack NFV Orchestration

Tacker is an official OpenStack project building a Generic VNF Manager (VNFM) and a NFV Orchestrator (NFVO) to deploy and operate Network Services and Virtual Network Functions (VNFs) on an NFV infrastructure platform like OpenStack. It is based on ETSI MANO Architectural Framework and provides a functional stack to Orchestrate Network Services end-to-end using VNFs.

8 Conclusion

OpenStack is an emerging and stable open source Cloud Infrastructure-as-a-Service operating solution. OpenStack, with it's usage as Compute service- to create cloud based computing or Storage service- for cloud based storage solutions or both combined, is an easily deployable cloud operating solution with minimum infrastructure to start hosting the Cloud service.

As this thesis is focussed on scheduling algorithm of requested virtual instances, it can be observed from the logs in Appendix B.2 that, with the standard nova FilterScheduler, when a large number of virtual instance creation is requested, it iterates for that number of times to provide a placement decision one after the another. There is an opportunity for improvement to solve a large requests with minimum time and better mathematical modelling.

In the section Comparision of the chapter Comparision of performance evaluation, it can be observed that the cPlex based scheduling algorithm is effective in providing a placement decision which would reduce the operating power expense of the OpenStack cloud cluster by minimising the active(powered on) number of hosts.

This is also effective in timely creation of large quantity of virtual instances compared to the existing standard nova scheduler.

This thesis is an approach to have a different possible solution in the scheduling algorithm which could be helpful to have constraint dependant scheduling.

This thesis can also be extended to have two different scheduling solutions depending on the number of requests for creation of the virtual instances. With a condition based on quantity of instance creation, the selection of the scheduler driver can be switched at runtime.

If the "Live Migration" could work effectively, the scheduling algorithm could also include the migration of existing instances to re-arrange itself which would increase the available capacity on the hosts and place new instances with larger configuration requirements. This could also be helpful for live migration of multiple virtual instances when the compute nodes needs a maintenance downtime.

The cloud computing is the growing field of interest which creates lots of opportunities for research and as well as the ideas for business.

Bibliography

[1] Cloud computing – apps on tap. URL http://www.salesforce.com/eu/cloudcomputing/. Online; accessed 06-November-2016.

[2] What is cloud computing?, . URL https://www.ibm.com/cloud-computing/what-is-cloud-computing. Online; accessed 06-November-2016.

[3] What is OpenStack?, . URL http://www.openstack.org/software/. Online; accessed 06-November-2016.

[4] What is cloud computing SaaS?, . URL http://www.ibm.com/cloud-computing/images/what_is_cc_SaaS_580x530.jpg. Online; accessed 06-November-2016.

[5] What is cloud computing PaaS?, . URL http://www.ibm.com/cloud-computing/images/what_is_cc_PaaS_580x425.jpg. Online; accessed 06-November-2016.

[6] What is cloud computing IaaS?, . URL http://www.ibm.com/cloud-computing/images/what_is_cc_IaaS_580x425-2.jpg. Online; accessed 06-November-2016.

[7] Openstack Software diagram, . URL http://www.openstack.org/themes/openstack/images/software/openstack-software-diagram.png. Online; accessed 06-November-2016.

[8] Virtual Machine States and Transitions, . URL http://docs.openstack.org/developer/nova/_images/graphviz-aa3b50f03fc95e5ea0896d5f2c606d809e3e2741.png. Online; accessed 06-November-2016.

[9] Filter Scheduler, . URL http://docs.openstack.org/developer/nova/_images/filteringWorkflow1.png. Online; accessed 06-November-2016.

[10] Filter Scheduler, . URL http://docs.openstack.org/developer/nova/_images/filteringWorkflow2.png. Online; accessed 06-November-2016.

[11] Overview, . URL http://docs.openstack.org/liberty/install-guide-ubuntu/_images/hwreqs.png. Online; accessed 06-November-2016.

[12] Host networking, . URL http://docs.openstack.org/liberty/install-guide-ubuntu/_images/networklayout.png. Online; accessed 06-November-2016.

[13] Accelerating NFV Delivery with OpenStack, OpenStack Foundation Report, . URL http://www.openstack.org/assets/telecoms-and-nfv/OpenStack-Foundation-NFV-Report.pdf. Online; accessed 05-November-2016.

[14] Open Baton features, . URL http://openbaton.github.io/images/feature1.png. Online; accessed 05-November-2016.

[15] CPLEX Optimizer. URL https://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/. Online; accessed 22-November-2016.

[16] Open Platform for NFV (OPNFV). URL https://www.opnfv.org/. Online; accessed 05-November-2016.

Appendix

Appendix A

Configuration Content

In this Appendix Configuration Content, the reader can find the specifically mentioned configuration.

A.1 RabbitMQ configuration

The RabbitMQ configuration parameters set in the `[oslo_messaging_rabbit]` are:

```
[oslo_messaging_rabbit]
rabbit_host = controller
rabbit_port = 5672 #define ports
rabbit_hosts = controller:5672
rabbit_userid = openstack
rabbit_password = user
rabbit_use_ssl = false
```

Appendix B

Logs

In this Appendix, the logs from the `scheduler.log` file have been provided for different log purpose.

B.1 FilterScheduler log trace

```
2016-10-23 22:04:46.471 14886 INFO nova.scheduler.filter_scheduler [
    req-6f237d8e-24c6-42ab-b996-ba9e7203601c 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: does it come here twice?
2016-10-23 22:04:46.472 14886 INFO nova.scheduler.filter_scheduler [
    req-6f237d8e-24c6-42ab-b996-ba9e7203601c 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: select_destinations FilterScheduler: for number of
    instances: 1
2016-10-23 22:04:46.481 14886 INFO nova.scheduler.host_manager [req-6
    f237d8e-24c6-42ab-b996-ba9e7203601c 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: get_all_host_states
2016-10-23 22:04:46.499 14886 INFO nova.scheduler.host_manager [req-6
    f237d8e-24c6-42ab-b996-ba9e7203601c 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: __init__: HostState
2016-10-23 22:04:46.501 14886 INFO nova.scheduler.host_manager [req-6
    f237d8e-24c6-42ab-b996-ba9e7203601c 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: _add_instance_info, host_manager = compute01
2016-10-23 22:04:46.519 14886 INFO nova.scheduler.host_manager [req-6
    f237d8e-24c6-42ab-b996-ba9e7203601c 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: __init__: HostState
2016-10-23 22:04:46.520 14886 INFO nova.scheduler.host_manager [req-6
    f237d8e-24c6-42ab-b996-ba9e7203601c 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: _add_instance_info, host_manager = compute03
```

```
2016-10-23 22:04:46.521 14886 INFO nova.scheduler.host_manager [req-6
    f237d8e-24c6-42ab-b996-ba9e7203601c 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: __init__: HostState
2016-10-23 22:04:46.521 14886 INFO nova.scheduler.host_manager [req-6
    f237d8e-24c6-42ab-b996-ba9e7203601c 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: _add_instance_info, host_manager = compute02
2016-10-23 22:04:46.550 14886 INFO nova.scheduler.host_manager [req-6
    f237d8e-24c6-42ab-b996-ba9e7203601c 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: __init__: HostState
2016-10-23 22:04:46.551 14886 INFO nova.scheduler.host_manager [req-6
    f237d8e-24c6-42ab-b996-ba9e7203601c 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: _add_instance_info, host_manager = compute04
2016-10-23 22:04:46.552 14886 INFO nova.scheduler.host_manager [req-6
    f237d8e-24c6-42ab-b996-ba9e7203601c 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: __init__: HostState
2016-10-23 22:04:46.552 14886 INFO nova.scheduler.host_manager [req-6
    f237d8e-24c6-42ab-b996-ba9e7203601c 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: _add_instance_info, host_manager = compute05
2016-10-23 22:04:46.553 14886 INFO nova.scheduler.filter_scheduler [
    req-6f237d8e-24c6-42ab-b996-ba9e7203601c 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: _schedule FilterScheduler:
2016-10-23 22:04:46.554 14886 INFO nova.scheduler.host_manager [req-6
    f237d8e-24c6-42ab-b996-ba9e7203601c 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: get_filtered_hosts: <dictionary-valueiterator object
    at 0x7f9578245fc8>
2016-10-23 22:04:46.554 14886 INFO nova.scheduler.filters.ram_filter
    [req-6f237d8e-24c6-42ab-b996-ba9e7203601c 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes: BaseRamFilter: 11598.0
2016-10-23 22:04:46.555 14886 INFO nova.scheduler.filters.ram_filter
    [req-6f237d8e-24c6-42ab-b996-ba9e7203601c 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes: BaseRamFilter: 48187.5
2016-10-23 22:04:46.555 14886 INFO nova.scheduler.filters.ram_filter
    [req-6f237d8e-24c6-42ab-b996-ba9e7203601c 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes: BaseRamFilter: 48081.0
2016-10-23 22:04:46.556 14886 INFO nova.scheduler.filters.ram_filter
    [req-6f237d8e-24c6-42ab-b996-ba9e7203601c 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes: BaseRamFilter: 23287.5
```

```
2016-10-23 22:04:46.556 14886 INFO nova.scheduler.filters.ram_filter [req-6f237d8e-24c6-42ab-b996-ba9e7203601c 12 b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 - - -] Hanif: host_passes: BaseRamFilter: 23946.0
2016-10-23 22:04:46.557 14886 INFO nova.scheduler.filters.disk_filter [req-6f237d8e-24c6-42ab-b996-ba9e7203601c 12 b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 - - -] Hanif: host_passes disk_filter: 911.0
2016-10-23 22:04:46.557 14886 INFO nova.scheduler.filters.disk_filter [req-6f237d8e-24c6-42ab-b996-ba9e7203601c 12 b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 - - -] Hanif: host_passes disk_filter: 219.0
2016-10-23 22:04:46.558 14886 INFO nova.scheduler.filters.disk_filter [req-6f237d8e-24c6-42ab-b996-ba9e7203601c 12 b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 - - -] Hanif: host_passes disk_filter: 884.0
2016-10-23 22:04:46.558 14886 INFO nova.scheduler.filters.disk_filter [req-6f237d8e-24c6-42ab-b996-ba9e7203601c 12 b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 - - -] Hanif: host_passes disk_filter: 229.0
2016-10-23 22:04:46.559 14886 INFO nova.scheduler.filters.disk_filter [req-6f237d8e-24c6-42ab-b996-ba9e7203601c 12 b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 - - -] Hanif: host_passes disk_filter: 458.0
2016-10-23 22:04:46.559 14886 INFO nova.scheduler.filters.compute_filter [req-6f237d8e-24c6-42ab-b996-ba9e7203601c 12 b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 - - -] Hanif: host_passes: ComputeFilter
2016-10-23 22:04:46.560 14886 INFO nova.scheduler.filters.compute_filter [req-6f237d8e-24c6-42ab-b996-ba9e7203601c 12 b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 - - -] Hanif: host_passes: ComputeFilter
2016-10-23 22:04:46.560 14886 INFO nova.scheduler.filters.compute_filter [req-6f237d8e-24c6-42ab-b996-ba9e7203601c 12 b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 - - -] Hanif: host_passes: ComputeFilter
2016-10-23 22:04:46.561 14886 INFO nova.scheduler.filters.compute_filter [req-6f237d8e-24c6-42ab-b996-ba9e7203601c 12 b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 - - -] Hanif: host_passes: ComputeFilter
2016-10-23 22:04:46.561 14886 INFO nova.scheduler.filters.compute_filter [req-6f237d8e-24c6-42ab-b996-ba9e7203601c 12 b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 - - -] Hanif: host_passes: ComputeFilter
2016-10-23 22:04:46.562 14886 INFO nova.scheduler.weights.metrics [req-6f237d8e-24c6-42ab-b996-ba9e7203601c 12 b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 - - -] Hanif: _weigh_object: value: 0.0, for host: compute04
```

```
2016−10−23 22:04:46.562 14886 INFO nova.scheduler.weights.metrics [
    req−6f237d8e−24c6−42ab−b996−ba9e7203601c 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 −
    − −] Hanif: _weigh_object: value: 0.0, for host: compute05
2016−10−23 22:04:46.563 14886 INFO nova.scheduler.weights.metrics [
    req−6f237d8e−24c6−42ab−b996−ba9e7203601c 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 −
    − −] Hanif: _weigh_object: value: 0.0, for host: compute02
2016−10−23 22:04:46.563 14886 INFO nova.scheduler.weights.metrics [
    req−6f237d8e−24c6−42ab−b996−ba9e7203601c 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 −
    − −] Hanif: _weigh_object: value: 0.0, for host: compute01
2016−10−23 22:04:46.564 14886 INFO nova.scheduler.weights.metrics [
    req−6f237d8e−24c6−42ab−b996−ba9e7203601c 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 −
    − −] Hanif: _weigh_object: value: 0.0, for host: compute03
2016−10−23 22:04:46.564 14886 INFO nova.scheduler.host_manager [req−6
    f237d8e−24c6−42ab−b996−ba9e7203601c 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 −
    − −] Hanif: consume_from_instance host_manager.py
2016−10−23 22:04:46.566 14886 INFO nova.scheduler.filter_scheduler [
    req−6f237d8e−24c6−42ab−b996−ba9e7203601c 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 −
    − −] 1 number of instances scheduled with filter scheduler in
    0.0127098560333 seconds
2016−10−23 22:04:46.568 14886 INFO nova.scheduler.manager [req−6
    f237d8e−24c6−42ab−b996−ba9e7203601c 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 −
    − −] Hanif: select_destinations SchedulerManager [{'host': u'
    compute05', 'nodename': u'compute05', 'limits': {'memory_mb':
    48187.5, 'disk_gb': 219.0}}]
```

Listing B.1: The filter scheduler code trace log

The trace starts with date and time of the logs, type of the log like, INFO for information, DEBUG for debug, WARN for warning and ERROR for error types of logs. The path of the python file is mentioned in the next column. At the end, the custom logs are printed with a message.

B.2 FilterScheduler log trace for creation of 10 virtual instances

```
2016−11−23 04:30:00.948 14886 INFO nova.scheduler.filter_scheduler [
    req−a2449432−6b13−444f−941c−0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 −
    - -] Hanif: select_destinations FilterScheduler: for number of
    instances: 10
2016-11-23 04:30:00.963 14886 INFO nova.scheduler.host_manager [req-
    a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: get_all_host_states
2016-11-23 04:30:00.984 14886 INFO nova.scheduler.host_manager [req-
    a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: _add_instance_info, host_manager = compute01
2016-11-23 04:30:00.990 14886 INFO nova.scheduler.host_manager [req-
    a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: _add_instance_info, host_manager = compute03
2016-11-23 04:30:00.993 14886 INFO nova.scheduler.host_manager [req-
    a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: _add_instance_info, host_manager = compute02
2016-11-23 04:30:00.994 14886 INFO nova.scheduler.host_manager [req-
    a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: _add_instance_info, host_manager = compute04
2016-11-23 04:30:00.995 14886 INFO nova.scheduler.host_manager [req-
    a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: _add_instance_info, host_manager = compute05
2016-11-23 04:30:00.996 14886 INFO nova.scheduler.filter_scheduler [
    req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: _schedule FilterScheduler:
2016-11-23 04:30:00.997 14886 INFO nova.scheduler.host_manager [req-
    a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: get_filtered_hosts: <dictionary-valueiterator object
    at 0x7f957819d838>
2016-11-23 04:30:00.998 14886 INFO nova.scheduler.filters.ram_filter
    [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes: BaseRamFilter: 11598.0
2016-11-23 04:30:00.998 14886 INFO nova.scheduler.filters.ram_filter
    [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes: BaseRamFilter: 48187.5
2016-11-23 04:30:00.998 14886 INFO nova.scheduler.filters.ram_filter
    [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes: BaseRamFilter: 48081.0
2016-11-23 04:30:00.999 14886 INFO nova.scheduler.filters.ram_filter
    [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes: BaseRamFilter: 23287.5
2016-11-23 04:30:00.999 14886 INFO nova.scheduler.filters.ram_filter
    [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes: BaseRamFilter: 23946.0
2016-11-23 04:30:00.999 14886 INFO nova.scheduler.filters.disk_filter
     [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes disk_filter: 911.0
2016-11-23 04:30:01.000 14886 INFO nova.scheduler.filters.disk_filter
     [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes disk_filter: 219.0
2016-11-23 04:30:01.000 14886 INFO nova.scheduler.filters.disk_filter
     [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes disk_filter: 884.0
2016-11-23 04:30:01.001 14886 INFO nova.scheduler.filters.disk_filter
     [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes disk_filter: 229.0
2016-11-23 04:30:01.001 14886 INFO nova.scheduler.filters.disk_filter
     [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes disk_filter: 458.0
2016-11-23 04:30:01.001 14886 INFO nova.scheduler.filters.
    compute_filter [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes: ComputeFilter
2016-11-23 04:30:01.002 14886 INFO nova.scheduler.filters.
    compute_filter [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes: ComputeFilter
2016-11-23 04:30:01.002 14886 INFO nova.scheduler.filters.
    compute_filter [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes: ComputeFilter
2016-11-23 04:30:01.003 14886 INFO nova.scheduler.filters.
    compute_filter [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes: ComputeFilter
2016-11-23 04:30:01.003 14886 INFO nova.scheduler.filters.
    compute_filter [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes: ComputeFilter
2016-11-23 04:30:01.004 14886 INFO nova.scheduler.weights.metrics [
    req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: _weigh_object: value: 0.0, for host: compute04
2016-11-23 04:30:01.005 14886 INFO nova.scheduler.weights.metrics [
    req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: _weigh_object: value: 0.0, for host: compute05
2016-11-23 04:30:01.006 14886 INFO nova.scheduler.weights.metrics [
    req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: _weigh_object: value: 0.0, for host: compute02
2016-11-23 04:30:01.007 14886 INFO nova.scheduler.weights.metrics [
    req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: _weigh_object: value: 0.0, for host: compute01
2016-11-23 04:30:01.007 14886 INFO nova.scheduler.weights.metrics [
    req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: _weigh_object: value: 0.0, for host: compute03
2016-11-23 04:30:01.007 14886 INFO nova.scheduler.host_manager [req-
    a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: consume_from_instance host_manager.py
2016-11-23 04:30:01.009 14886 INFO nova.scheduler.host_manager [req-
    a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: get_filtered_hosts: [(compute04, compute04) ram:7220
    disk:880640 io_ops:0 instances:0, (compute05, compute05) ram:31101
     disk:207872 io_ops:1 instances:1, (compute02, compute02) ram
    :31542 disk:855040 io_ops:0 instances:0, (compute01, compute01)
    ram:15013 disk:217088 io_ops:0 instances:0, (compute03, compute03)
     ram:15452 disk:440320 io_ops:0 instances:0]
2016-11-23 04:30:01.010 14886 INFO nova.scheduler.filters.ram_filter
    [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes: BaseRamFilter: 11598.0
2016-11-23 04:30:01.011 14886 INFO nova.scheduler.filters.ram_filter
    [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes: BaseRamFilter: 48187.5
2016-11-23 04:30:01.011 14886 INFO nova.scheduler.filters.ram_filter
    [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes: BaseRamFilter: 48081.0
2016-11-23 04:30:01.011 14886 INFO nova.scheduler.filters.ram_filter
    [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes: BaseRamFilter: 23287.5
2016-11-23 04:30:01.012 14886 INFO nova.scheduler.filters.ram_filter
    [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    − −] Hanif: host_passes: BaseRamFilter: 23946.0
2016−11−23 04:30:01.012 14886 INFO nova.scheduler.filters.disk_filter
     [req−a2449432−6b13−444f−941c−0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 −
    − −] Hanif: host_passes disk_filter: 911.0
2016−11−23 04:30:01.013 14886 INFO nova.scheduler.filters.disk_filter
     [req−a2449432−6b13−444f−941c−0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 −
    − −] Hanif: host_passes disk_filter: 219.0
2016−11−23 04:30:01.013 14886 INFO nova.scheduler.filters.disk_filter
     [req−a2449432−6b13−444f−941c−0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 −
    − −] Hanif: host_passes disk_filter: 884.0
2016−11−23 04:30:01.015 14886 INFO nova.scheduler.filters.disk_filter
     [req−a2449432−6b13−444f−941c−0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 −
    − −] Hanif: host_passes disk_filter: 229.0
2016−11−23 04:30:01.015 14886 INFO nova.scheduler.filters.disk_filter
     [req−a2449432−6b13−444f−941c−0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 −
    − −] Hanif: host_passes disk_filter: 458.0
2016−11−23 04:30:01.016 14886 INFO nova.scheduler.weights.metrics [
    req−a2449432−6b13−444f−941c−0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 −
    − −] Hanif: _weigh_object: value: 0.0, for host: compute04
2016−11−23 04:30:01.017 14886 INFO nova.scheduler.weights.metrics [
    req−a2449432−6b13−444f−941c−0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 −
    − −] Hanif: _weigh_object: value: 0.0, for host: compute05
2016−11−23 04:30:01.017 14886 INFO nova.scheduler.weights.metrics [
    req−a2449432−6b13−444f−941c−0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 −
    − −] Hanif: _weigh_object: value: 0.0, for host: compute02
2016−11−23 04:30:01.017 14886 INFO nova.scheduler.weights.metrics [
    req−a2449432−6b13−444f−941c−0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 −
    − −] Hanif: _weigh_object: value: 0.0, for host: compute01
2016−11−23 04:30:01.018 14886 INFO nova.scheduler.weights.metrics [
    req−a2449432−6b13−444f−941c−0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 −
    − −] Hanif: _weigh_object: value: 0.0, for host: compute03
2016−11−23 04:30:01.018 14886 INFO nova.scheduler.host_manager [req−
    a2449432−6b13−444f−941c−0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 −
    − −] Hanif: consume_from_instance host_manager.py
2016−11−23 04:30:01.021 14886 INFO nova.scheduler.host_manager [req−
    a2449432−6b13−444f−941c−0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 −
    − −] Hanif: get_filtered_hosts: [(compute04, compute04) ram:7220
    disk:880640 io_ops:0 instances:0, (compute05, compute05) ram:31101
     disk:207872 io_ops:1 instances:1, (compute02, compute02) ram
    :31030 disk:854016 io_ops:1 instances:1, (compute01, compute01)
    ram:15013 disk:217088 io_ops:0 instances:0, (compute03, compute03)
     ram:15452 disk:440320 io_ops:0 instances:0]
2016-11-23 04:30:01.022 14886 INFO nova.scheduler.filters.ram_filter
    [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes: BaseRamFilter: 11598.0
2016-11-23 04:30:01.022 14886 INFO nova.scheduler.filters.ram_filter
    [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes: BaseRamFilter: 48187.5
2016-11-23 04:30:01.023 14886 INFO nova.scheduler.filters.ram_filter
    [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes: BaseRamFilter: 48081.0
2016-11-23 04:30:01.023 14886 INFO nova.scheduler.filters.ram_filter
    [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes: BaseRamFilter: 23287.5
2016-11-23 04:30:01.024 14886 INFO nova.scheduler.filters.ram_filter
    [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes: BaseRamFilter: 23946.0
2016-11-23 04:30:01.024 14886 INFO nova.scheduler.filters.disk_filter
     [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes disk_filter: 911.0
2016-11-23 04:30:01.025 14886 INFO nova.scheduler.filters.disk_filter
     [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes disk_filter: 219.0
2016-11-23 04:30:01.025 14886 INFO nova.scheduler.filters.disk_filter
     [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes disk_filter: 884.0
2016-11-23 04:30:01.026 14886 INFO nova.scheduler.filters.disk_filter
     [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes disk_filter: 229.0
2016-11-23 04:30:01.026 14886 INFO nova.scheduler.filters.disk_filter
     [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes disk_filter: 458.0
2016-11-23 04:30:01.027 14886 INFO nova.scheduler.weights.metrics [
    req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: _weigh_object: value: 0.0, for host: compute04
```

```
2016-11-23 04:30:01.027 14886 INFO nova.scheduler.weights.metrics [
    req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: _weigh_object: value: 0.0, for host: compute05
2016-11-23 04:30:01.028 14886 INFO nova.scheduler.weights.metrics [
    req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: _weigh_object: value: 0.0, for host: compute02
2016-11-23 04:30:01.028 14886 INFO nova.scheduler.weights.metrics [
    req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: _weigh_object: value: 0.0, for host: compute01
2016-11-23 04:30:01.028 14886 INFO nova.scheduler.weights.metrics [
    req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: _weigh_object: value: 0.0, for host: compute03
2016-11-23 04:30:01.029 14886 INFO nova.scheduler.host_manager [req-
    a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: consume_from_instance host_manager.py
2016-11-23 04:30:01.031 14886 INFO nova.scheduler.host_manager [req-
    a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: get_filtered_hosts: [(compute04, compute04) ram:7220
    disk:880640 io_ops:0 instances:0, (compute05, compute05) ram:31101
     disk:207872 io_ops:1 instances:1, (compute02, compute02) ram
    :31030 disk:854016 io_ops:1 instances:1, (compute01, compute01)
    ram:15013 disk:217088 io_ops:0 instances:0, (compute03, compute03)
     ram:14940 disk:439296 io_ops:1 instances:1]
2016-11-23 04:30:01.032 14886 INFO nova.scheduler.filters.ram_filter
    [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes: BaseRamFilter: 11598.0
2016-11-23 04:30:01.032 14886 INFO nova.scheduler.filters.ram_filter
    [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes: BaseRamFilter: 48187.5
2016-11-23 04:30:01.033 14886 INFO nova.scheduler.filters.ram_filter
    [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes: BaseRamFilter: 48081.0
2016-11-23 04:30:01.033 14886 INFO nova.scheduler.filters.ram_filter
    [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes: BaseRamFilter: 23287.5
2016-11-23 04:30:01.034 14886 INFO nova.scheduler.filters.ram_filter
    [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes: BaseRamFilter: 23946.0
```

```
2016-11-23 04:30:01.035 14886 INFO nova.scheduler.filters.disk_filter
     [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes disk_filter: 911.0
2016-11-23 04:30:01.036 14886 INFO nova.scheduler.filters.disk_filter
     [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes disk_filter: 219.0
2016-11-23 04:30:01.036 14886 INFO nova.scheduler.filters.disk_filter
     [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes disk_filter: 884.0
2016-11-23 04:30:01.037 14886 INFO nova.scheduler.filters.disk_filter
     [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes disk_filter: 229.0
2016-11-23 04:30:01.037 14886 INFO nova.scheduler.filters.disk_filter
     [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes disk_filter: 458.0
2016-11-23 04:30:01.037 14886 INFO nova.scheduler.weights.metrics [
    req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: _weigh_object: value: 0.0, for host: compute04
2016-11-23 04:30:01.038 14886 INFO nova.scheduler.weights.metrics [
    req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: _weigh_object: value: 0.0, for host: compute05
2016-11-23 04:30:01.038 14886 INFO nova.scheduler.weights.metrics [
    req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: _weigh_object: value: 0.0, for host: compute02
2016-11-23 04:30:01.039 14886 INFO nova.scheduler.weights.metrics [
    req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: _weigh_object: value: 0.0, for host: compute01
2016-11-23 04:30:01.039 14886 INFO nova.scheduler.weights.metrics [
    req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: _weigh_object: value: 0.0, for host: compute03
2016-11-23 04:30:01.040 14886 INFO nova.scheduler.host_manager [req-
    a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: consume_from_instance host_manager.py
2016-11-23 04:30:01.042 14886 INFO nova.scheduler.host_manager [req-
    a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: get_filtered_hosts: [(compute04, compute04) ram:7220
    disk:880640 io_ops:0 instances:0, (compute05, compute05) ram:31101
    disk:207872 io_ops:1 instances:1, (compute02, compute02) ram
    :31030 disk:854016 io_ops:1 instances:1, (compute01, compute01)
    ram:14501 disk:216064 io_ops:1 instances:1, (compute03, compute03)
     ram:14940 disk:439296 io_ops:1 instances:1]
2016-11-23 04:30:01.043 14886 INFO nova.scheduler.filters.ram_filter
    [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes: BaseRamFilter: 11598.0
2016-11-23 04:30:01.044 14886 INFO nova.scheduler.filters.ram_filter
    [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes: BaseRamFilter: 48187.5
2016-11-23 04:30:01.044 14886 INFO nova.scheduler.filters.ram_filter
    [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes: BaseRamFilter: 48081.0
2016-11-23 04:30:01.044 14886 INFO nova.scheduler.filters.ram_filter
    [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes: BaseRamFilter: 23287.5
2016-11-23 04:30:01.045 14886 INFO nova.scheduler.filters.ram_filter
    [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes: BaseRamFilter: 23946.0
2016-11-23 04:30:01.045 14886 INFO nova.scheduler.filters.disk_filter
     [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes disk_filter: 911.0
2016-11-23 04:30:01.046 14886 INFO nova.scheduler.filters.disk_filter
     [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes disk_filter: 219.0
2016-11-23 04:30:01.046 14886 INFO nova.scheduler.filters.disk_filter
     [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes disk_filter: 884.0
2016-11-23 04:30:01.047 14886 INFO nova.scheduler.filters.disk_filter
     [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes disk_filter: 229.0
2016-11-23 04:30:01.047 14886 INFO nova.scheduler.filters.disk_filter
     [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes disk_filter: 458.0
2016-11-23 04:30:01.048 14886 INFO nova.scheduler.weights.metrics [
    req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: _weigh_object: value: 0.0, for host: compute04
```

```
2016-11-23 04:30:01.048 14886 INFO nova.scheduler.weights.metrics [req-a2449432-6b13-444f-941c-0ffd742e6737 12 b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 - - -] Hanif: _weigh_object: value: 0.0, for host: compute05
2016-11-23 04:30:01.049 14886 INFO nova.scheduler.weights.metrics [req-a2449432-6b13-444f-941c-0ffd742e6737 12 b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 - - -] Hanif: _weigh_object: value: 0.0, for host: compute02
2016-11-23 04:30:01.049 14886 INFO nova.scheduler.weights.metrics [req-a2449432-6b13-444f-941c-0ffd742e6737 12 b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 - - -] Hanif: _weigh_object: value: 0.0, for host: compute01
2016-11-23 04:30:01.050 14886 INFO nova.scheduler.weights.metrics [req-a2449432-6b13-444f-941c-0ffd742e6737 12 b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 - - -] Hanif: _weigh_object: value: 0.0, for host: compute03
2016-11-23 04:30:01.050 14886 INFO nova.scheduler.host_manager [req-a2449432-6b13-444f-941c-0ffd742e6737 12 b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 - - -] Hanif: consume_from_instance host_manager.py
2016-11-23 04:30:01.052 14886 INFO nova.scheduler.host_manager [req-a2449432-6b13-444f-941c-0ffd742e6737 12 b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 - - -] Hanif: get_filtered_hosts: [(compute04, compute04) ram:6708 disk:879616 io_ops:1 instances:1, (compute05, compute05) ram:31101 disk:207872 io_ops:1 instances:1, (compute02, compute02) ram:31030 disk:854016 io_ops:1 instances:1, (compute01, compute01) ram:14501 disk:216064 io_ops:1 instances:1, (compute03, compute03) ram:14940 disk:439296 io_ops:1 instances:1]
2016-11-23 04:30:01.053 14886 INFO nova.scheduler.filters.ram_filter [req-a2449432-6b13-444f-941c-0ffd742e6737 12 b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 - - -] Hanif: host_passes: BaseRamFilter: 11598.0
2016-11-23 04:30:01.054 14886 INFO nova.scheduler.filters.ram_filter [req-a2449432-6b13-444f-941c-0ffd742e6737 12 b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 - - -] Hanif: host_passes: BaseRamFilter: 48187.5
2016-11-23 04:30:01.055 14886 INFO nova.scheduler.filters.ram_filter [req-a2449432-6b13-444f-941c-0ffd742e6737 12 b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 - - -] Hanif: host_passes: BaseRamFilter: 48081.0
2016-11-23 04:30:01.056 14886 INFO nova.scheduler.filters.ram_filter [req-a2449432-6b13-444f-941c-0ffd742e6737 12 b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 - - -] Hanif: host_passes: BaseRamFilter: 23287.5
2016-11-23 04:30:01.057 14886 INFO nova.scheduler.filters.ram_filter [req-a2449432-6b13-444f-941c-0ffd742e6737 12 b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 - - -] Hanif: host_passes: BaseRamFilter: 23946.0
```

```
2016-11-23 04:30:01.057 14886 INFO nova.scheduler.filters.disk_filter [req-a2449432-6b13-444f-941c-0ffd742e6737 12 b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 - - -] Hanif: host_passes disk_filter: 911.0
2016-11-23 04:30:01.058 14886 INFO nova.scheduler.filters.disk_filter [req-a2449432-6b13-444f-941c-0ffd742e6737 12 b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 - - -] Hanif: host_passes disk_filter: 219.0
2016-11-23 04:30:01.058 14886 INFO nova.scheduler.filters.disk_filter [req-a2449432-6b13-444f-941c-0ffd742e6737 12 b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 - - -] Hanif: host_passes disk_filter: 884.0
2016-11-23 04:30:01.058 14886 INFO nova.scheduler.filters.disk_filter [req-a2449432-6b13-444f-941c-0ffd742e6737 12 b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 - - -] Hanif: host_passes disk_filter: 229.0
2016-11-23 04:30:01.059 14886 INFO nova.scheduler.filters.disk_filter [req-a2449432-6b13-444f-941c-0ffd742e6737 12 b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 - - -] Hanif: host_passes disk_filter: 458.0
2016-11-23 04:30:01.059 14886 INFO nova.scheduler.weights.metrics [req-a2449432-6b13-444f-941c-0ffd742e6737 12 b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 - - -] Hanif: _weigh_object: value: 0.0, for host: compute04
2016-11-23 04:30:01.060 14886 INFO nova.scheduler.weights.metrics [req-a2449432-6b13-444f-941c-0ffd742e6737 12 b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 - - -] Hanif: _weigh_object: value: 0.0, for host: compute05
2016-11-23 04:30:01.060 14886 INFO nova.scheduler.weights.metrics [req-a2449432-6b13-444f-941c-0ffd742e6737 12 b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 - - -] Hanif: _weigh_object: value: 0.0, for host: compute02
2016-11-23 04:30:01.060 14886 INFO nova.scheduler.weights.metrics [req-a2449432-6b13-444f-941c-0ffd742e6737 12 b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 - - -] Hanif: _weigh_object: value: 0.0, for host: compute01
2016-11-23 04:30:01.061 14886 INFO nova.scheduler.weights.metrics [req-a2449432-6b13-444f-941c-0ffd742e6737 12 b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 - - -] Hanif: _weigh_object: value: 0.0, for host: compute03
2016-11-23 04:30:01.061 14886 INFO nova.scheduler.host_manager [req-a2449432-6b13-444f-941c-0ffd742e6737 12 b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 - - -] Hanif: consume_from_instance host_manager.py
2016-11-23 04:30:01.063 14886 INFO nova.scheduler.host_manager [req-a2449432-6b13-444f-941c-0ffd742e6737 12 b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 - - -] Hanif: get_filtered_hosts: [(compute04, compute04) ram:6708 disk:879616 io_ops:1 instances:1, (compute05, compute05) ram:30589
```

```
     disk:206848 io_ops:2 instances:2, (compute02, compute02) ram
    :31030 disk:854016 io_ops:1 instances:1, (compute01, compute01)
    ram:14501 disk:216064 io_ops:1 instances:1, (compute03, compute03)
     ram:14940 disk:439296 io_ops:1 instances:1]
2016-11-23 04:30:01.064 14886 INFO nova.scheduler.filters.ram_filter
    [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes: BaseRamFilter: 11598.0
2016-11-23 04:30:01.065 14886 INFO nova.scheduler.filters.ram_filter
    [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes: BaseRamFilter: 48187.5
2016-11-23 04:30:01.065 14886 INFO nova.scheduler.filters.ram_filter
    [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes: BaseRamFilter: 48081.0
2016-11-23 04:30:01.066 14886 INFO nova.scheduler.filters.ram_filter
    [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes: BaseRamFilter: 23287.5
2016-11-23 04:30:01.066 14886 INFO nova.scheduler.filters.ram_filter
    [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes: BaseRamFilter: 23946.0
2016-11-23 04:30:01.067 14886 INFO nova.scheduler.filters.disk_filter
     [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes disk_filter: 911.0
2016-11-23 04:30:01.067 14886 INFO nova.scheduler.filters.disk_filter
     [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes disk_filter: 219.0
2016-11-23 04:30:01.068 14886 INFO nova.scheduler.filters.disk_filter
     [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes disk_filter: 884.0
2016-11-23 04:30:01.068 14886 INFO nova.scheduler.filters.disk_filter
     [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes disk_filter: 229.0
2016-11-23 04:30:01.069 14886 INFO nova.scheduler.filters.disk_filter
     [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes disk_filter: 458.0
2016-11-23 04:30:01.069 14886 INFO nova.scheduler.weights.metrics [
    req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: _weigh_object: value: 0.0, for host: compute04
```

```
2016-11-23 04:30:01.070 14886 INFO nova.scheduler.weights.metrics [
    req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: _weigh_object: value: 0.0, for host: compute05
2016-11-23 04:30:01.070 14886 INFO nova.scheduler.weights.metrics [
    req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: _weigh_object: value: 0.0, for host: compute02
2016-11-23 04:30:01.071 14886 INFO nova.scheduler.weights.metrics [
    req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: _weigh_object: value: 0.0, for host: compute01
2016-11-23 04:30:01.071 14886 INFO nova.scheduler.weights.metrics [
    req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: _weigh_object: value: 0.0, for host: compute03
2016-11-23 04:30:01.072 14886 INFO nova.scheduler.host_manager [req-
    a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: consume_from_instance host_manager.py
2016-11-23 04:30:01.074 14886 INFO nova.scheduler.host_manager [req-
    a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: get_filtered_hosts: [(compute04, compute04) ram:6708
    disk:879616 io_ops:1 instances:1, (compute05, compute05) ram:30589
     disk:206848 io_ops:2 instances:2, (compute02, compute02) ram
    :30518 disk:852992 io_ops:2 instances:2, (compute01, compute01)
    ram:14501 disk:216064 io_ops:1 instances:1, (compute03, compute03)
     ram:14940 disk:439296 io_ops:1 instances:1]
2016-11-23 04:30:01.075 14886 INFO nova.scheduler.filters.ram_filter
    [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes: BaseRamFilter: 11598.0
2016-11-23 04:30:01.076 14886 INFO nova.scheduler.filters.ram_filter
    [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes: BaseRamFilter: 48187.5
2016-11-23 04:30:01.076 14886 INFO nova.scheduler.filters.ram_filter
    [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes: BaseRamFilter: 48081.0
2016-11-23 04:30:01.077 14886 INFO nova.scheduler.filters.ram_filter
    [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes: BaseRamFilter: 23287.5
2016-11-23 04:30:01.077 14886 INFO nova.scheduler.filters.ram_filter
    [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes: BaseRamFilter: 23946.0
```

```
2016-11-23 04:30:01.078 14886 INFO nova.scheduler.filters.disk_filter
     [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes disk_filter: 911.0
2016-11-23 04:30:01.078 14886 INFO nova.scheduler.filters.disk_filter
     [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes disk_filter: 219.0
2016-11-23 04:30:01.079 14886 INFO nova.scheduler.filters.disk_filter
     [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes disk_filter: 884.0
2016-11-23 04:30:01.080 14886 INFO nova.scheduler.filters.disk_filter
     [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes disk_filter: 229.0
2016-11-23 04:30:01.080 14886 INFO nova.scheduler.filters.disk_filter
     [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes disk_filter: 458.0
2016-11-23 04:30:01.081 14886 INFO nova.scheduler.weights.metrics [
    req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: _weigh_object: value: 0.0, for host: compute04
2016-11-23 04:30:01.081 14886 INFO nova.scheduler.weights.metrics [
    req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: _weigh_object: value: 0.0, for host: compute05
2016-11-23 04:30:01.082 14886 INFO nova.scheduler.weights.metrics [
    req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: _weigh_object: value: 0.0, for host: compute02
2016-11-23 04:30:01.082 14886 INFO nova.scheduler.weights.metrics [
    req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: _weigh_object: value: 0.0, for host: compute01
2016-11-23 04:30:01.083 14886 INFO nova.scheduler.weights.metrics [
    req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: _weigh_object: value: 0.0, for host: compute03
2016-11-23 04:30:01.084 14886 INFO nova.scheduler.host_manager [req-
    a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: consume_from_instance host_manager.py
2016-11-23 04:30:01.086 14886 INFO nova.scheduler.host_manager [req-
    a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: get_filtered_hosts: [(compute04, compute04) ram:6708
    disk:879616 io_ops:1 instances:1, (compute05, compute05) ram:30589
    disk:206848 io_ops:2 instances:2, (compute02, compute02) ram
    :30518 disk:852992 io_ops:2 instances:2, (compute01, compute01)
    ram:14501 disk:216064 io_ops:1 instances:1, (compute03, compute03)
     ram:14428 disk:438272 io_ops:2 instances:2]
2016-11-23 04:30:01.087 14886 INFO nova.scheduler.filters.ram_filter
    [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes: BaseRamFilter: 11598.0
2016-11-23 04:30:01.088 14886 INFO nova.scheduler.filters.ram_filter
    [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes: BaseRamFilter: 48187.5
2016-11-23 04:30:01.088 14886 INFO nova.scheduler.filters.ram_filter
    [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes: BaseRamFilter: 48081.0
2016-11-23 04:30:01.089 14886 INFO nova.scheduler.filters.ram_filter
    [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes: BaseRamFilter: 23287.5
2016-11-23 04:30:01.089 14886 INFO nova.scheduler.filters.ram_filter
    [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes: BaseRamFilter: 23946.0
2016-11-23 04:30:01.090 14886 INFO nova.scheduler.filters.disk_filter
     [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes disk_filter: 911.0
2016-11-23 04:30:01.090 14886 INFO nova.scheduler.filters.disk_filter
     [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes disk_filter: 219.0
2016-11-23 04:30:01.091 14886 INFO nova.scheduler.filters.disk_filter
     [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes disk_filter: 884.0
2016-11-23 04:30:01.091 14886 INFO nova.scheduler.filters.disk_filter
     [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes disk_filter: 229.0
2016-11-23 04:30:01.093 14886 INFO nova.scheduler.filters.disk_filter
     [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes disk_filter: 458.0
2016-11-23 04:30:01.093 14886 INFO nova.scheduler.weights.metrics [
    req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: _weigh_object: value: 0.0, for host: compute04
```

```
2016-11-23 04:30:01.094 14886 INFO nova.scheduler.weights.metrics [
    req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: _weigh_object: value: 0.0, for host: compute05
2016-11-23 04:30:01.094 14886 INFO nova.scheduler.weights.metrics [
    req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: _weigh_object: value: 0.0, for host: compute02
2016-11-23 04:30:01.095 14886 INFO nova.scheduler.weights.metrics [
    req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: _weigh_object: value: 0.0, for host: compute01
2016-11-23 04:30:01.095 14886 INFO nova.scheduler.weights.metrics [
    req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: _weigh_object: value: 0.0, for host: compute03
2016-11-23 04:30:01.096 14886 INFO nova.scheduler.host_manager [req-
    a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: consume_from_instance host_manager.py
2016-11-23 04:30:01.098 14886 INFO nova.scheduler.host_manager [req-
    a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: get_filtered_hosts: [(compute04, compute04) ram:6708
    disk:879616 io_ops:1 instances:1, (compute05, compute05) ram:30077
     disk:205824 io_ops:3 instances:3, (compute02, compute02) ram
    :30518 disk:852992 io_ops:2 instances:2, (compute01, compute01)
    ram:14501 disk:216064 io_ops:1 instances:1, (compute03, compute03)
     ram:14428 disk:438272 io_ops:2 instances:2]
2016-11-23 04:30:01.099 14886 INFO nova.scheduler.filters.ram_filter
    [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes: BaseRamFilter: 11598.0
2016-11-23 04:30:01.100 14886 INFO nova.scheduler.filters.ram_filter
    [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes: BaseRamFilter: 48187.5
2016-11-23 04:30:01.100 14886 INFO nova.scheduler.filters.ram_filter
    [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes: BaseRamFilter: 48081.0
2016-11-23 04:30:01.101 14886 INFO nova.scheduler.filters.ram_filter
    [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes: BaseRamFilter: 23287.5
2016-11-23 04:30:01.102 14886 INFO nova.scheduler.filters.ram_filter
    [req-a2449432-6b13-444f-941c-0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes: BaseRamFilter: 23946.0
```

```
2016-11-23 04:30:01.102 14886 INFO nova.scheduler.filters.disk_filter [req-a2449432-6b13-444f-941c-0ffd742e6737 12 b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 - - -] Hanif: host_passes disk_filter: 911.0
2016-11-23 04:30:01.103 14886 INFO nova.scheduler.filters.disk_filter [req-a2449432-6b13-444f-941c-0ffd742e6737 12 b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 - - -] Hanif: host_passes disk_filter: 219.0
2016-11-23 04:30:01.103 14886 INFO nova.scheduler.filters.disk_filter [req-a2449432-6b13-444f-941c-0ffd742e6737 12 b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 - - -] Hanif: host_passes disk_filter: 884.0
2016-11-23 04:30:01.104 14886 INFO nova.scheduler.filters.disk_filter [req-a2449432-6b13-444f-941c-0ffd742e6737 12 b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 - - -] Hanif: host_passes disk_filter: 229.0
2016-11-23 04:30:01.104 14886 INFO nova.scheduler.filters.disk_filter [req-a2449432-6b13-444f-941c-0ffd742e6737 12 b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 - - -] Hanif: host_passes disk_filter: 458.0
2016-11-23 04:30:01.106 14886 INFO nova.scheduler.weights.metrics [req-a2449432-6b13-444f-941c-0ffd742e6737 12 b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 - - -] Hanif: _weigh_object: value: 0.0, for host: compute04
2016-11-23 04:30:01.107 14886 INFO nova.scheduler.weights.metrics [req-a2449432-6b13-444f-941c-0ffd742e6737 12 b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 - - -] Hanif: _weigh_object: value: 0.0, for host: compute05
2016-11-23 04:30:01.107 14886 INFO nova.scheduler.weights.metrics [req-a2449432-6b13-444f-941c-0ffd742e6737 12 b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 - - -] Hanif: _weigh_object: value: 0.0, for host: compute02
2016-11-23 04:30:01.108 14886 INFO nova.scheduler.weights.metrics [req-a2449432-6b13-444f-941c-0ffd742e6737 12 b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 - - -] Hanif: _weigh_object: value: 0.0, for host: compute01
2016-11-23 04:30:01.108 14886 INFO nova.scheduler.weights.metrics [req-a2449432-6b13-444f-941c-0ffd742e6737 12 b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 - - -] Hanif: _weigh_object: value: 0.0, for host: compute03
2016-11-23 04:30:01.109 14886 INFO nova.scheduler.host_manager [req-a2449432-6b13-444f-941c-0ffd742e6737 12 b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 - - -] Hanif: consume_from_instance host_manager.py
2016-11-23 04:30:01.113 14886 INFO nova.scheduler.filter_scheduler [req-a2449432-6b13-444f-941c-0ffd742e6737 12 b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 - - -] 10 number of instances scheduled with filter scheduler in 0.116468191147 seconds
```

```
2016−11−23 04:30:01.115 14886 INFO nova.scheduler.manager [req−
    a2449432−6b13−444f−941c−0ffd742e6737 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 −
    − −] Hanif: select_destinations SchedulerManager [{'host': u'
    compute05', 'nodename': u'compute05', 'limits': {'memory_mb':
    48187.5, 'disk_gb': 219.0}}, {'host': u'compute02', 'nodename': u'
    compute02', 'limits': {'memory_mb': 48081.0, 'disk_gb': 884.0}},
    {'host': u'compute03', 'nodename': u'compute03', 'limits': {'
    memory_mb': 23946.0, 'disk_gb': 458.0}}, {'host': u'compute01', '
    nodename': u'compute01', 'limits': {'memory_mb': 23287.5, 'disk_gb
    ': 229.0}}, {'host': u'compute04', 'nodename': u'compute04', '
    limits': {'memory_mb': 11598.0, 'disk_gb': 911.0}}, {'host': u'
    compute05', 'nodename': u'compute05', 'limits': {'memory_mb':
    48187.5, 'disk_gb': 219.0}}, {'host': u'compute02', 'nodename': u'
    compute02', 'limits': {'memory_mb': 48081.0, 'disk_gb': 884.0}},
    {'host': u'compute03', 'nodename': u'compute03', 'limits': {'
    memory_mb': 23946.0, 'disk_gb': 458.0}}, {'host': u'compute05', '
    nodename': u'compute05', 'limits': {'memory_mb': 48187.5, 'disk_gb
    ': 219.0}}, {'host': u'compute02', 'nodename': u'compute02', '
    limits': {'memory_mb': 48081.0, 'disk_gb': 884.0}}]
```

Listing B.2: The filter scheduler log trace for 10 virtual instances

B.3 cPlex based Scheduler log trace for creation of 10 virtual instances

```
2016−11−23 05:30:56.706 15170 INFO nova.scheduler.host_manager [req
    −8849ed8c−fd84−43bc−85de−9a65e9dcabbd 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 −
    − −] Hanif: get_all_host_states
2016−11−23 05:30:56.729 15170 INFO nova.scheduler.host_manager [req
    −8849ed8c−fd84−43bc−85de−9a65e9dcabbd 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 −
    − −] Hanif: _add_instance_info, host_manager = compute01
2016−11−23 05:30:56.754 15170 INFO nova.scheduler.host_manager [req
    −8849ed8c−fd84−43bc−85de−9a65e9dcabbd 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 −
    − −] Hanif: _add_instance_info, host_manager = compute03
2016−11−23 05:30:56.776 15170 INFO nova.scheduler.host_manager [req
    −8849ed8c−fd84−43bc−85de−9a65e9dcabbd 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 −
    − −] Hanif: _add_instance_info, host_manager = compute02
2016−11−23 05:30:56.800 15170 INFO nova.scheduler.host_manager [req
    −8849ed8c−fd84−43bc−85de−9a65e9dcabbd 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: _add_instance_info, host_manager = compute04
2016-11-23 05:30:56.823 15170 INFO nova.scheduler.host_manager [req
    -8849ed8c-fd84-43bc-85de-9a65e9dcabbd 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: _add_instance_info, host_manager = compute05
2016-11-23 05:30:56.853 15170 INFO nova.scheduler.host_manager [req
    -8849ed8c-fd84-43bc-85de-9a65e9dcabbd 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: get_filtered_hosts: <dictionary-valueiterator object
    at 0x7f548d5c57e0>
2016-11-23 05:30:56.854 15170 INFO nova.scheduler.filters.ram_filter
    [req-8849ed8c-fd84-43bc-85de-9a65e9dcabbd 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes: BaseRamFilter: 11598.0
2016-11-23 05:30:56.854 15170 INFO nova.scheduler.filters.ram_filter
    [req-8849ed8c-fd84-43bc-85de-9a65e9dcabbd 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes: BaseRamFilter: 48187.5
2016-11-23 05:30:56.855 15170 INFO nova.scheduler.filters.ram_filter
    [req-8849ed8c-fd84-43bc-85de-9a65e9dcabbd 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes: BaseRamFilter: 48081.0
2016-11-23 05:30:56.855 15170 INFO nova.scheduler.filters.ram_filter
    [req-8849ed8c-fd84-43bc-85de-9a65e9dcabbd 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes: BaseRamFilter: 23287.5
2016-11-23 05:30:56.855 15170 INFO nova.scheduler.filters.ram_filter
    [req-8849ed8c-fd84-43bc-85de-9a65e9dcabbd 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes: BaseRamFilter: 23946.0
2016-11-23 05:30:56.856 15170 INFO nova.scheduler.filters.disk_filter
     [req-8849ed8c-fd84-43bc-85de-9a65e9dcabbd 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes disk_filter: 911.0
2016-11-23 05:30:56.856 15170 INFO nova.scheduler.filters.disk_filter
     [req-8849ed8c-fd84-43bc-85de-9a65e9dcabbd 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes disk_filter: 219.0
2016-11-23 05:30:56.857 15170 INFO nova.scheduler.filters.disk_filter
     [req-8849ed8c-fd84-43bc-85de-9a65e9dcabbd 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes disk_filter: 884.0
2016-11-23 05:30:56.857 15170 INFO nova.scheduler.filters.disk_filter
     [req-8849ed8c-fd84-43bc-85de-9a65e9dcabbd 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes disk_filter: 229.0
2016-11-23 05:30:56.858 15170 INFO nova.scheduler.filters.disk_filter
     [req-8849ed8c-fd84-43bc-85de-9a65e9dcabbd 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes disk_filter: 458.0
2016-11-23 05:30:56.858 15170 INFO nova.scheduler.filters.
    compute_filter [req-8849ed8c-fd84-43bc-85de-9a65e9dcabbd 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes: ComputeFilter
2016-11-23 05:30:56.859 15170 INFO nova.scheduler.filters.
    compute_filter [req-8849ed8c-fd84-43bc-85de-9a65e9dcabbd 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes: ComputeFilter
2016-11-23 05:30:56.859 15170 INFO nova.scheduler.filters.
    compute_filter [req-8849ed8c-fd84-43bc-85de-9a65e9dcabbd 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes: ComputeFilter
2016-11-23 05:30:56.859 15170 INFO nova.scheduler.filters.
    compute_filter [req-8849ed8c-fd84-43bc-85de-9a65e9dcabbd 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes: ComputeFilter
2016-11-23 05:30:56.860 15170 INFO nova.scheduler.filters.
    compute_filter [req-8849ed8c-fd84-43bc-85de-9a65e9dcabbd 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: host_passes: ComputeFilter
2016-11-23 05:30:56.860 15170 INFO nova.scheduler.weights.metrics [
    req-8849ed8c-fd84-43bc-85de-9a65e9dcabbd 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: _weigh_object: value: 0.0, for host: compute04
2016-11-23 05:30:56.861 15170 INFO nova.scheduler.weights.metrics [
    req-8849ed8c-fd84-43bc-85de-9a65e9dcabbd 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: _weigh_object: value: 0.0, for host: compute05
2016-11-23 05:30:56.861 15170 INFO nova.scheduler.weights.metrics [
    req-8849ed8c-fd84-43bc-85de-9a65e9dcabbd 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: _weigh_object: value: 0.0, for host: compute02
2016-11-23 05:30:56.862 15170 INFO nova.scheduler.weights.metrics [
    req-8849ed8c-fd84-43bc-85de-9a65e9dcabbd 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: _weigh_object: value: 0.0, for host: compute01
2016-11-23 05:30:56.863 15170 INFO nova.scheduler.weights.metrics [
    req-8849ed8c-fd84-43bc-85de-9a65e9dcabbd 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: _weigh_object: value: 0.0, for host: compute03
2016-11-23 05:30:56.985 15170 INFO nova.scheduler.host_manager [req
    -8849ed8c-fd84-43bc-85de-9a65e9dcabbd 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: consume_from_instance host_manager.py
2016-11-23 05:30:56.987 15170 INFO nova.scheduler.host_manager [req
    -8849ed8c-fd84-43bc-85de-9a65e9dcabbd 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: consume_from_instance host_manager.py
2016-11-23 05:30:56.989 15170 INFO nova.scheduler.host_manager [req
    -8849ed8c-fd84-43bc-85de-9a65e9dcabbd 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: consume_from_instance host_manager.py
2016-11-23 05:30:56.992 15170 INFO nova.scheduler.host_manager [req
    -8849ed8c-fd84-43bc-85de-9a65e9dcabbd 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: consume_from_instance host_manager.py
2016-11-23 05:30:56.994 15170 INFO nova.scheduler.host_manager [req
    -8849ed8c-fd84-43bc-85de-9a65e9dcabbd 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: consume_from_instance host_manager.py
2016-11-23 05:30:56.996 15170 INFO nova.scheduler.host_manager [req
    -8849ed8c-fd84-43bc-85de-9a65e9dcabbd 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: consume_from_instance host_manager.py
2016-11-23 05:30:56.999 15170 INFO nova.scheduler.host_manager [req
    -8849ed8c-fd84-43bc-85de-9a65e9dcabbd 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: consume_from_instance host_manager.py
2016-11-23 05:30:57.001 15170 INFO nova.scheduler.host_manager [req
    -8849ed8c-fd84-43bc-85de-9a65e9dcabbd 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: consume_from_instance host_manager.py
2016-11-23 05:30:57.003 15170 INFO nova.scheduler.host_manager [req
    -8849ed8c-fd84-43bc-85de-9a65e9dcabbd 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: consume_from_instance host_manager.py
2016-11-23 05:30:57.005 15170 INFO nova.scheduler.host_manager [req
    -8849ed8c-fd84-43bc-85de-9a65e9dcabbd 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] Hanif: consume_from_instance host_manager.py
2016-11-23 05:30:57.007 15170 INFO nova.scheduler.tuc_ccn_scheduler [
    req-8849ed8c-fd84-43bc-85de-9a65e9dcabbd 12
    b8f78400724c8fa549aec66ad1a43a 9ea57ff8c67544b78c8096ea4dc1d081 -
    - -] 10 number of instances scheduled with tuc scheduler in
    0.142965078354 seconds
2016-11-23 05:30:57.009 15170 INFO nova.scheduler.manager [req-8849
    ed8c-fd84-43bc-85de-9a65e9dcabbd 12b8f78400724c8fa549aec66ad1a43a
    9ea57ff8c67544b78c8096ea4dc1d081 - - -] Hanif: select_destinations
     SchedulerManager [{'host': u'compute03', 'nodename': u'compute03
    ', 'limits': {'memory_mb': 23946.0, 'disk_gb': 458.0}}, {'host': u
    'compute03', 'nodename': u'compute03', 'limits': {'memory_mb':
    23946.0, 'disk_gb': 458.0}}, {'host': u'compute03', 'nodename': u'
    compute03', 'limits': {'memory_mb': 23946.0, 'disk_gb': 458.0}},
    {'host': u'compute03', 'nodename': u'compute03', 'limits': {'
    memory_mb': 23946.0, 'disk_gb': 458.0}}, {'host': u'compute03', '
    nodename': u'compute03', 'limits': {'memory_mb': 23946.0, 'disk_gb
    ': 458.0}}, {'host': u'compute03', 'nodename': u'compute03', '
    limits': {'memory_mb': 23946.0, 'disk_gb': 458.0}}, {'host': u'
    compute03', 'nodename': u'compute03', 'limits': {'memory_mb':
    23946.0, 'disk_gb': 458.0}}, {'host': u'compute03', 'nodename': u'
    compute03', 'limits': {'memory_mb': 23946.0, 'disk_gb': 458.0}},
    {'host': u'compute03', 'nodename': u'compute03', 'limits': {'
    memory_mb': 23946.0, 'disk_gb': 458.0}}, {'host': u'compute03', '
    nodename': u'compute03', 'limits': {'memory_mb': 23946.0, 'disk_gb
    ': 458.0}}]
```

Listing B.3: The cPlex based scheduler log trace for 10 virtual instances

Appendix C

Performance Data

In this Appendix Performance Data, the time required for placement decision of the virtual instances is captured in the logs of the `scheduler.log` file and have been used for comparision and performance evaluation.

C.1 FilterScheduler's Scheduling Time Logs Data

The time taken for the FilterScheduler for placement decision of virtual instances on the hosts for different number of data set has been recorded.

```
2016-11-23 06:17:59.466 17314 INFO nova.scheduler.filter_scheduler:
    Hanif: _schedule FilterScheduler:
2016-11-23 06:17:59.471 17314 INFO nova.scheduler.filter_scheduler: 1
     number of instances scheduled with filter scheduler in
    0.00402498245239 seconds

2016-11-23 06:23:39.579 17314 INFO nova.scheduler.filter_scheduler:
    Hanif: _schedule FilterScheduler:
2016-11-23 06:23:39.590 17314 INFO nova.scheduler.filter_scheduler: 5
     number of instances scheduled with filter scheduler in
    0.0108880996704 seconds

2016-11-23 06:25:55.949 17314 INFO nova.scheduler.filter_scheduler:
    Hanif: _schedule FilterScheduler:
2016-11-23 06:25:55.975 17314 INFO nova.scheduler.filter_scheduler:
    10 number of instances scheduled with filter scheduler in
    0.0240979194641 seconds

2016-11-23 06:37:36.863 17314 INFO nova.scheduler.filter_scheduler:
    Hanif: _schedule FilterScheduler:
2016-11-23 06:37:36.914 17314 INFO nova.scheduler.filter_scheduler:
    20 number of instances scheduled with filter scheduler in
    0.0496470928192 seconds
```

```

2016-11-23 06:41:16.201 17314 INFO nova.scheduler.filter_scheduler:
    Hanif: _schedule FilterScheduler:
2016-11-23 06:41:16.282 17314 INFO nova.scheduler.filter_scheduler:
    30 number of instances scheduled with filter scheduler in
    0.0802478790283 seconds

2016-11-23 06:50:20.638 17314 INFO nova.scheduler.filter_scheduler:
    Hanif: _schedule FilterScheduler:
2016-11-23 06:50:20.746 17314 INFO nova.scheduler.filter_scheduler 40
     number of instances scheduled with filter scheduler in
    0.10618185997 seconds
```

Listing C.1: The filter scheduler based scheduler time logs for scheduling different numbers of virtual instances

Requested Number of Instances	Time taken by Filter-Scheduler in secs	Percentage change in time with reference to time taken for scheduling 1 instance
1	4.03ms	0.00%
10	24.10ms	498.70%
20	49.65ms	1133.47%
30	80.25ms	1893.74%
40	106.18ms	2538.07%

Table C.1: FilterScheduler Time Data

C.2 TUC CCN Scheduler's Scheduling Time Logs Data

The time taken for the TUC_CCN_Scheduler for placement decision of virtual instances on the hosts for different numbers of data set has been recorded.

```
2016-11-23 07:01:33.293 27964 INFO nova.scheduler.tuc_ccn_scheduler:
    Hanif: _schedule tuc_ccn_scheduler
2016-11-23 07:01:33.310 27964 INFO nova.scheduler.tuc_ccn_scheduler:
    1 number of instances scheduled with tuc scheduler in
    0.0136959552765 seconds

```

```
2016-11-23 07:25:37.228 27964 INFO nova.scheduler.tuc_ccn_scheduler:
    Hanif: _schedule tuc_ccn_scheduler
2016-11-23 07:25:37.283 27964 INFO nova.scheduler.tuc_ccn_scheduler:
    5 number of instances scheduled with tuc scheduler in
    0.049889087677 seconds

2016-11-23 07:12:07.047 27964 INFO nova.scheduler.tuc_ccn_scheduler:
    Hanif: _schedule tuc_ccn_scheduler
2016-11-23 07:12:07.097 27964 INFO nova.scheduler.tuc_ccn_scheduler:
    10 number of instances scheduled with tuc scheduler in
    0.0462019443512 seconds

2016-11-23 07:28:47.675 27964 INFO nova.scheduler.tuc_ccn_scheduler:
    Hanif: _schedule tuc_ccn_scheduler
2016-11-23 07:28:47.741 27964 INFO nova.scheduler.tuc_ccn_scheduler:
    20 number of instances scheduled with tuc scheduler in
    0.0637698173523 seconds

2016-11-23 07:37:45.585 27964 INFO nova.scheduler.tuc_ccn_scheduler:
    Hanif: _schedule tuc_ccn_scheduler
2016-11-23 07:37:45.672 27964 INFO nova.scheduler.tuc_ccn_scheduler:
    30 number of instances scheduled with tuc scheduler in
    0.0839061737061 seconds

2016-11-23 08:11:37.874 27964 INFO nova.scheduler.tuc_ccn_scheduler:
    Hanif: _schedule tuc_ccn_scheduler
2016-11-23 08:11:37.979 27964 INFO nova.scheduler.tuc_ccn_scheduler:
    40 number of instances scheduled with tuc scheduler in
    0.103586912155 seconds
```

Listing C.2: The tuc_ccn_scheduler scheduler based time logs for scheduling different amounts of virtual instances

Requested Number of Instances	Time taken by Filter-Scheduler in secs	Percentage change in time
1	13.69ms	0%
10	46.20ms	237%
20	63.77ms	365%
30	83.91ms	512%
40	103.59ms	656%

Table C.2: cPlex based Scheduler Time Data

Versicherung

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe; die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht.
Bei der Auswahl und Auswertung des Materials sowie bei der Herstellung des Manuskripts habe ich Unterstützungsleistungen von folgenden Personen erhalten:

keine

Weitere Personen waren an der Abfassung der vorliegenden Arbeit nicht beteiligt. Die Hilfe eines Promotionsberaters habe ich nicht in Anspruch genommen. Weitere Personen haben von mir keine geldwerten Leistungen für Arbeiten erhalten, die im Zusammenhang mit dem Inhalt der vorgelegten Dissertation stehen.

Die Arbeit wurde bisher weder im Inland noch im Ausland in gleicher oder ähnlicher Form einer anderen Prüfungsbehörde vorgelegt.

Chemnitz, February 24, 2017

Nishant Ravi