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, March 14, 2017

Autor:	**Nishant Ravi**
Email:	**nisr@hrz.tu-chemnitz.de**
Matrikelnummer:	**355151**
Prüfer:	**Prof. Dr.-Ing. Thomas Bauschert**
Betreuer:	**Dipl.-Ing. Florian Schlegel**
Ausgabedatum:	Feb 21, 2017
Abgabedatum:	March 14, 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 iv

List of Tables v

1 Introduction 1
1.1 Contribution 2
1.2 Results 2
1.3 Research Context[1] 2
1.4 Thesis Structure 3

2 Background 4
2.1 Software Defined Networking [2] 4
2.2 IEEE 802.11 MAC [3] 5
2.3 Hotspot 2.0 [4] 6

3 Software Defined Networking 7
3.1 Existing SDN Controllers 8
3.1.1 Ryu Controller [5] 8
3.1.2 Floodlight Controller [6] 9
3.1.3 OpenDaylight 11
3.2 Applications of SDN 12

4 Installation and Configuration of OpenStack 14
4.1 Minimum hardware requirements for the compute purpose of the cloud 14
4.1.1 Controller 15
4.1.2 Compute 16
4.1.3 Block Storage 16
4.1.4 Object Storage 16
4.2 Networking Options 16
4.3 Configuration of Services and config files 17
4.3.1 Host networking 18
4.3.2 Network Time Protocol (NTP) 19
4.3.3 OpenStack packages 19

4.3.4 Add the Identity service 19
4.3.5 Add the Image service 20
4.3.6 Add the Compute service 21
4.3.7 Add the Networking service 22
4.3.8 Add the dashboard 24
4.3.9 Add the Telemetry service 26

5 Standard Nova Scheduling Algorithm 29
5.1 Theory of Nova Algorithm 29
5.2 Code Trace 31

6 Implementation of Cplex based Nova Scheduler 34
6.1 Mathematical Formulation 34
6.2 The overview of implementation 36

7 Comparision of performance evaluation 42
7.1 Observations of standard Nova scheduler 42
7.2 Observations of cPlex based scheduler 43
7.3 Comparision 44
7.4 Space for further improvements 45

8 Further possible extensions in OpenStack 47
8.1 Open Baton - NFV Orchestrator 49
8.2 Tacker - OpenStack NFV Orchestration 50

9 Conclusion 51

Bibliography 52

Appendix 54

A Configuration Content 56
A.1 RabbitMQ configuration 56

B Logs 57
B.1 FilterScheduler log trace 57
B.2 FilterScheduler log trace for creation of 10 virtual instances 60
B.3 cPlex based Scheduler log trace for creation of 10 virtual instances . 77

C Performance Data 82
C.1 FilterScheduler's Scheduling Time Logs Data 82
C.2 TUC CCN Scheduler's Scheduling Time Logs Data 83

Versicherung **85**

List of Figures

2.1 SDN architecture diagram[7] 5

3.1 RYU SDN Controller Framework [8] 9
3.2 Floodlight Controller architecture [9] 10
3.3 OpenDaylight Architecture Framework [10] 11

4.1 Minimum hardware requirements to set up the OpenStack test bed[11] 15
4.2 Network layout among all the host machines[12] 18
4.3 Login screen for the dashboard 25
4.4 The dashboard screen showing the diagrammatic representation of network topology 26
4.5 An example of the metering with the number of incoming bytes on a network for a virtual machine interface. 28

7.1 Placement decision time for Standard Nova Scheduler from the data C.1 43
7.2 Placement decision time for cPlex based Scheduler from the data C.2 44
7.3 Comparision of placement decision time between Standard Nova FilterScheduler and cPlex based Scheduler 45

8.1 NFV functional overview[13] 48
8.2 Placement of VNFM and VNF over the OpenStack Vitual Machines[14] 49

List of Tables

C.1 FilterScheduler Time Data 83
C.2 cPlex based Scheduler Time Data 84

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.

1.1 Contribution

This thesis provides a novel approach towards separating the data traffic between the different network providers within an access point. This is made possible through the simple, yet effective use of OpenFlow protocol that enables the development of different enterprise WLAN services as applications such as, using software defined network controllers. The performance benefits achieved though this system is possible without any changes to the existing 802.11 client. The proposed system is compatible with the existing enterprise WLAN security protocols like WPA2 enterprise.

1.2 Results

The expected outcome of this thesis is to demonstrate a prototype system that runs an AP on an OpenFLow controlled switch. The AP also provides enterprise grade authentication system WPA2 enterprise alongside a RADIUS server, and host two separate networks. The separation of data traffic between the two networks will take place within the same AP and provide Hotspot 2.0 functionality.

1.3 Research Context[1]

The research described in this thesis was done based on the BIC-IRAP project which is focused on combining the strengths of LTE and Wireless-LAN seamlessly. Through the integration of small and micro cells of LTE with WLAN in the BIC-IRAP system, the two radio technologies are available through a single dynamically configurable hardware configuration.

1.4 Thesis Structure

This thesis report is organized as follows, Chapter 2 describes the background for this thesis. Chapter 3 describes in detail about SDN and the different types in use today. Chapter 4 talks about the control and authentication mechanism such as the protocols and technologies used. Chapter 5 talks about the environment required to build the system such as the tools and software's. Chapter 6 describes in detail how the application is developed, from conceptualization to coding in python. Chapter 7 shows how the system is being implemented. Chapter 8 presents the results obtained from the system after series of testing and enhancement. Chapter 9 concludes the thesis and describes further improvements and drawbacks of this method.

2 Background

This chapter discusses about the topics relevant this thesis, it includes an introduction to software defined networking, 802.11 protocol, Hotspot 2.0 and BIC-IRAP project.

2.1 Software Defined Networking [2]

SDN is nothing but the physical separation of the network control plane from the forwarding plane. The control plane consists of all the logic that the switch requires to correctly setup a forwarding plane, that is, the signaling associated with the switch.

Traditionally, the vendor has the control over the logic necessary for signaling since they run a proprietary firmware. This makes the devices non-interoperable with other vendors which hampers flexibility. Though most of these switches provide SNMP based management solution via CLI, they still do not allow the introduction of custom control plane function or protocol into the switch. This makes experimenting with new protocols cumbersome. Software Defines Networking aims to alleviate these problems by making the switched control plane be easily accessible remotely and be modifiable using the OpenFlow protocol. Any third-party software can than take advantage of this open protocol to manage and orchestrate an entire network.

SDN architecture generally has three components or groups of functionality as shown in the figure 2.1 below.

- **Application Layer:** Consists of programs that communicate the behaviors and needed resources with the SDN controller via the application protocol interfaces (API's). It can also build an abstracted view of the network by collecting information from the controller.
- **Control Layer:** This logical layer functions as a relay that sends the instructions or resources sent by the application layer to the networking components.
- **Infrastructure Layer:** This holds the SDN networking devices that control the forwarding and data processing capabilities of the network including the function to forward and process the data paths.

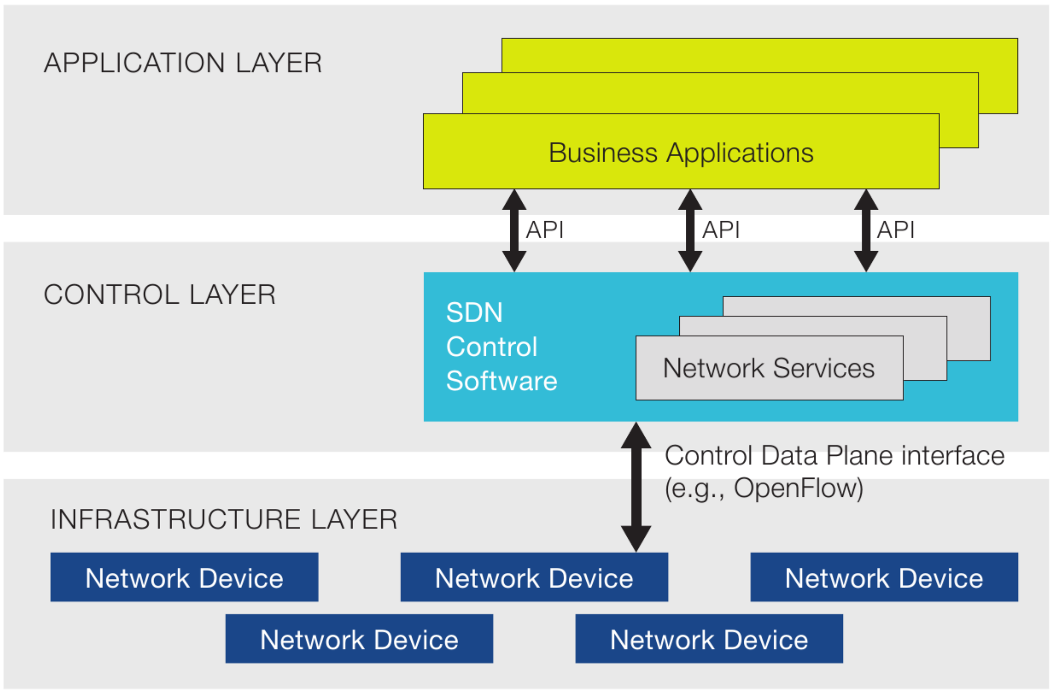


Figure 2.1: SDN architecture diagram[7]

2.2 IEEE 802.11 MAC [3]

The IEEE 802.11 Media Access Control Layer (MAC) [3] defines the protocol for stations to establish connections with each other and transmit data frames. Medium access in 802.11 is performed by a distributed coordination function (DCF), which uses carrier sense multiple access with collision avoidance (CSMA/CA) to enable random medium access among all contending stations (STAs). Hence, it reduces the amount of collisions. Logically, the MAC is divided into two parts, an upper MAC, and a lower MAC. The upper MAC handles management frames, which include probe, authentication, association requests and their corresponding responses. The lower MAC handles control frames, which includes acknowledgement (ACK) frames, along with request-to-send (RTS) and clear-to-send (CTS) frames. The frames handled by the lower MAC have real-time constraints. For instance, ACK frame timeouts are within the order of micro-seconds. For this reason, control frames are handled and generated within hardware. Management frames, however, have softer time constraints, and can be handled in software locally (as is the case in Linux systems that use hostapd [15]), or remotely (as is the case when using a centralized WLAN controller [16]).

An 802.11 based wireless interface can operate under the following operating modes: STA (client), access point (AP), mesh, ad-hoc and item Monitor mode. The most common mode of operation is the infrastructure mode (which includes enterprise WLAN environments). In this mode of operation, clients connect to the AP using a series of message exchanges in a process called “association”. The decision on which AP to associate with is left entirely to the client. Clients learn about APs either passively through beacon frames that are periodically broadcasted by the access points, or actively by performing a probe scan.

In a probe scan, clients first send out probe request frames over all channels. APs that receive these frames and are willing to accept a connection from a client respond with a probe response frame. All APs from which the client receives probe responses are candidates for the client to associate with. Next, the client sends an authentication frame, and waits for an authentication response from the AP. This is followed by the client sending an association request, and receiving an association response from the AP. If the network is operating in open authentication mode, the client is considered to be associated at this point, and can now transmit data frames to be forwarded by the AP. If the AP is configured to use WPA, WPA2, or WPA2 Enterprise, the corresponding 802.1X [15] handshake is performed after the association phase before clients can forward data frames through the AP.

2.3 Hotspot 2.0 [4]

It is a new wireless network standard that is designed to make connection to public Wi-Fi hotspots more easy and secure. They are already supported on many mobile devices running some of the popular operating systems such as Windows 10, Mac OS 10.9 or newer, Android 6.0 or newer, and iOS 7 or newer.

The main purpose of Hotspot 2.0 is to provide seamless mobility like cellular style “roaming” for Wi-Fi networks. The device will automatically connect to the available networks based on the networks partners on the home networks while roaming globally. This is made possible using the latest 802.11u [17] protocol designed for the same purpose. Some organizations also call this as Passpoint [18].

3 Software Defined Networking

The Open Network foundation, a non-profit organization, has been undertaking an extensive research for the past couple of years in designing and standardizing open network components such as OpenFlow, SDN etc. which, after being rolled out on to a variety of network devices and software's from different vendors has been delivering substantial benefits to both enterprises and carriers include: [19]

- **Directly Programable:**Network directly programmable because the control functions are decoupled from forwarding functions, which enable the network to be programmatically configured by proprietary or open source automation tools.
- **Centralized Management:** Network intelligence is logically centralized in SDN controller software that maintains a global view of the network, which appears to applications and policy engines as a single, logical switch.
- **Reduce CapEx:** Software Defined Networking potentially limits the need to purchase purpose-built, ASIC-based networking hardware, and instead supports pay-as-you-grow models
- **Reduce OpEX:** SDN enables algorithmic control of the network of network elements (such as hardware or software switches/routers that are increasingly programmable, making it easier to design, deploy, manage, and scale networks. The ability to automate provisioning and orchestration optimizes service availability and reliability by reducing overall management time and the chance for human error.
- **Deliver Agility and Flexibility:** Software Defined Networking helps organizations rapidly deploy new applications, services, and infrastructure to quickly meet changing business goals and objectives.
- **Enable Innovation:** SDN enables organizations to create new types of applications, services, and business models that can offer new revenue streams and more value from the network.

3.1 Existing SDN Controllers

For this Master thesis, a few available SDN controllers are first studied for its functionality that can be manipulated for data path segregation. A brief overview on each controller is discussed in the following sections.

3.1.1 Ryu Controller [5]

Ryu is a component-based software defined networking framework. It provides software components with well-defined API that make it easy to create new network management and control applications. The component that is of particular interest for this master thesis is the switching hub by using OpenFlow.

Switching hubs have a variety of functions, some of which are discussed below.

- Learns the MAC address of the host connected to a port and retains it in the MAC address table.
- When receiving, packets addressed to a host already learned, transfers them to the port connected to the host.
- When receiving, packets addressed to an unknown host, performs flooding.

The main reason to choose RYU over other controllers is due to its customizability and easy to create core applications using Python. RYU allows users to modify core functions or use these functions to create custom applications that suits specific needs, in this case, it was used to create a switching application that can segregate users within the OpenVswitch, instead of being controlled each time by the controller.

The software components provided by RYU with well-defined Application Programming Interface (API's), makes it easy for developers to create custom network management or control applications. The existing components can be quickly and easily modified or implement a custom component so that the underlying network can meet the changing demands of the application. RYU is designed to increase the agility of network by being more easily manageable and adapt how traffic is handled.

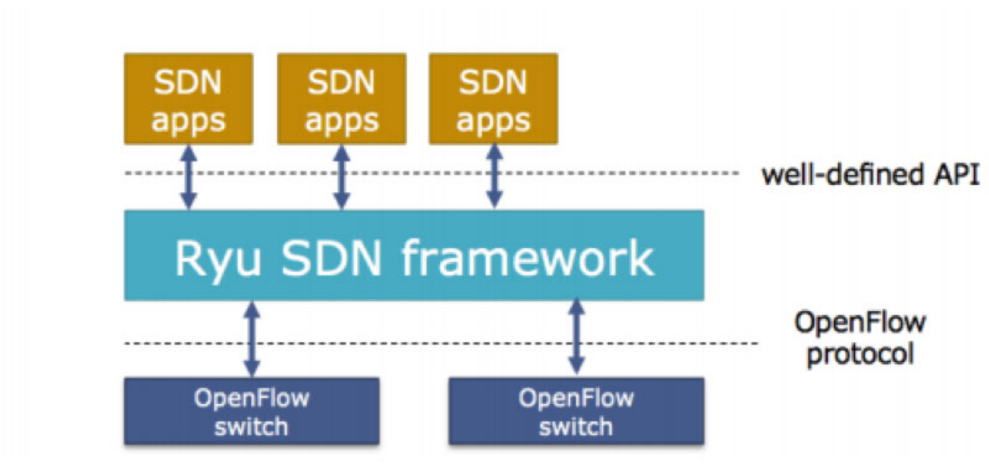


Figure 3.1: RYU SDN Controller Framework [8]

RYU Controller is supported by NTT of Japan and has a strong open source RYU community that maintain and manage the code which is hosted at GitHub. OpenStack also supports deployment of RYU as network controller in its cloud operating systems.

3.1.2 Floodlight Controller [6]

It is yet another open source SDN controller similar to RYU. The benefit of using this controller is the ability to easily develop applications using Java, which is widely used for high level programming by developers and to adapt the software as per requirement. Flood Light offers Representational state transfer application program interface (REST API's) which help developers to easily program interfaces with the product.

Floodlight is used to run as the network backend for OpenStack. When used with the Neutron plugin with OpenStack, the Floodlight controller functions as a network-as-a-service model with the help of REST API offered by Floodlight. The following diagram below shows the architecture of Floodlight controller.

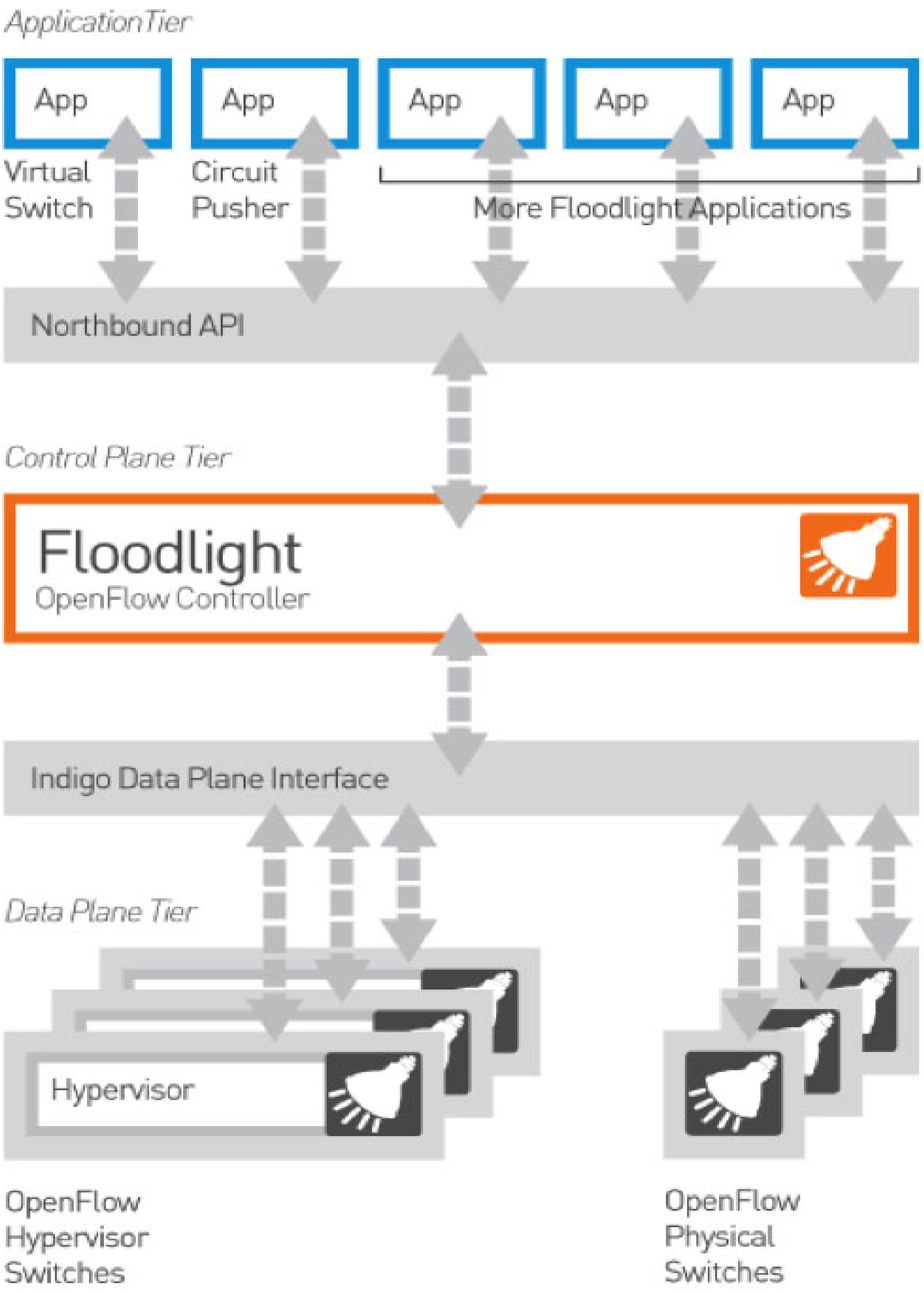


Figure 3.2: Floodlight Controller architecture [9]

3.1.3 OpenDaylight

OpenDaylight controller is based on JVM, similar to Floodlight, which was a derivative of OpenDaylight that can be deployed on any systems that supports Java. OpenDaylight controller uses the following tools as its framework:

- **Maven:** OpenDaylight uses Maven, which uses Project Object Model to script the dependencies between the bundles for easier build automation.
- **OSGi:** It works as the back-end for OpenDaylight as it loads bundles dynamically and packages JAR files and binding them together for exchange of information.
- **JAVA interfaces:** They are used for event listening, specifications, and forming patterns.
- **REST APIs:** These are the northbound APIs that manage the topology, flow program, host tracking, static routing and so on.

The following figure shows the framework of OpenDaylight with the above tools mentioned:

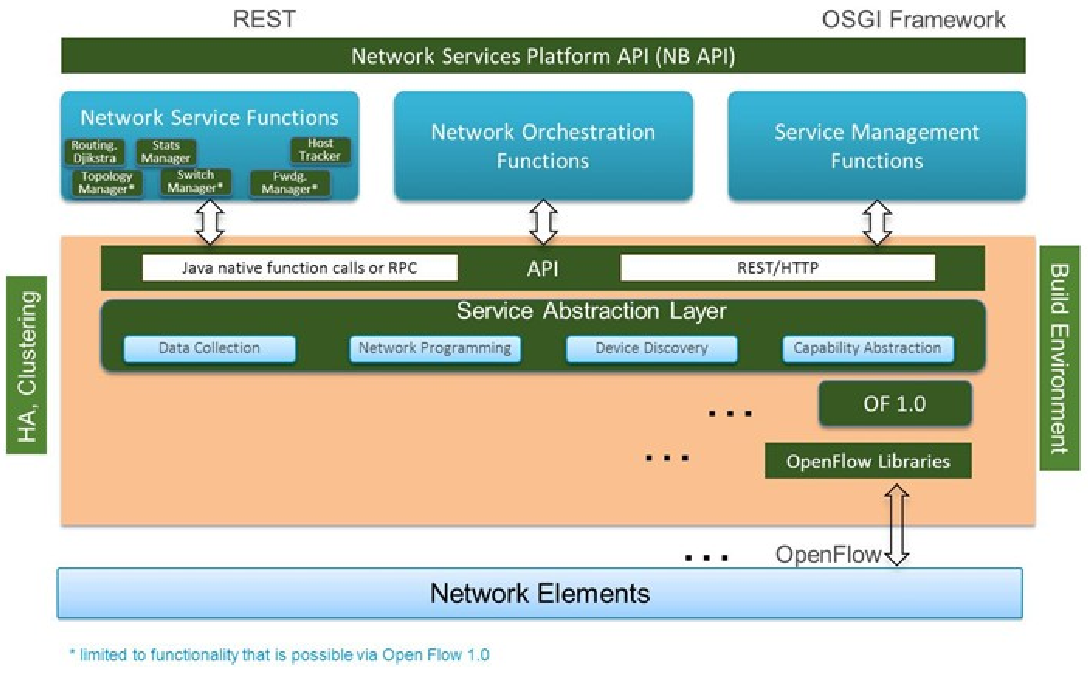


Figure 3.3: OpenDaylight Architecture Framework [10]

3.2 Applications of SDN

Many research efforts has been done till now in writing SDN applications. Jose et. al. [20] propose using commodity OpenFlow enabled switches for traffic measurement. The authors propose a framework where a collection of rules are installed on OpenFlow switches, and having a controller track the corresponding flow match counters. The controller can then draw inferences from the counters and dynamically tune the rules as required in order to identify different traffic aggregates.

Resonance [21] is another application uses programmable switches to enforce access control in the network. The authors try to prove that today's enterprise networks rely on different combinations of middle boxes, intrusion detection systems, and network configurations in order to enforce access control policies, whilst placing a burden on end-hosts in the system to remain patched and secure. The proposed system uses an SDN approach comprising programmable switches and a controller, which together implement a network monitoring framework, a policy specification framework, and the ability to trigger specific actions at the switch level.

OpenSAFE [22] is a framework that enables network monitoring using OpenFlow. It addresses the problem of routing traffic for network analysis in a reliable manner without affecting normal traffic.

Hedera [23] is an adaptive flow scheduling system for data center networks. The premise for Hedera is that existing IP multipathing techniques used in data centers usually rely on per-flow static hashing, which can lead to under-utilisation of some network paths over time due to hash collisions. The system works by detecting large flows at the edge switches of a data center, and using placement algorithms to find good paths for the flows in the network. Experiments performed using simulations indicate significant improvements over static load balancing techniques.

In the paper *OpenFlow based server load balancing gone wild* [24], the authors address the problem of server load balancing using OpenFlow switches. The number of flow entries that can be saved on an OpenFlow switch is much less than the number of unique flows that a switch might need to handle in data center workloads. Thus, micro flow management using per-flow rules is not practical for performing distributing flows between different servers using a switch. The authors thus take advantage of OpenFlow's wildcard based rules capability, and propose algorithms to compute concise wildcard rules that achieve a specific distribution of traffic.

These are some of the applications that have been written for SDN controllers but none of them address the challenge to dynamically redirect packets in real time, based on different clients and their credentials used for authentication. This thesis proposed to build one such application that can segregate packets coming from different clients in such a way that there is no possible connection between multiple clients associated within the same access point.

4 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.

4.1 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 4.1 gives a general idea about the required nodes and optional nodes to start with an OpenStack for compute.

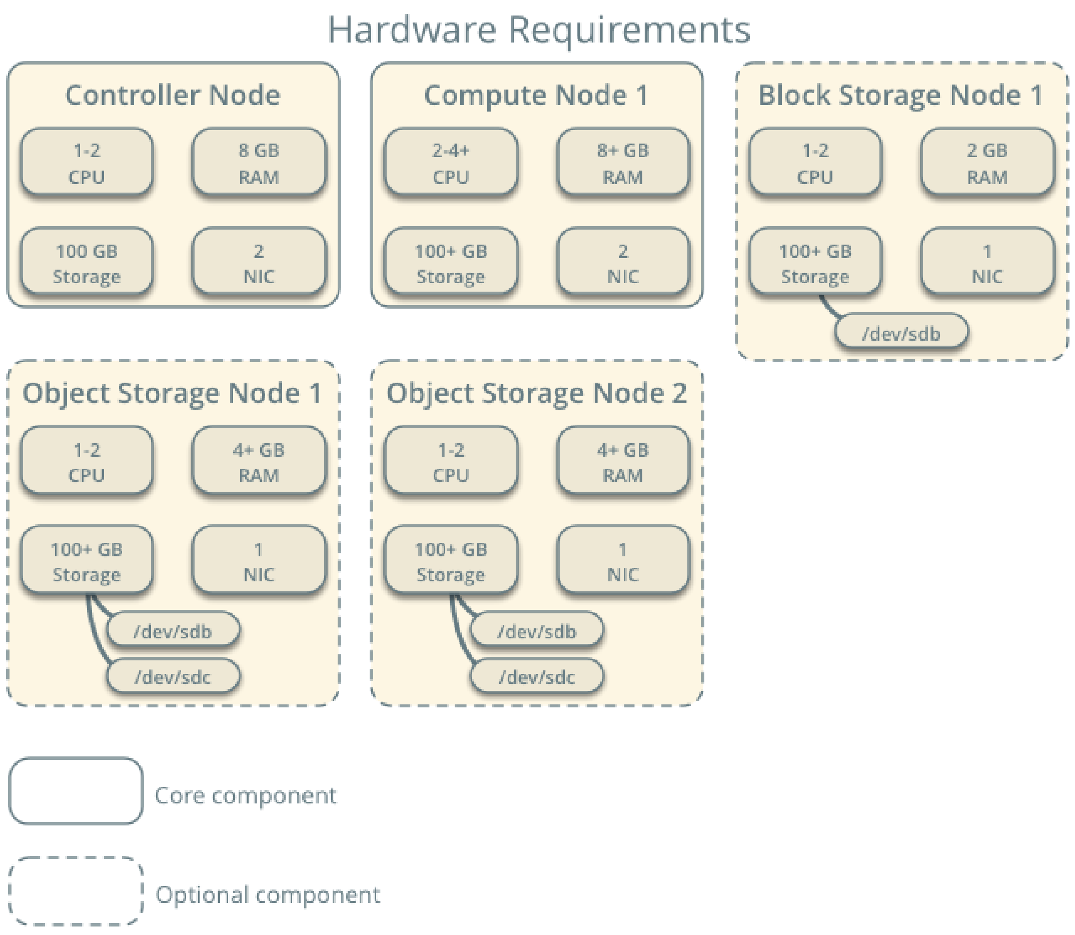


Figure 4.1: Minimum hardware requirements to set up the OpenStack test bed[11]

4.1.1 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.

4.1.2 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.

4.1.3 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.

4.1.4 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.

4.2 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.

4.3 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`.

4.3.1 Host networking

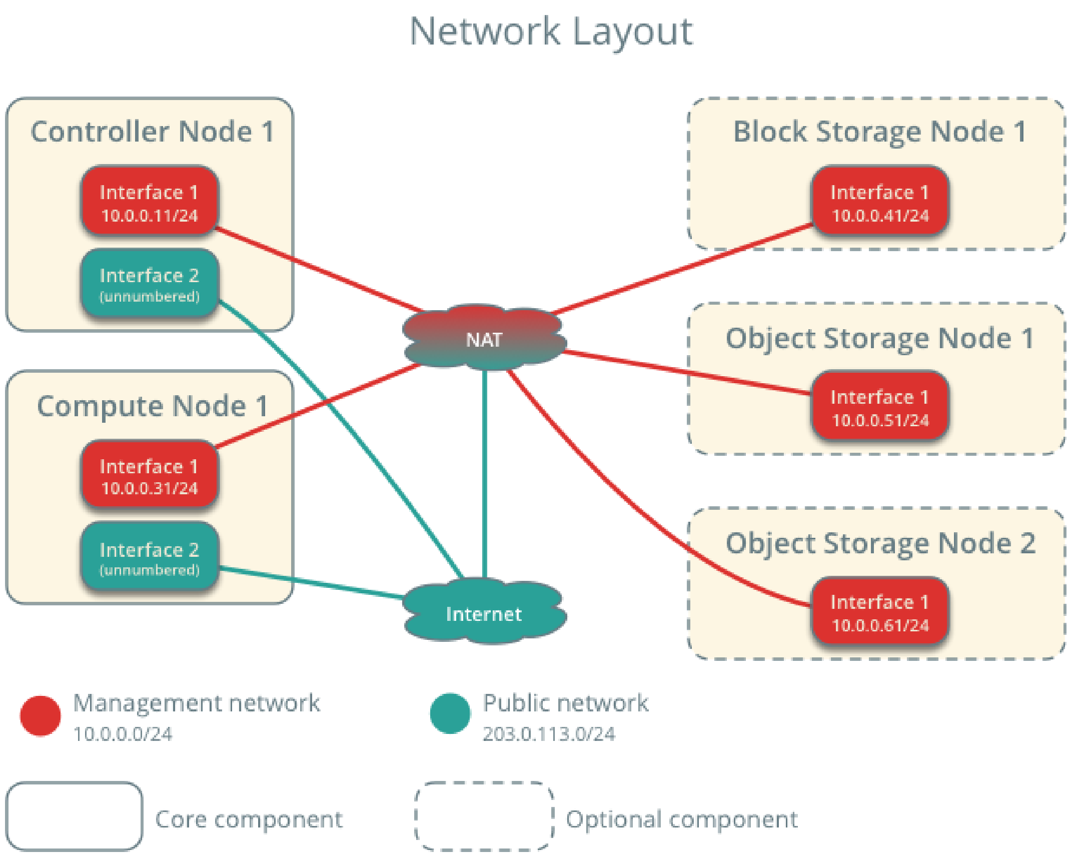


Figure 4.2: Network layout among all the host machines[12]

The figure 4.2 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.

4.3.2 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.

4.3.3 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.

4.3.4 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).

4.3.5 Add the Image service

The OpenStack Image service is central to Infrastructure-as-a-Service (IaaS) as shown in figure ??. 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.

4.3.6 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.

4.3.7 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.

4.3.8 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 4.3 shows the login screen for the OpenStack dashboard.

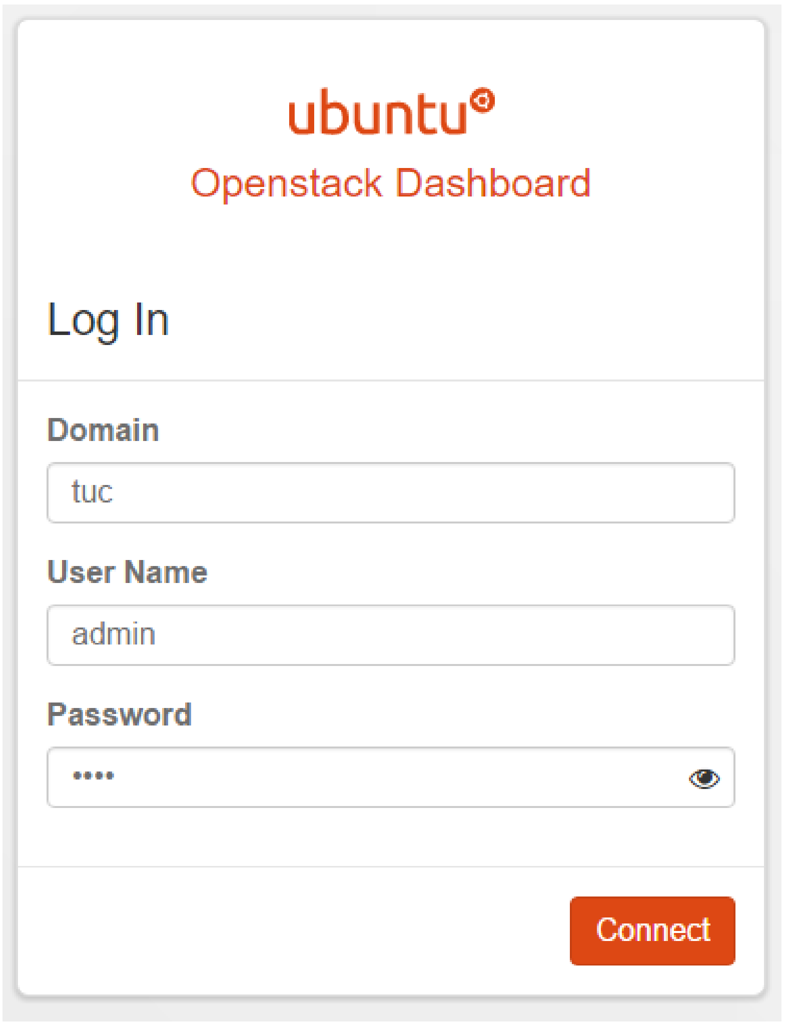


Figure 4.3: Login screen for the dashboard

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

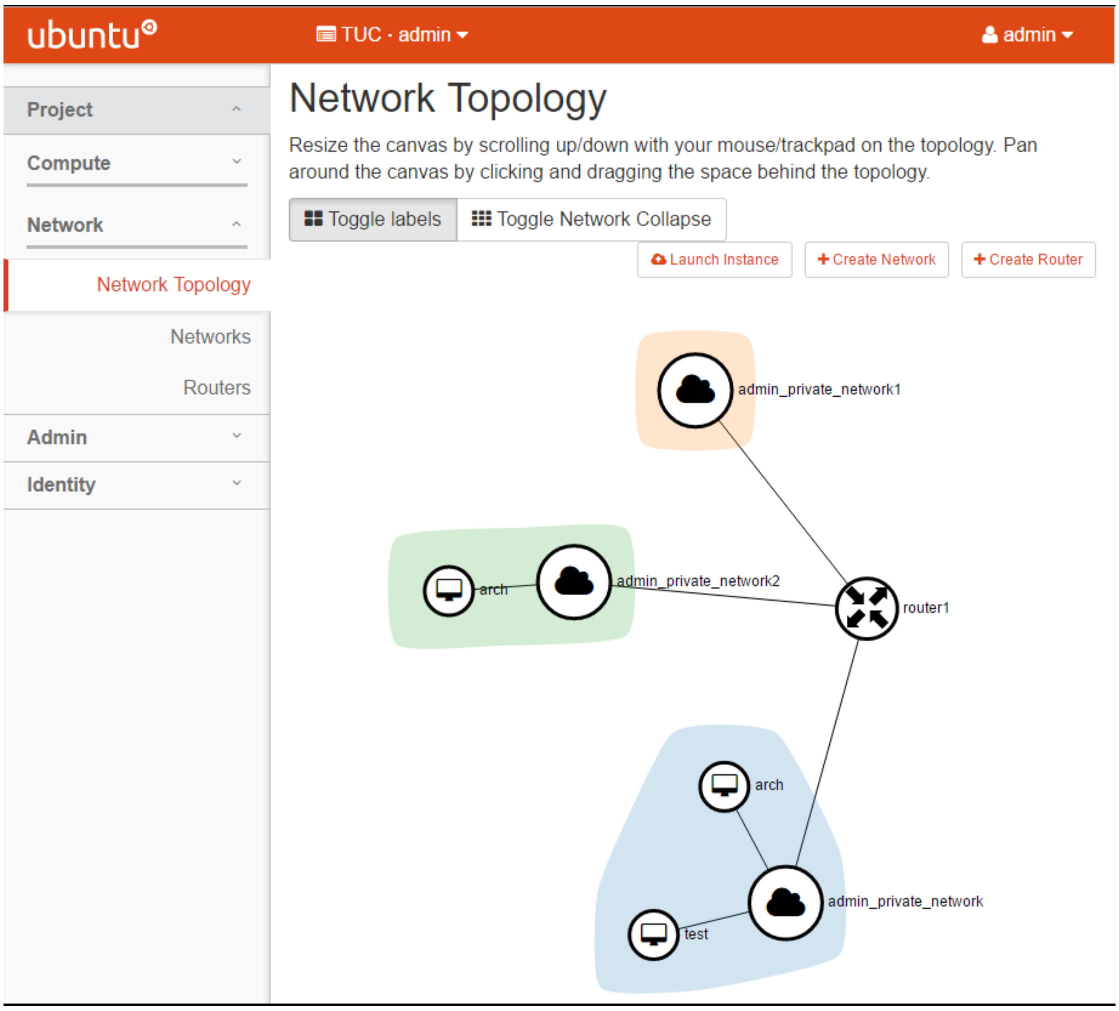


Figure 4.4: The dashboard screen showing the diagrammatic representation of network topology

4.3.9 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
```

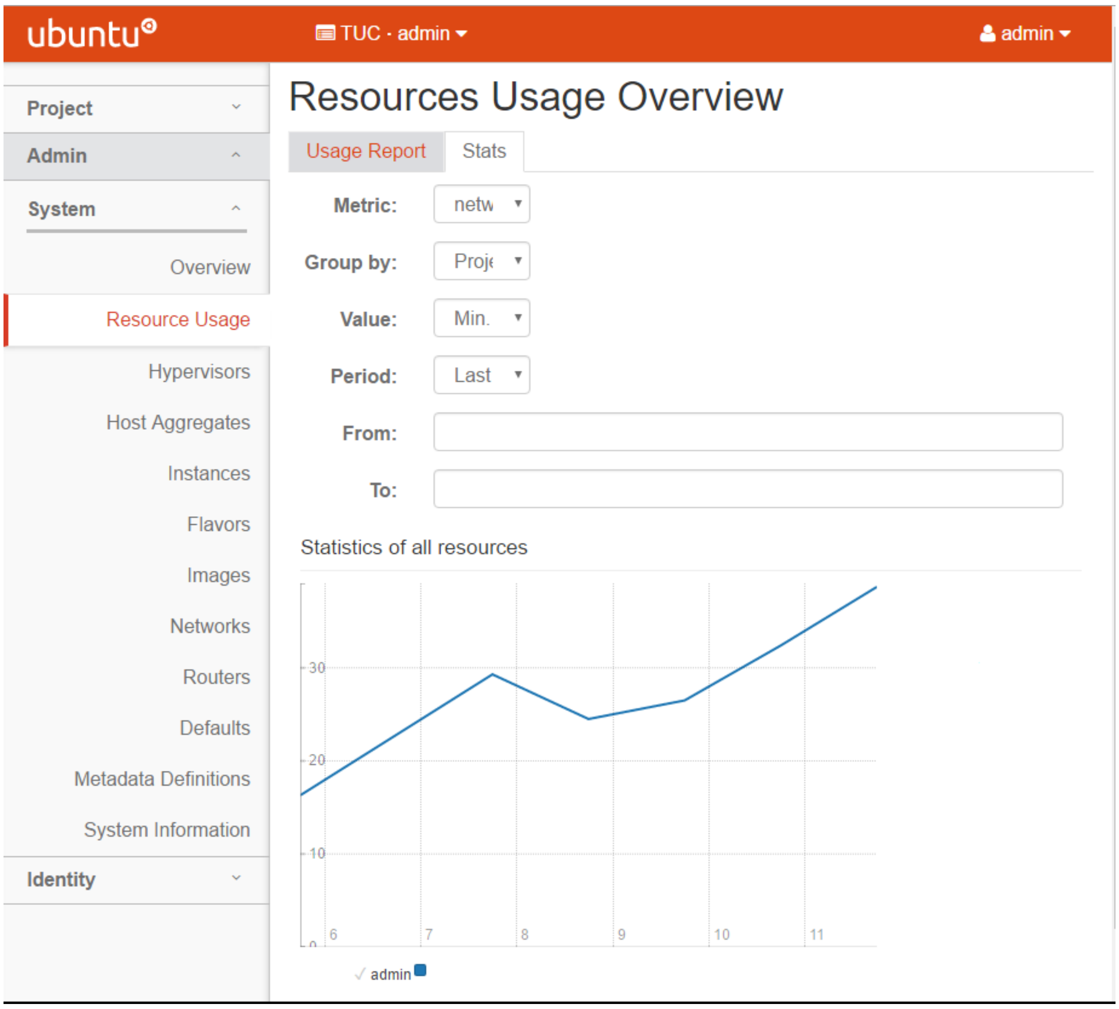
Once the Telemetry service has been installed on controller, it is configured to be consumed by Image service by editing the existing /etc/glance/glance-api.conf and /etc/glance/glance-registry.conf configuration files by adding the [oslo_messaging_rabbit].
Please refer to Appendix RabbitMQ configuration 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 4.5: 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.

5 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.

5.1 Theory of Nova Algorithm

As explained in ??, 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.

5.2 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 5.1: The function select_destination

In the above code listing 5.1, 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 5.2.

```
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 5.2: The function _schedule

In the listing 5.2, 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 5.3.

```
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 5.3: The function get_filtered_hosts

In this listing 5.3, 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 5.2, 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 5.4 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 ??

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

Listing 5.4: 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.

6 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.

6.1 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:

$$\min \sum_{n_s \in N_s} x_{n_s} \tag{6.1}$$

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:

$$\sum_{n_s \in N_s} x_{n_s}^{n_v} suit_{n_s}^{n_v} = 1, \forall n_v \in N_v \tag{6.2}$$

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

$$x_{n_s}^{n_v} \leq suit_{n_s}^{n_v}, \forall n_s \in N_s, n_v \in N_v \tag{6.3}$$

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:

$$x_{n_s} \geq x_{n_s}^{n_v}, \forall n_s \in N_s, n_v \in N_v \tag{6.4}$$

The RAM capacity constraints are given as follows:

$$\sum_{n_v \in N_v} x_{ns}^{n_v} d_{n_v}^{RAM} \leq c_{n_s}^{RAM}, \forall n_s \in N_s \tag{6.5}$$

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

$$\sum_{n_v \in N_v} x_{ns}^{n_v} d_{n_v}^{HDD} \leq c_{n_s}^{HDD}, \forall n_s \in N_s \tag{6.6}$$

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

6.2 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[25]. 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

        chosen_host = weighed_hosts[0]
        LOG.debug("Selected host: %(host)s", {'host': chosen_host})

        # append the selected hosts  to the array mapping
        selected_hosts.append(chosen_host)

        # Now consume the resources so the filter/weights
        # will change for the next instance.
        chosen_host.obj.consume_from_instance(instance_properties)
        if update_group_hosts is True:
            # NOTE(sbauza): Group details are serialized into a list now
            # that they are populated by the conductor, we need to
            # deserialize them
            if isinstance(filter_properties['group_hosts'], list):
                filter_properties['group_hosts'] = set(
                    filter_properties['group_hosts'])
            filter_properties['group_hosts'].add(chosen_host.obj.host)
    LOG.info('%s number of instances scheduled with tuc scheduler in %s seconds' % (num_instances, (rtime.time() - start_time)))
    return selected_hosts
```

Listing 6.1: The cPlex based TUC_scheduler's _schedule function

In the above code listing 6.1, 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 6.2.

```
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 = []
```

```python
            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 6.2: The cPlex based TUC_scheduler's solve_TUC_Cplex function

The above code listing 6.2 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.

7 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.

7.1 Observations of standard Nova scheduler

Observing the listing 5.2, 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.

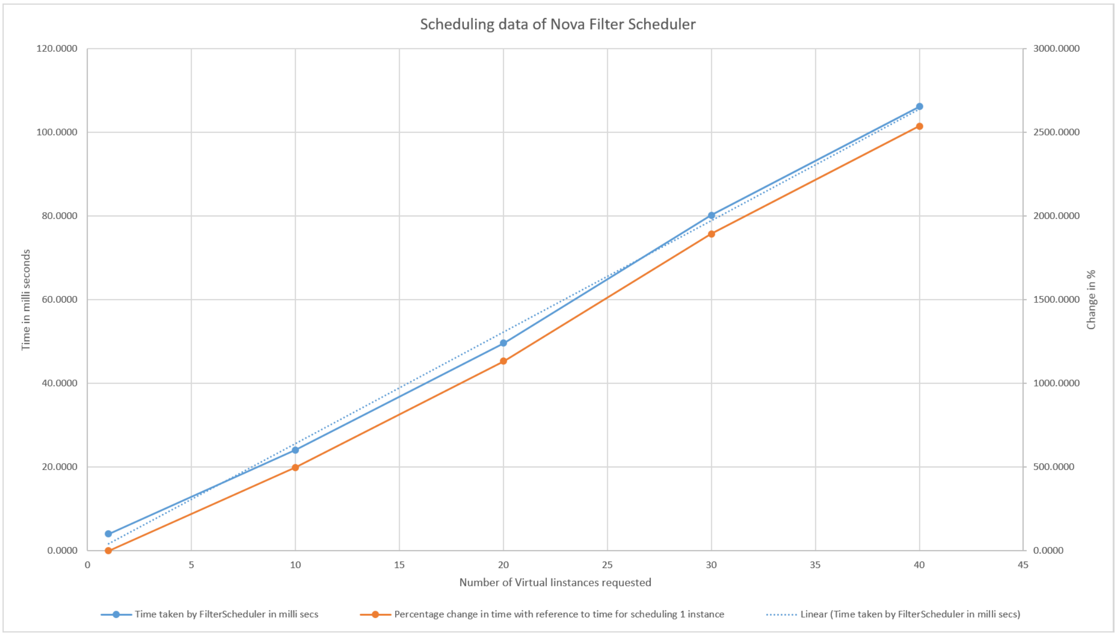


Figure 7.1: Placement decision time for Standard Nova Scheduler from the data C.1

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.

7.2 Observations of cPlex based scheduler

Observing the listing 6.1, 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.

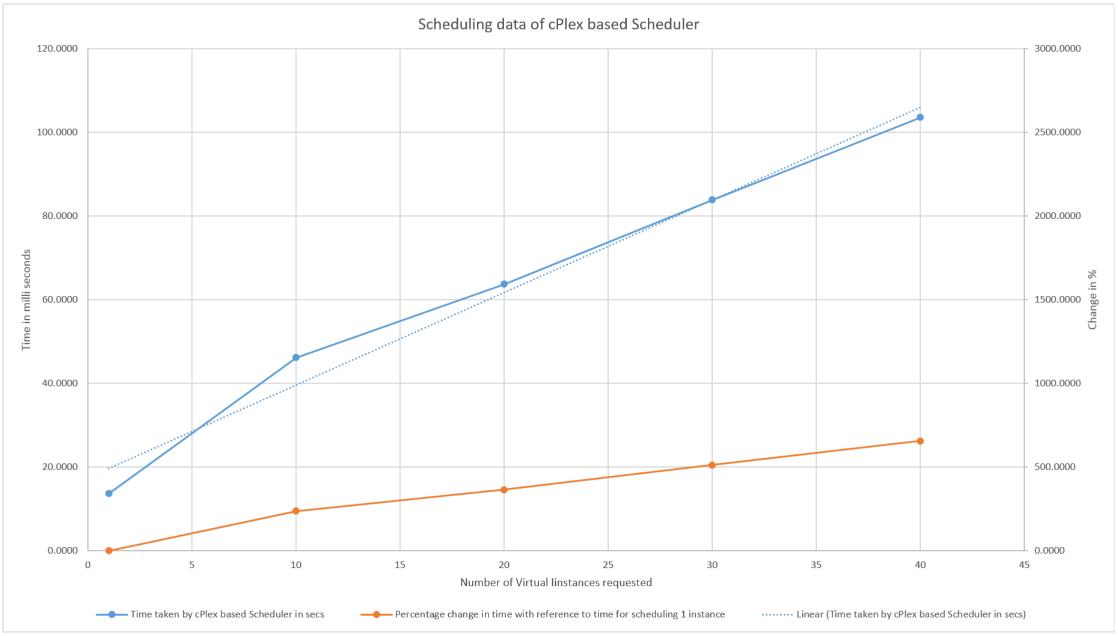


Figure 7.2: Placement decision time for cPlex based Scheduler from the data C.2

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.

7.3 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 7.1 and 7.2, 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.

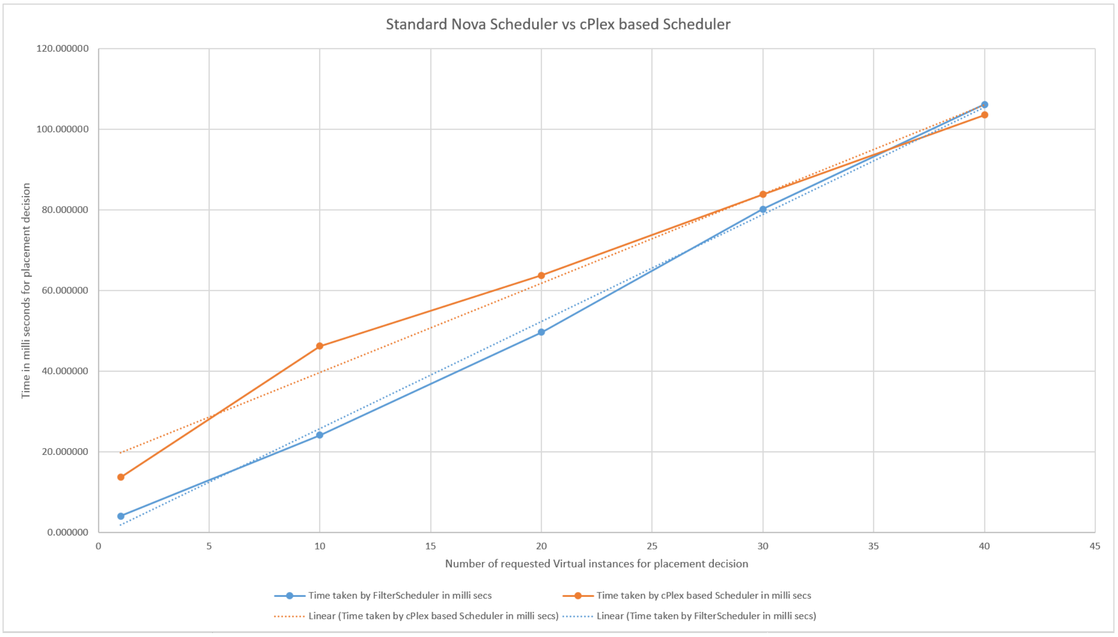


Figure 7.3: Comparision of placement decision time between Standard Nova FilterScheduler and cPlex based Scheduler

From the above chart 7.3, 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.

7.4 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.

8 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.[26]

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.

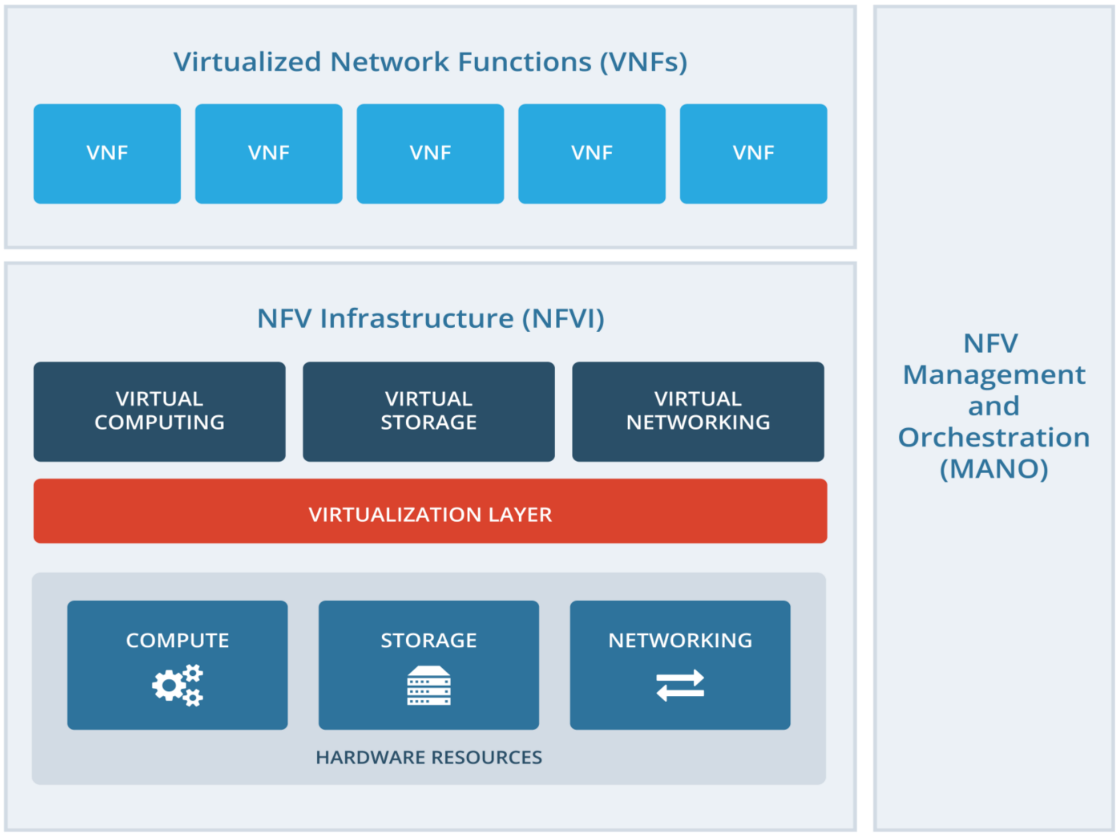


Figure 8.1: NFV functional overview[13]

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

8.1 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.

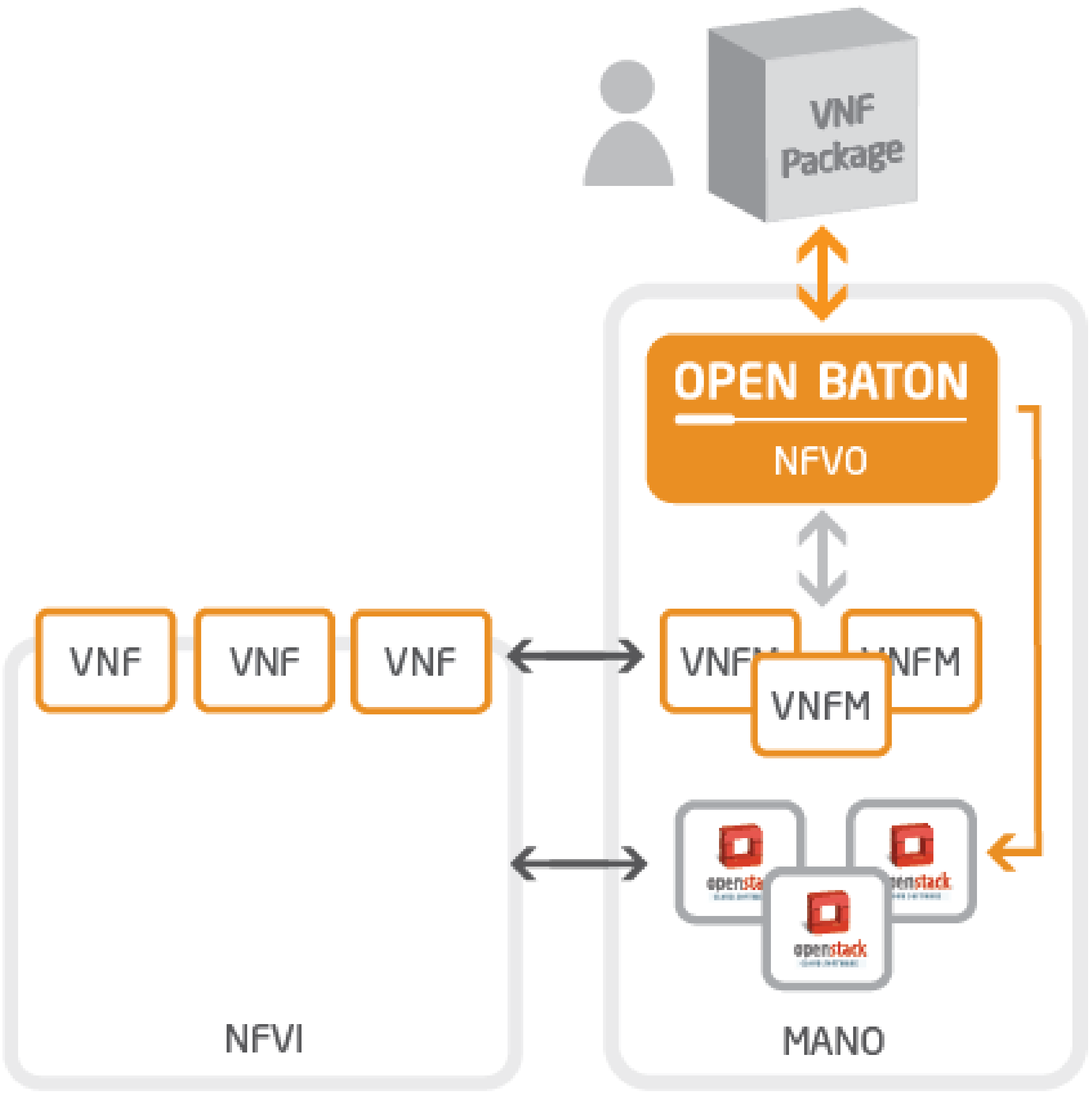


Figure 8.2: Placement of VNFM and VNF over the OpenStack Vitual Machines[14]

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.

8.2 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.

9 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] What is BIC-IRAP? URL http://www.bic-irap.de/index.php/en. Online; accessed 01-March-2017.

[2] What is SDN?, . URL https://www.opennetworking.org/sdn-resources/sdn-definition. Online; accessed 01-March-2017.

[3] IEEE Standards Association et al. 802.11-2012-ieee standard for information technology–telecommunications and information exchange between systems local and metropolitan area networks–specific requirements part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications. *IEEE Std*, 802, 2012.

[4] Understanding wi-fi hotspot 2.0 and how to leverage it for your business, by jason guest. http://hotelexecutive.com/business_review/3674/understanding-wi-fi-hotspot-20-and-how-to-leverage-it-for-your-business. (Accessed on 03/02/2017).

[5] Switching hub — ryubook 1.0 documentation. URL https://osrg.github.io/ryu-book/en/html/switching_hub.html. (Accessed on 03/06/2017).

[6] What is a floodlight controller? - defined. https://www.sdxcentral.com/sdn/definitions/sdn-controllers/open-source-sdn-controllers/what-is-floodlight-controller/, . (Accessed on 03/08/2017).

[7] Sdn architecture diagram, . URL https://www.sdxcentral.com/wp-content/uploads/2015/03/sdn-architecture.png. Online; accessed 02-March-2017.

[8] Ryu-controller-sdn-framework.jpg (jpeg image, 900 $\times$ 495 pixels). URL https://www.sdxcentral.com/wp-content/uploads/2014/09/ryu-controller-sdn-framework.jpg. (Accessed on 03/08/2017).

[9] Floodlight architecture diagram. https://www.sdxcentral.com/wp-content/uploads/2014/09/floodlight-open-sdn-controller-diagram.jpg, . (Accessed on 03/08/2017).

[10] Architectural_framework.jpg (717$\times$435). https://wiki.opendaylight.org/images/b/b1/Architectural_Framework.jpg, . (Accessed on 03/14/2017).

[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] hostapd: Ieee 802.11 ap, ieee 802.1x/wpa/wpa2/eap/radius authenticator. http://w1.fi/hostapd/. (Accessed on 03/02/2017).

[16] Rfc 5412 - lightweight access point protocol. https://tools.ietf.org/html/rfc5412. (Accessed on 03/02/2017).

[17] Ieee xplore full-text pdf:. http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=5721908. (Accessed on 03/02/2017).

[18] Wi-fi certified passpoint | wi-fi alliance. http://www.wi-fi.org/discover-wi-fi/wi-fi-certified-passpoint. (Accessed on 03/02/2017).

[19] What's software-defined networking (sdn)? https://www.sdxcentral.com/sdn/definitions/what-the-definition-of-software-defined-networking-sdn/. (Accessed on 03/06/2017).

[20] Lavanya Jose, Minlan Yu, and Jennifer Rexford. Online measurement of large traffic aggregates on commodity switches. http://static.usenix.org/events/hotice11/tech/full_papers/Jose.pdf. (Accessed on 03/14/2017).

[21] Ankur Kumar Nayak, Alex Reimers, Nick Feamster, and Russ Clark. Resonance: Dynamic access control for enterprise networks. In *Proceedings of the 1st ACM Workshop on Research on Enterprise Networking*, WREN '09, pages 11–18, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-443-0. doi: 10.1145/1592681.1592684. URL http://doi.acm.org/10.1145/1592681.1592684.

[22] Jeffrey R Ballard, Ian Rae, and Aditya Akella. Extensible and scalable network monitoring using opensafe. In *INM/WREN*, 2010.

[23] Mohammad Al-Fares, Sivasankar Radhakrishnan, Barath Raghavan, Nelson Huang, and Amin Vahdat. Hedera: Dynamic flow scheduling for data center networks. In *NSDI*, volume 10, pages 19–19, 2010.

[24] Richard Wang, Dana Butnariu, Jennifer Rexford, et al. Openflow-based server load balancing gone wild. *Hot-ICE*, 11:12–12, 2011.

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

[26] 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 12b8f78400724c8fa549aec66ad1a43a 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 12b8f78400724c8fa549aec66ad1a43a 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 12b8f78400724c8fa549aec66ad1a43a 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 12b8f78400724c8fa549aec66ad1a43a 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 12b8f78400724c8fa549aec66ad1a43a 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 12b8f78400724c8fa549aec66ad1a43a 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 12b8f78400724c8fa549aec66ad1a43a 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 12b8f78400724c8fa549aec66ad1a43a 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 12b8f78400724c8fa549aec66ad1a43a 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 12b8f78400724c8fa549aec66ad1a43a 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 12b8f78400724c8fa549aec66ad1a43a 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 12b8f78400724c8fa549aec66ad1a43a 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, March 14, 2017

Nishant Ravi