Datacenter Management as a Service (DMaaS)

Virtual Abstraction Interface

Functional Specification

Version 1.1

By

Debasis Dash

debasis.dash@sjsu.edu

Submitted to

Dr. Youngee Park

Table of Contents

Table of Figures

List of Tables

Version History

Version	Date	Changes
1.0	10/27/2015	First draft presenting the required components, relationship, functionality and approach to test for functional specification document for virtual abstraction interface of DMaaS service.
1.1	11/10/2015	Added the sequence call flow for Virtual Abstraction Interface, additional test cases, conclusion and reference to the work book 2

Introduction

Datacenter Management as a Service (DMaaS) provides a new approach for datacenter management [7]. It provides complete functionality for virtual machine creation, deletion, performance monitoring, predictive recommendation on resource usage, role based access and security in a multi-tenant model.

One of the major component of DMaaS is Virtual Abstraction Interface (**VAI**) module. This module provides abstraction to the **virtual environment specific implementation** for the application layer. This will provides a single presentation interface for disparate virtual environment. In addition to the functionality like creation and deletion of the virtual machine **VAI will provide the resource creation and extending TCP/IP layer 2 functionality using VxLAN overlay networking protocol in remote datacenters using Open vSwitch** [2]. This will give **unique features** to create large number of virtual machines spreading across various datacenters with disparate virtual environment and present as in a single datacenter.

VAI will provides the extensibility to the DMaaS project by hiding the complexity of the hypervisor based implementation from the DMaaS server module. The current requirements are to support OpenStack [1] and VMware [4] based cloud implementation. This report describes the inter-modules communications.

This document describes the major components (stats collector, snapshot manager, request validator, recovery agent, network adaptor, alarm module, logger, policy manager and rabbit message queue) required to develop this module and their relationship.

References

[1] OpenStack benefit and concept from http://www.openstack.org/

Understanding, deployment related information are retrieved from the OpenStack official webpage. The project support to control OpenStack based datacenter environment

[2] Concept of open vSwitch from http://openvswitch.org/

Open vSwitch (OVS) is open source multi-layer virtual switch. The installation and usage will be followed from this page. OVS will be used to extend the layer 2 functionality across datacenters. VM can be created in different datacenter and connected using VxLAN over OVS.

[3] OpenStack API reference from http://developer.openstack.org/api-ref.html

The required OpenStack specific module will use the nova APIs for creating and deleting virtual machine

[4] VMware concept, usage and advantage from
https://www.vmware.com/support/developer/vc-sdk/visdk41pubs/ApiReference/
The required VMWare specific module will use the vSphere APIs for creating and deleting virtual machine

[5] Distributed open message queue from https://www.rabbitmq.com/

RabbitMQ is open source messaging application. This will used for inter-process communication in the distributed architecture.

[6] OpenStack architecture from http://searchtelecom.techtarget.com/tip/Is-OpenStack-IPv6-ready

OpenStack architecture is explained with respect to the IPV6 readiness.

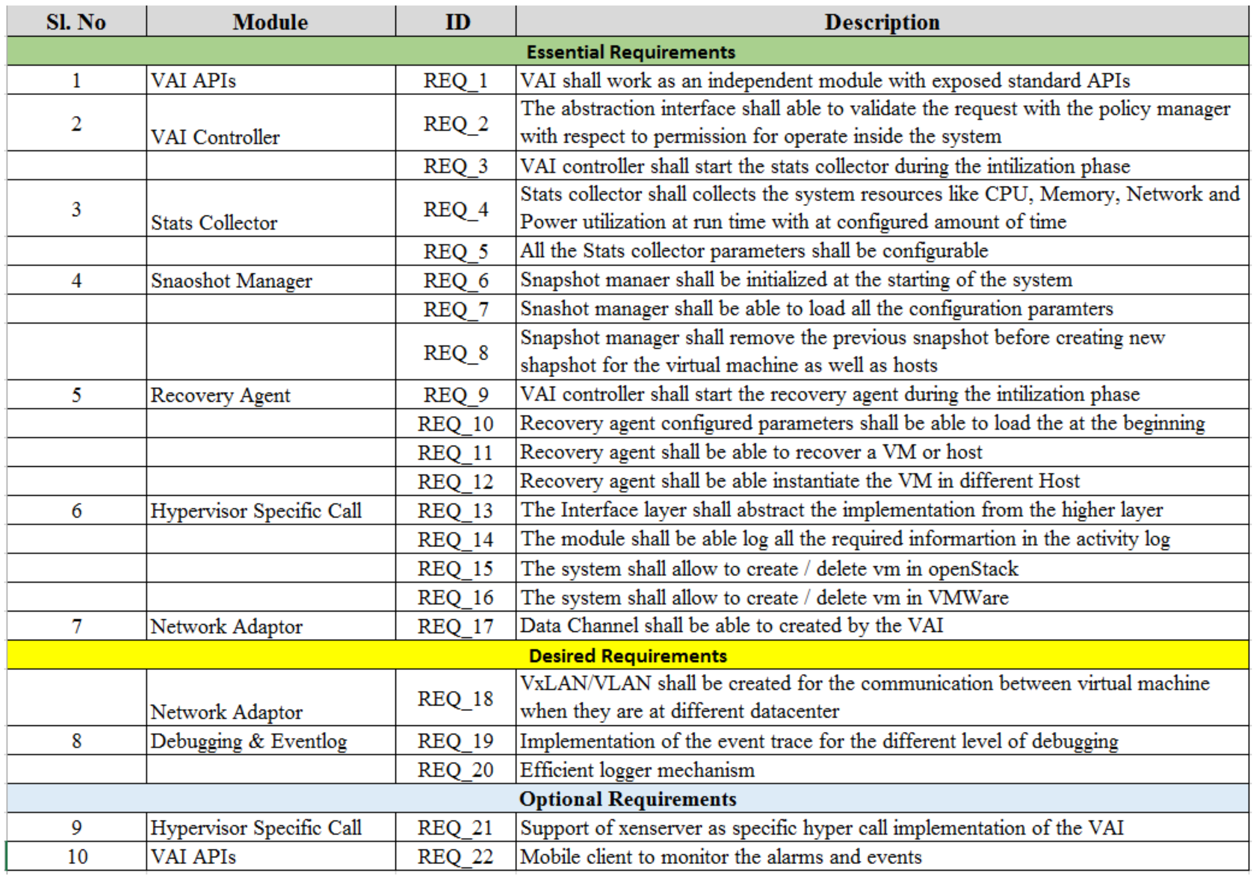
[7] Chaubal, T., Dash, D., Dash, J. & Patil, S. "**Work Book 2 of DMaaS for CMPE 295A**"

Work Book 2 describe functional overview, implementation, project schedule tracking and test cases for Datacenter Management as a Service in details.

Requirements

High level functional requirements are listed in the table-1. Requirements are arranged based on the level of necessity as mentioned in the IEEE standard.

Table -1 is arranged with Sl. No, Module (Module name to complete VAI functionality), ID (Requirement ID) and Description. Each requirement categorized one of Essential / Desired / Optional requirements.



Sl. No	Module	ID	Description
Essential Requirements			
1	VAI APIs	REQ_1	VAI shall work as an independent module with exposed standard APIs
2	VAI Controller	REQ_2	The abstraction interface shall able to validate the request with the policy manager with respect to permission for operate inside the system
		REQ_3	VAI controller shall start the stats collector during the intilization phase
3	Stats Collector	REQ_4	Stats collector shall collects the system resources like CPU, Memory, Network and Power utilization at run time with at configured amount of time
		REQ_5	All the Stats collector parameters shall be configurable
4	Snaoshot Manager	REQ_6	Snapshot manaer shall be initialized at the starting of the system
		REQ_7	Snashot manager shall be able to load all the configuration paramters
		REQ_8	Snapshot manager shall remove the previous snapshot before creating new shapshot for the virtual machine as well as hosts
5	Recovery Agent	REQ_9	VAI controller shall start the recovery agent during the intilization phase
		REQ_10	Recovery agent configured parameters shall be able to load the at the beginning
		REQ_11	Recovery agent shall be able to recover a VM or host
		REQ_12	Recovery agent shall be able instantiate the VM in different Host
6	Hypervisor Specific Call	REQ_13	The Interface layer shall abstract the implementation from the higher layer
		REQ_14	The module shall be able log all the required informartion in the activity log
		REQ_15	The system shall allow to create / delete vm in openStack
		REQ_16	The system shall allow to create / delete vm in VMWare
7	Network Adaptor	REQ_17	Data Channel shall be able to created by the VAI
Desired Requirements			
	Network Adaptor	REQ_18	VxLAN/VLAN shall be created for the communication between virtual machine when they are at different datacenter
8	Debugging & Eventlog	REQ_19	Implementation of the event trace for the different level of debugging
		REQ_20	Efficient logger mechanism
Optional Requirements			
9	Hypervisor Specific Call	REQ_21	Support of xenserver as specific hyper call implementation of the VAI
10	VAI APIs	REQ_22	Mobile client to monitor the alarms and events

Table 1- Virtual Abstraction Interface requirements list

Test cases are developed and additional test cases will be added to validate each requirements. Categorized in essential, required and optional will help deliver in different stages. Validation team can run validation test cases in phase. Each request number can have multiple sub-request to fully describe the exact requirements. I will be update this part as an incremental manner.

Functional Overview

Functional overview comprises of architecture of Virtual Abstraction Interface (VAI) with major components. Figure-1 depicts the components and relationship among them. This section provides a brief description for each component to describe implementation overview of the functionality.

The important features are as follows:

- Creation/ Deletion of VMs in OpenStack based datacenter
- Creation/ Deletion of VMs in VMWare based datacenter
- Track the VM usage and generate alarms on crossing threshold
- Resource creation in remote datacenter upon un-availability on local servers
- Network stitching using VxLAN and Open vSwitch
- Event logging
- Distributed message queue using Rabbit MQ
- Policy Validation

VAI APIs:
Virtual Abstraction Interface (VAI) APIs are the set of APIs exposed by the VAI service to the other modules. Other modules in the solution will be utilizing this API to communicate.

VAI Controller:
This module will act as hub between request submitted to VAI, validating policy and virtual platform implementation. All the communication will through Rabbit MQ message queue.

Access Service:
Access Service is external and part of the Access Control Service (ACL) of the project. Each request receives a token from the high layer as a part of the request. API layer will confirm the token by calling Access Service. This is an external entity to this module. It shown here as the VAI will utilize its service.

Policy Manager:
This process will allow the uses to create various policies related to the resource allocation for each request based on the role and access permission. Each request from the DMaaS server will be validated based on the role and types of resource requested.

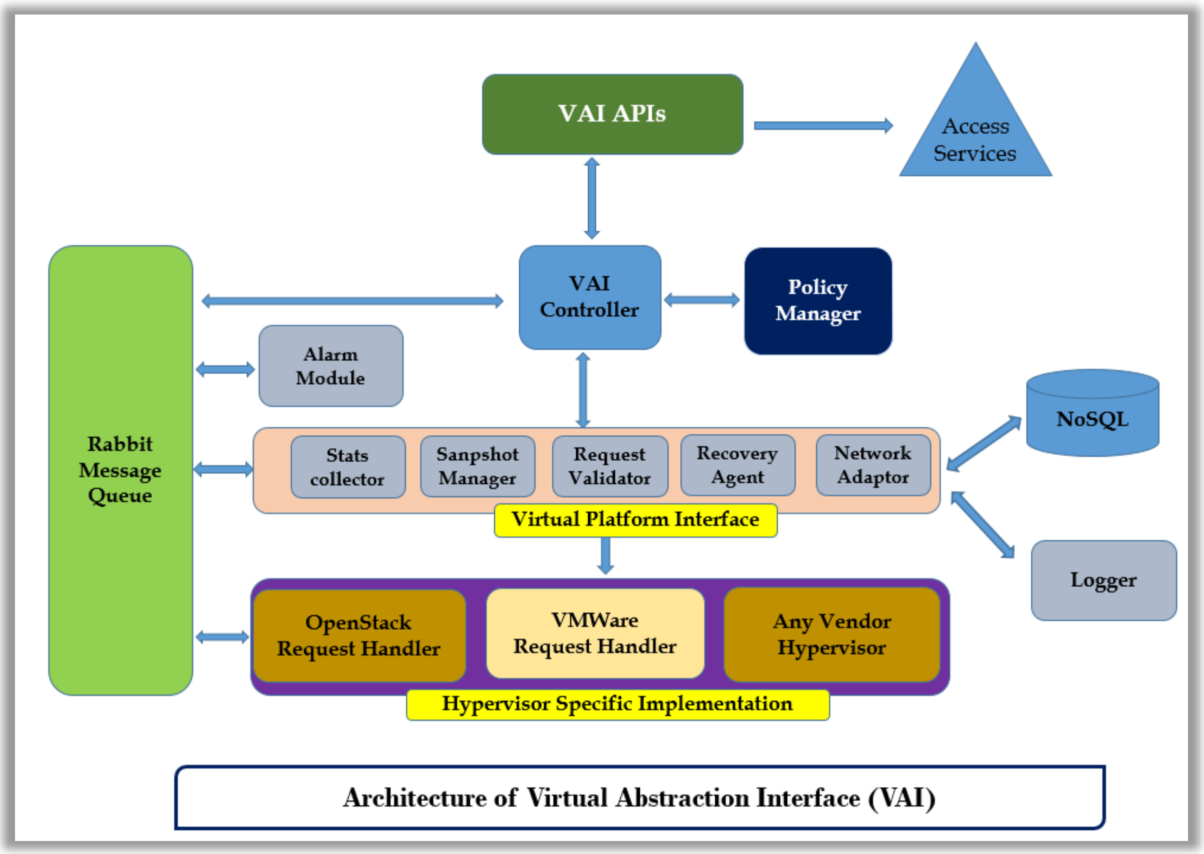


Figure 1 - Architecture Diagram for Virtual Abstraction Interface Module

Rabbit MQ:
Rabbit MQ will be used as the message queue service in the system. In the distributed environment, various process, will communicate using this process. This module will provide the callback registration for each user. These callback will invoked when the data arrived for a registered user.

Alarm Module:
This is external process. VAI only access this process for sending alarm, which will be delivered to the users based on the configuration.

Stats Collector:
Stats collector thread will interact with the hypervisor and collect the performance matrices like CPU, Memory and Network usage. This will also collect the power status and system running history. All the collected statistics will be formatted in specific manner and send it to the database using DB Proxy module for further usage.

Snapshot Manager:
Snapshot thread main job is to connect all the hosts and virtual machine to create snapshots, which will be used to recover the system from failure. The duration of the snapshot creation is configurable parameter. This will remove the prior snapshot before it creates new snapshot.

Request Validator:
The request validator process will active all the time and called when any resource request is submitted to the system. Each request is validated based on the role allocated for tenant. If the user is allocated with role of tester and trying to put a request for VM with high resource then the request may not qualify and negative response will send back to the user interface. The available resource will also be taken in to account while validating the request.

Recovery Agent:
The recovery agent thread will keep tracking the health status for the VMs and Hosts. When any VM or Host failed, can be detected by this thread it will start the recovery procedure. If the auto-recovery is not configured by the user then it will not initiate the procedure.

Network Adaptor:
When the datacenter doesn't have enough resource to serve the user request then VAI will get the resource from the users other datacenter. As a part of making the VM in the same network the layer 2 of the VM has to be extend. This will be achieved by VxLAN. Network adaptor thread will handle all the network creation and stitching work.

OpenStack Request Handler:
This process will take care OpenStack related implementation. All the north bound requests will be mapped to the hypervisor's specific calls at this point. All the OpenStack NOVA APIs will be implemented here.

VMWare Request Handler:
This process will take care VMWare related implementation. All the north bound requests will be mapped to the hypervisor's specific call at this point. All the VMWare VSphere APIs will be implemented here.

Logger:
This logger module will take care of the database connectivity and communication. All the debug / event logs will be submitted to the database logger module. This will implement the database proxy for MongoDB.

Configuration/ External Interfaces

This module will not use any external components other than cloud platform to host the solution. In addition to its own modules it will use Access Service, Logger and Alarm Service which are part of complete project. The cloud platform (Amazon Web Service) configurations will be done manually, while setting up the service.

All the configurations related to Access Service, Logger and Alarm Service will be configured in the database and loaded to VAI controller during system booting time. It might include code to reload the configuration parameters if updated by the user.

Debug

This module will implement the logging mechanism for debugging purpose. There is an optional requirement to add AOP (Aspect Oriented Programming), using which we can more debug information, when needed.

Logging

This module will update the following types of logs:

1. **Event trace logs:**
   Event tracing can be of different level, ranging from level-1 to level-5. Based on the debug parameters it will set. Level-1 will have less number of logs with high level data and Level -5 will have high volume of logs with rich of information on specific module for analysis.

2. **Debugging logs:**
   Debugging logs can be enabled / disabled for printing different parameter value at runtime.

3. **Need basis will update the LINUX syslog:**
   It might also possible to add log syslog. Users can utilize the syslog to predictive analysis.

Implementation

Virtual Abstraction Interface (VAI) module provides following functionality:

- Abstraction layer to create virtual machines (VMs) for any underlying virtual environment.
- This solution includes VMWare hypervisor and OpenStack environment for VM creation.
- The objective of this module to give a simple interface for users to operate their datacenter.
- Users can create/delete VMs.
- Read different performance indicators.
- If the resource is not available in one of the data center then it will create VM in a different datacenter and stich the network using VxLAN and Open vSwitch.

VAI module is combination multiple sub-modules like: VAI Controller, Stats Collector, Snapshot Manager, Request Validator, Recovery Agent, Network Adaptor, OpenStack Request Handler, VMWare Request Handler, Logger, Rabbit MQ and Alarm Module. It is broken down to more specific task based sub-modules to make it more modular. This will help for enhancement and debugging.

Virtual Abstraction Interface module interact with other layer's service using application program interface. It has four **interface** like:

1. VAI API
   This interface will be used for the message communication between DMaaS server and VAI layer.

2. Logging interface
   VAI activities will be logged in addition to the system logs and user usage pattern in the database for analysis and usage recommendation

3. Alarm interface
   On crossing the configured threshold values VAI layer will notify alarm module to raise appropriate alarm to user using this interface

4. Access service interface
   Each request from DMaaS will contain token which VAI API service will validate against access service for security purpose.

VAI Call Flow:

Figure-2 depicts the call flow of VAI, which will be running for each individual cloud cluster. The sequence of execution is numbered from 1 to 12. The below describe each steps in the resource creation and stitching.

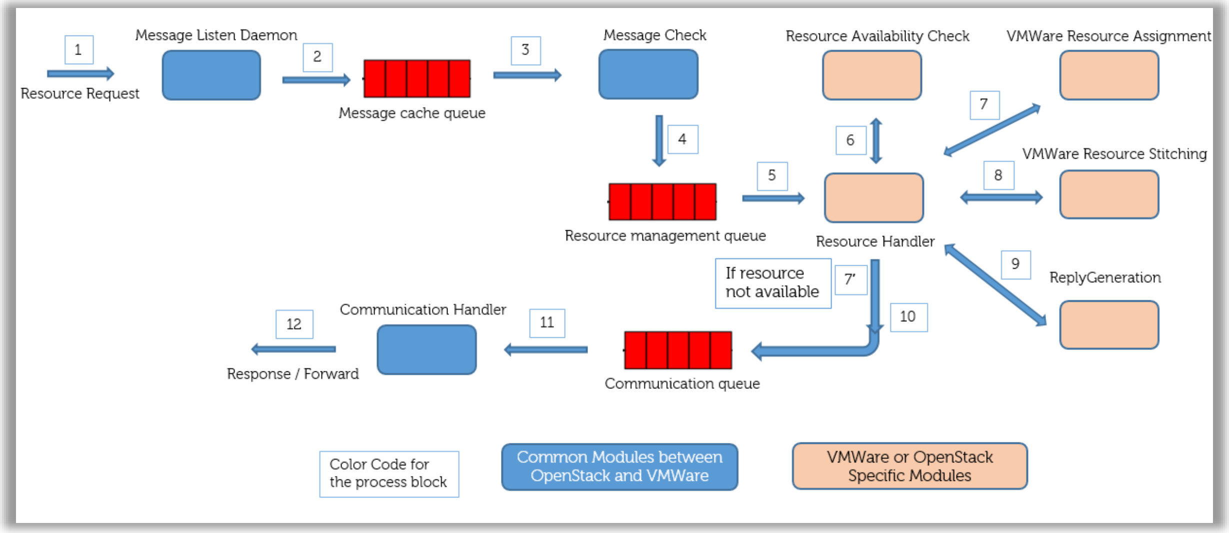

1
Resource Request
Message Listen Daemon
2
Message cache queue
3
Message Check
4
Resource management queue
5
Resource Availability Check
6
Resource Handler
7
VMWare Resource Assignment
8
VMWare Resource Stitching
9
ReplyGeneration
If resource not available
7'
10
Communication queue
11
Communication Handler
12
Response / Forward
Color Code for the process block
Common Modules between OpenStack and VMWare
VMWare or OpenStack Specific Modules

Figure 2- Call flow diagram for VAI Module

1- Each resource request will be first received by Message Listen Daemon (MLD).

2- MLD will read the message from the TCP socket and push the data in the Rabbit Message Broker (Message Cache Queue)

3- Message Check will provide the callback mechanism, which will be registered with RabbitMQ. When the data will arrive for Message Check, the callback will be invoked and message will be processed.

4- After the message is read from the MQ it will be processed and pushed to the Resource Management Queue.

5- Resource Handler will provide the callback mechanism to read the data from Resource Management Queue and start the resource discovery.

6- Resource availability will be under Virtual Abstraction Interface module. This will have platform dependent implementation. This will validate the available resource against the requested resource. Currently this solution will implement only OpenStack and VMWare functionality.

7- Resource creation stage comes after validation phase. This stage will also have platform specific implementation. The required virtual machine will be created at this stage.

8- This stage will implement the code to extend the TCP/IP layer 2 functionality across different datacenter using VxLAN overlay networking

9- This module will create the appropriate message back for the requester. All the message buffer will be used JSON object. This is a standardized method of data communication across different processes.

10- Resource Handler will create the virtual machine and pushed the configuration details back the requester by pushing in to the communication queue.

11- Communication handler will have the callback mechanism to reading message from the communication queue and send to the outside entity using socket.

12- Message that is going back to the requester and forwarding to other datacenter in case of unavailability of the local resources.

Integration Point:

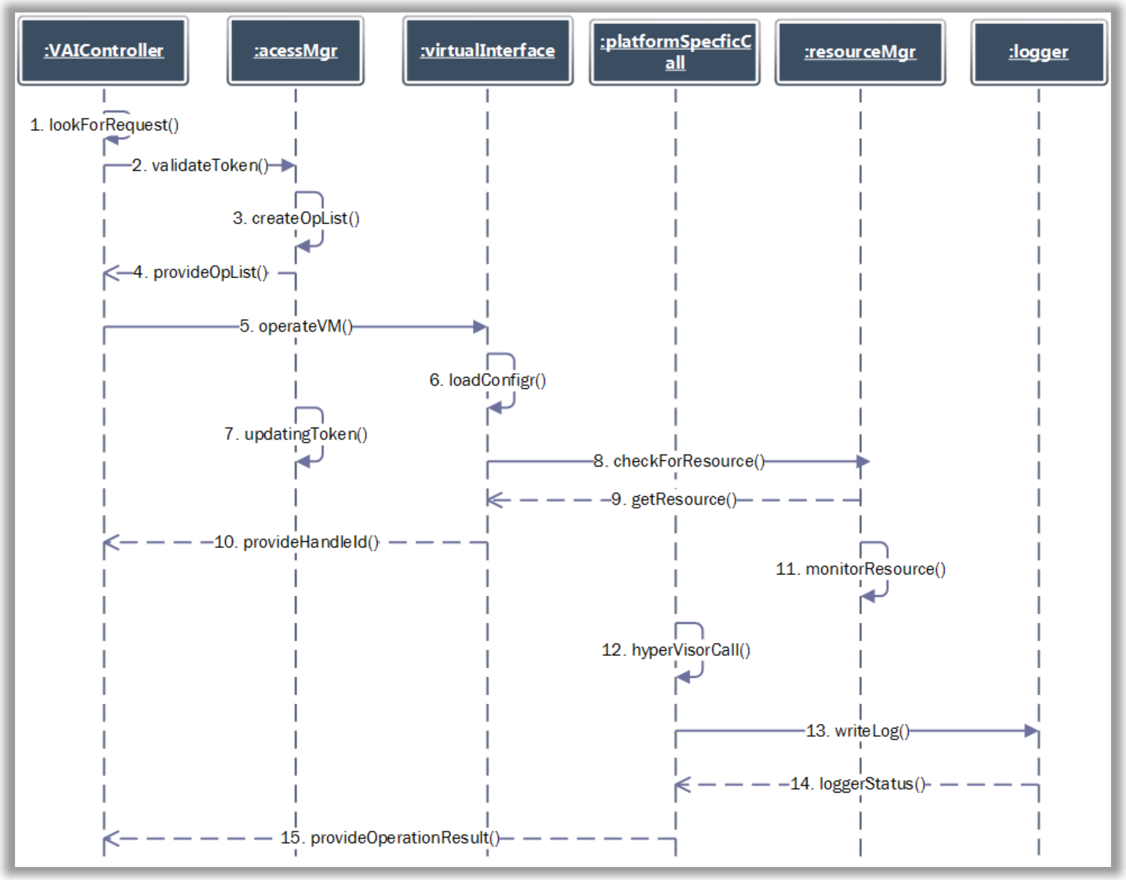


Figure 3 - High level sequence diagram for VAI

Virtual Abstraction Interface needs integration with logger, access service API and alarm module for compilation purpose. The design of DMaaS provides loosely coupled model, so that the enhancement to any module is independent in nature. High level call flows for VAI are depicted in Figure-3. VAI controller will always active and look for new request. On receiving of request it will validate the token with Access Service before it pass the request to VAI. VAI module will interact with resource manager and validate the request. VAI will complete the request log the activity and provide the response back to the user.

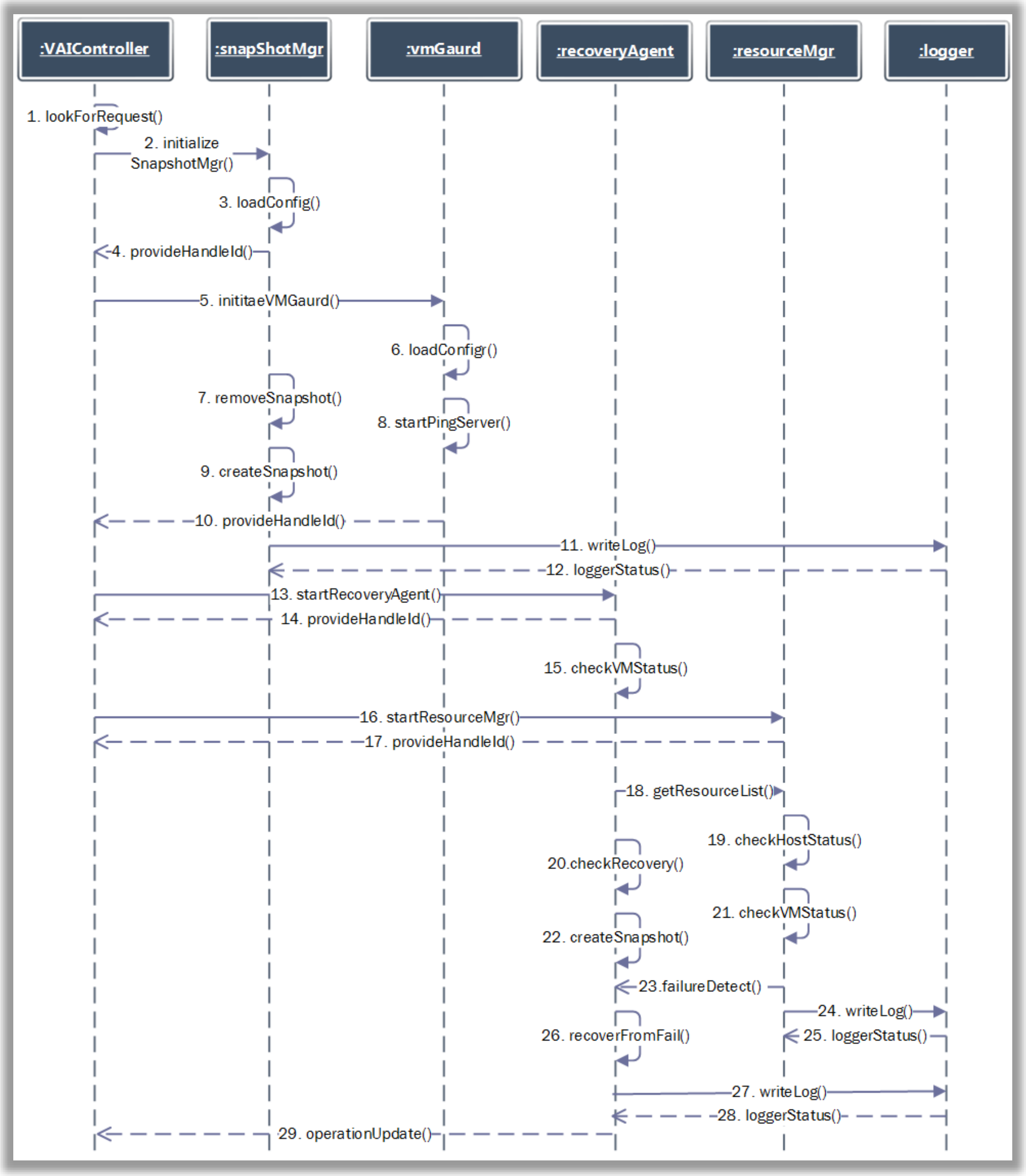
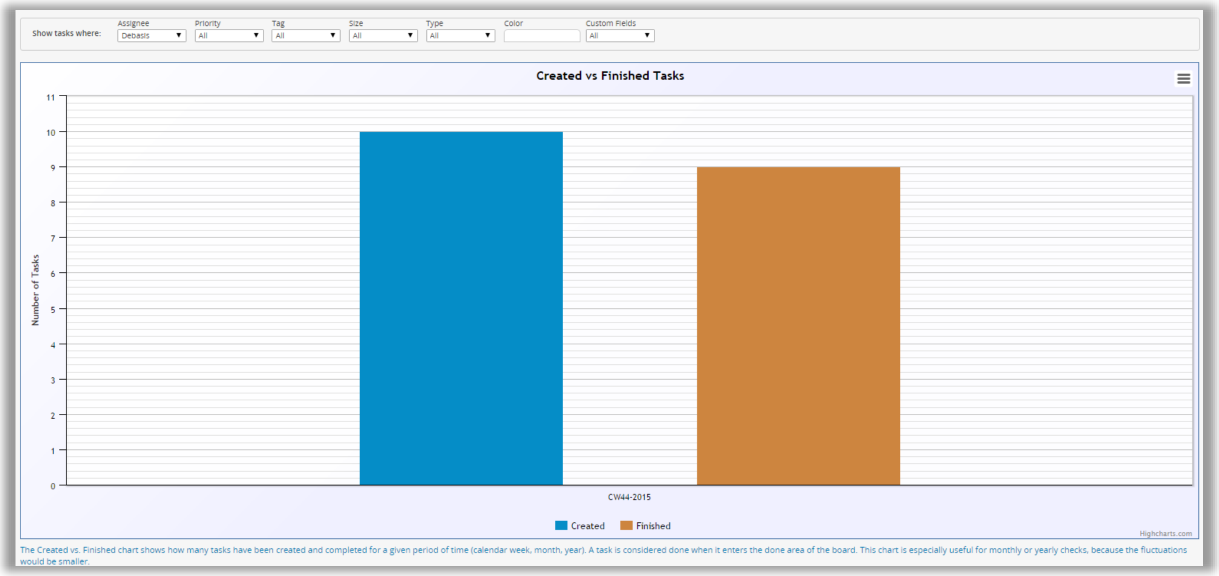


Figure 4 - Sequence diagram for failure detection and recovery

Figure-4 presents the call flows for failure detection and recovery of virtual machines (VMs) and hosts. The major component like snapshot manager, recovery agent will be initialized by VAI controller, when the process started. Snapshot manager will keep creating snapshot of VM and Host on configured amount of time. Each time it creates the snapshot, it will remove the previous snapshot to preserve storage space. Recovery Agent will continue monitoring the health status using ping message. If ping fails, it will retry configured amount of time. On the failure of ping procedure, recovery agent will start the recovery process. As a part of recovery process it will try to recover the VM from the snapshot, if the host is not accessible then, it will first recover the host, then VM. If the host is not able to recover then it will try to launch the VM in other available host.



The Created vs. Finished chart shows how many tasks have been created and completed for a given period of time (calendar week, month, year). A task is considered done when it enters the done area of the board. This chart is especially useful for monthly or yearly checks, because the fluctuations would be smaller.

Figure 5 - Individual task progress status for Debasis

The figure- 5 shows the number task created vs number task completed. The project work is going as per the plan. Nine out of ten task is in completes stage. This graph is generated from the "Kanabaize" tool, which we are using our project planning and progress tracking. This shows the progress status only for me.

Testing

This section describes the approaches to test VAI module. The testing includes the following type:

Integration Testing

The Integration testing will executed to validate the integration of this module with the complete project delivery. The **integration test cases** will be **covered as a part of the work book-2**. The scope includes end to call flow inside the project. Some of the example of the test cases as follow:

- User shall able create/delete VMs from the user interface
- DMaaS server should only allow the validated tenant which has right access to the system
- User shall able to see the recommendation on the resource usage pattern
- Web based service will be protected from the DDoS attack

Function Testing

As a part of the functional testing, the test cases will be developed to test the complete functionality as whole. Mostly all the **black box test cases** will covered under this.

Unit Testing

Unit testing will cover the test cases to validate code, such that it will run through each of line of code. This will cover **white box testing**. Following test cases are one part of the unit test coverage.

Unit Test case:

The following test cases will be executed as a part of the unit test coverage:

Case Id	Functionality	Description	Expected Result	Actual Result	Remark
1	**Stats Collector**	Loading Configuration	Stats Collector shall able load the configuration parameters form the database		In Progress
2		Update Configuration	Stats Collector shall able to reload the modified parameters using DB Proxy		In Progress
3		Connectivity to hypervisor	It shall able to connect to the specific cloud cluster like OpenStack, VMWare with the tenant credentials	As expected	Done

4		Collection of Performance counters	Stats collector shall able to collect the performance metrics like CPU, Memory and Network usage.	As expected	Done
5	**Snapshot Manager**	Loading Configuration	Snapshot Manager shall able to load the configuration parameters like interval time to create snapshots from the database		Yet to complete
6		Reloading of the parameters	It shall able to use the new configuration parameters when the user update the configuration		Yet to complete
7		Removing of snapshots	The module shall able to connect to the hypervisor and remove the snapshots	As Expected	Done
8		Taking new snapshot	This module shall able to connect to the hypervisor to create new snapshots and rename with the time stamp	As Expected	Done
9	**Request Validator**	validating the request	Request validator shall able to access the datacenter resources availability to validate the request from the user		Yet to complete
10		Recommend different location	If the resources are not available in the chosen datacenter, it should able to check the availability in other accessible datacenter and recommend it to the user		Yet to complete

11	**Recovery Agent**	Failure Detection	Recovery Agent shall be running all the time in an independent thread and keep checking the health status of the virtual machines. Shall able to detect any failure		Yet to complete
12		Recovery of VM	If the virtual machine is failed due to some reason and host is accessible. It shall be able to deploy the previous taken snapshot		Yet to complete
13		Recovery of Host	If the host failed the module shall able to recover it from the host snapshot		Yet to complete
14		VM transition	It shall able to move the VM from one host to another if it so configured		Yet to complete
15	**Resource Manager**	Collection of Resources Availability	Resource Manager shall able to connect all the configured datacenter and collect the available resource		Yet to complete
16		Validate resource request	This module shall able to validate the resource request from the request validator		Yet to complete
17		Resource consumption report	The Resource Manager shall keep track of the consumption of the Resource usage inside the datacenter		Yet to complete
18	**Alarm Module**	Alarm generation	The system shall able to create alarm for crossing threshold limit		Yet to complete
19		Lowering Alarm	When the system usage comes down from the threshold limit it shall able to		Yet to complete

			reduce the severity of the alarms		
20		Sending alarm to DMaaS Server	This module shall able to send the alarm status to the DMaaS server process to the presentation to the user		Yet to complete
21	**Rabbit Message Queue**	Initialization of the message queue	Rabbit message queue shall able to initialize from the configuration parameter and keep running to receive the messages		Yet to complete
22		Allow to register the callback methods	Rabbit MQ shall allow to user to configure the callback methods and take action when data is available for the said module		Yet to complete
23	**Logger**	Record activity log	Logger module shall able to capture all the activity and event logs. This will be pushed to the database using the DB Proxy Module	As Expected	Done
24	**Database Proxy**	Connectivity to Database	Database Proxy module shall able to connect to the required database with the configured credentials	As Expected	Done
25		Archiving Logs	DB Proxy shall able to push all the activity logs to the database for further analysis		Yet to complete
26	**Policy Manager**	Validating the Tenant Policy	This will validate the request against the configured policy		Yet to complete

27	**Hypervisor Specific Calls**	OpenStack API Call	Hypervisor specific calls will be resolved here by providing the abstraction. The system shall able to connect to the OpenStack and complete all the functionality	As Expected	Done
28		VMWare API Call	When the configured datacenter is VMWare based this module able to resolve the request and do the appropriate calls to the south	As Expected	Done
29		Providing Abstraction to the higher layers	The user shall able to similar functionality with same manner for disparate virtual environment	As Expected	Done

Table 2- Test case list for VAI Module

Conclusion

VAI will provides the extensibility to the DMaaS project by providing unique approach to handle requirement for the virtual machines. This not only satisfy the demand of the resource, but it also provide ability to extend datacenter across regions. This provides a very user friendly, cost effective solution to the growing demand of virtual servers.

Appendix

OpenStack has seven core modules like: Compute (Nova), Network (Quantum), Object Storage (swift), Block Storage (Cinder), Dashboard (Horizon), Image (Glance) and Identity Server (Keystone). We will be using these components for the project delivery.

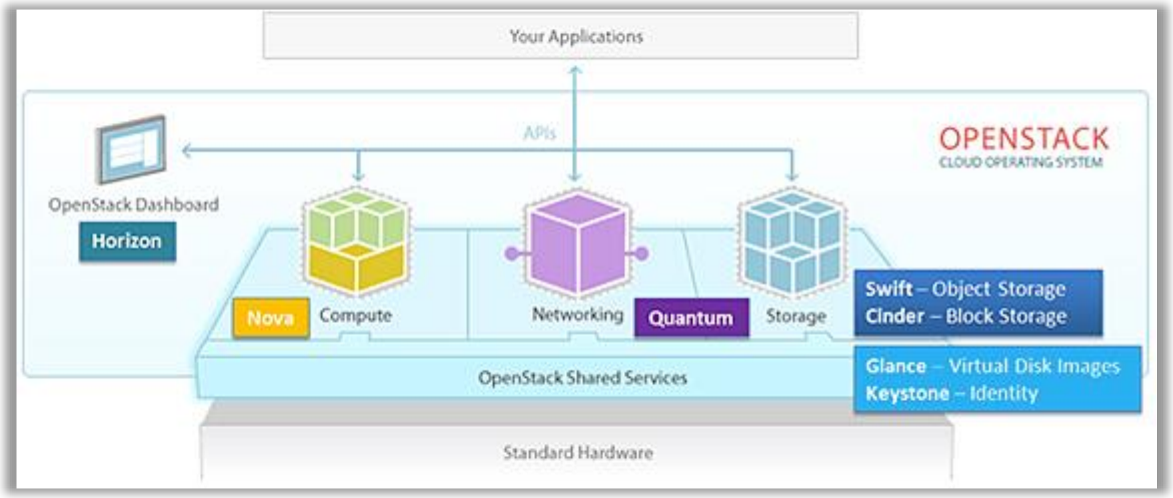


Figure 6 - Core components of the OpenStack architecture [6]