# Gap Analysis on Openstack for DPACC

Table of Contents

[Gap Analysis on Openstack for dpacc 1](#_Toc438976281)

[1 Introduction 1](#_Toc438976282)

[1.1 OpenStack Introduction 2](#_Toc438976283)

[2 Architecture 3](#_Toc438976284)

[3 Life Cycle Phases 3](#_Toc438976285)

[3.1 Resource Discovery 4](#_Toc438976286)

[3.1.1 Requirements Summary 4](#_Toc438976287)

[3.2 Resource Scheduling 5](#_Toc438976288)

[3.2.1 Requirements Summary 5](#_Toc438976289)

[3.3 Association Setup 5](#_Toc438976290)

[3.3.1 Requirements Summary 6](#_Toc438976291)

[3.4 Status Update 6](#_Toc438976292)

[3.4.1 Requirements Summary 6](#_Toc438976293)

[3.5 Association Termination 7](#_Toc438976294)

[3.5.1 Requirements Summary 7](#_Toc438976295)

[4 References 7](#_Toc438976296)

[5 Editors 8](#_Toc438976297)

[6 Contributors 8](#_Toc438976298)

# Introduction

This document serves as the gap analysis on OpenStack [1] for DPACC [2] which includes five sections. Section 1 is the brief introduction. Section 2 introduces the architecture of host AML as part of DPACC Architecture [3] for local/remote acceleration orchestration and managing, where a new managing entity is defined as Local Manager which is responsible for interfacing with acceleration resource locally on a compute node or remotely as a resource pool. Section 4 which is key section of the whole document introduces the life cycle phases of accelerator orchestration and management, including [Resource Discovery](#_Resource_Discovery), [Resource Scheduling](#_Resource_Scheduling), [Association Setup](#_Association__Setup), [Status Update](#_Status_Update) and [Association Termination](#_Association_Closing). Section 5 specifies future work plan to heal the current gap on OpenStack for above specified requirements for acceleration orchestration and management.

## OpenStack Introduction

OpenStack [1] software controls large pools of compute, storage, and networking resources throughout a datacenter, managed through a dashboard or via the OpenStack API. OpenStack is not a single piece of software, but instead consists of several projects that each one covers a specific functionality. Among these projects, Nova, Cinder, Neutron, Swift, Glance, and Ceilometer are considered to be the core projects that are providing basic functions.

The gap analysis within the current document will focus particularly on three OpenStack projects described above: Nova, Cinder and Neutron.

Nova [4], which provides compute capability of executing related workloads for OpenStack, is not restricted to any specific virtualization technologies such as KVM, Hyper-V, Docker, QEMU etc. Nova is composed of major components like Nova API, Nova Scheduler, Nova Conductor, and Nova Compute. Nova Scheduler is responsible for placing compute instances on hosts, with various filtering rules set according to CPU type, virt type, Usages, PCI availability, etc. Nova Conductor handles database access, where hosts issue RPC calls and conductor update the database. Nova Compute talks to the host.

Cinder [5] offers the persistent block storage for Nova. It supports multiple storage backends and expose iSCSI (primarily) I/F to the host.

Neutron [6] deals with network management, and supports multiple L2/L3 plugins.

# Definitions

TBA. (direct definitions or indirect reference)

# Architecture



Figure 2‑1 Architecture of DPACC Orchestration and Management
 ([https://wiki.opnfv.org/\_media/dpacc/figure\_openstack\_fig1.rar)](https://wiki.opnfv.org/_media/dpacc/figure_openstack_fig1.rar%29)

In the above figure of architecture. Network element is divided into three layers.

The first layer is MANO which includes NFVO and VNFM, who are responsible for NS and VNF life cycle managements respectively as defined in [7].

The second layer is VIM which can be implemented by OpenStack, is responsible for manage and control the cloud infrastructure resource pool.

The third layer is the local manager who is responsible for acceleration orchestration and management for local accelerators locally on the compute node or remotely from a shared resource pool.

It is intended that the local manager is separated from hypervisor and located in the user space.

# Life Cycle Phases

In this section, for each intended acceleration management work flow, the requirements for VIM are specified and the gaps on the current OpenStack implementation are identified.

In particular, from the perspective of an accelerator's lifecycle, five DPACC-specific work flows that involving the VIM implementation are specified, including: one for accelerator discovery (i.e. the Discovery Phase), one for management and scheduling, three for accelerator employment (including Scheduling and Setting-up and Update sub-phases) and one for resource revocation (i.e. the Closing Phase).

## Resource Discovery

To perform resource management, VIM needs to have awareness for the existence of Accelerator Devices as well as its capacity and Acceleration type though the Local Manager once it attaches to a compute node (as a local accelerator), or when the compute node is powered on (as an integrated accelerator), or when it is ready to be used remotely by a compute node via a general resource pool (as a remote accelerator), who in turn reports to MANO.

However, it may cause headache for MANO, if VIM actively reports device discoveries it has no intention to be aware of. Therefore, it is expected that VIM need to support passive query or active subscription of device discovery/status from MANO, where MANO sends a query or subscription request for certain acceleration resources that is required in certain NS/VNF deployment/monitory as specified in corresponding VNFD, and later once discovers corresponding device as specified in earlier request, VIM reports back as a notification/publication to that query/subscription about the existence of or up-to-date state of the acceleration resources.

Note the discovery phase does not include the software-accelerators contained in the VNF.

### Requirements Summary

Requirement 3‑1 VIM SHOULD have the capability to discover, or to be notified of, configuration features such as capabilities, compatibilities, versioning and so forth, for a given set of acceleration resources.

Requirement 3‑2 VIM SHOULD support notification of acceleration resource discovery under the circumstance that MANO have requested notification of this type of event.

## Resource Scheduling

In DPACC context, the acceleration capabilities are integrated into the servers where allocated VMs run on top of. Such integrated acceleration may be software acceleration, integrated hardware acceleration into the processor of the server, or acceleration coprocessor plugged into the server.

Specifically, VNFD may include specific requests for certain acceleration capability. When VIM identifies and schedules VM(s) for processing a VNF. VIM needs to be able to identify acceleration specific requirements from received VM template and do filtering on available compute nodes accordingly in order to provide VM(s) which meet the VNFD request in terms of acceleration capabilities.

In addition, VIM needs to maintain a catalog for recognizable acceleration resources as well as an inventory for up-to-date running status for all available instances, as specified later in Section 3.4, to facilitate acceleration-specific resource scheduling.

### Requirements Summary

Requirement 3‑3 VIM MUST support scheduling of VMs which support certain desired acceleration capabilities, as specified originally in VNFD.

Requirement 3‑4 VIM MUST be able to use attributes described in VNFD to filter resources in order to provide VM(s) which meets the VNFD request in terms of acceleration capabilities.

Requirement 3‑5 VIM SHOULD support management of acceleration resource within, or across multiple NFVI-POPs.

Requirement 3‑6 VIM MAY support the data store of acceleration resource configuration feature.

## Association Setup

Once a VNF instance has been scheduled to run on a certain VM which provides the acceleration capabilities that the VNF's VNFD describes, the next step is to set up the connection between the VM and the assigned acceleration resources.

To setup a data path from the intended application to be accelerated (e.g. a VNF instance) to the allocated accelerator, VIM will be involved into the following two processes:

1. Acceleration Scheduling: VIM receives request from MANO to allocate set of virtual accelerators to VNF. VNFD includes description of required acceleration capability.
2. Acceleration Setup: VIM contacts with local manager to associate acceleration via API access for hardware and/or software accelerators.

### Requirements Summary

Requirement 3‑7 VIM MUST support contacting with local manager to associate acceleration via API access for hardware and/or software accelerators, as part of VNF instantiation.

## Status Update

Once the Acceleration Resource is allocated or changed during its lifecycle, the VIM needs to have awareness for its up-to-date status via the local manager (e.g.by periodical polling/asynchronous update) and reports back to MANO.

In case of resource KPI or acceleration requirement of a running VNF rises/declines, the VIM may trigger updates for acceleration association, upon request from MANO, for the VNF to scale up/down, i.e. increase/decrease the allocated acceleration resource of a given VM.

### Requirements Summary

Requirement 3‑8 VIM MUST manages a repository inventory with regard to both hardware and software accelerators.

Requirement 3‑9 VIM MUST manages a catalogue of virtualized acceleration resources can be consumed from NFVI.

Requirement 3‑10 VIM MAY support the capability of modification (Increase, Decrease, Update) of NFVI acceleration resources upon request from MANO

 Requirement 3‑11 VIM SHOULD support notification of NFVI acceleration resource modification as per MANO’s request or earlier subscription.

Requirement 3‑12 VIM SHOULD provide a general interface for the MANO to query for or subscribe to service-specific accelerator status/stats (.e.g. traffic stats or duration time for the IPsec tunnels).

## Association Termination

There are two situations in consideration here:

1. **Graceful Closing**, where the closing request is coming down from MANO. In this case, MANO requests VIM to bring down VNF or to unassociate an accelerator from a running VNF. The local manager unassociate the virtual accelerators from the vNF. VIM updates its local accelerator database based on local manager's execution report.
2. **Non Graceful Closing,**when VM crashes or accelerator crashes. The local manager recovers virtual accelerators from crashed VNF or remove/reinstall crashed accelerator, and informs the VIM about relevant stats/status update. VIM updates its local data store and notifies MANO if needed.

### Requirements Summary

Requirement 3‑13 VIM SHOULD support the capability to terminate the association of a given set of acceleration resources from a give VNF upon request from MANO

Requirement 3‑14 VIM SHOULD support notification of acceleration resource termination under the as per MANO’s request or earlier subscription.

# Gap Analysis and Work Plans

TBA

# References

1. OpenStack: [www.**openstack**.org/](http://www.openstack.org/)
2. DPACC Project: <https://wiki.opnfv.org/dpacc/>
3. Dpacc Architecture: <https://etherpad.opnfv.org/p/dpacc_framework>
4. OpenStack Nova Project: <http://nova.openstack.org/>
5. OpenStack Cinder Project: <http://cinder.openstack.org/>
6. OpenStack Neutron Project: <https://wiki.openstack.org/wiki/Neutron>
7. ETSI GS MANO 1.0.0: <http://www.etsi.org/deliver/etsi_gs/NFV-MAN/001_099/001/01.01.01_60/gs_NFV-MAN001v010101p.pdf>

# Editors

Lingli Deng

Vic Liu

Howard Huang

# Contributors

Mario Cho

Keith Wiles

Kin-Yip Liu

Peng Yuan