POSTER PAPER

Orchestration based Hybrid or Multi Clouds and Interoperability Standardization

Dinkar Sitaram (Dinkars@pes.edu), Sudheendra Harwalkar(sudheendraharwalkar@pes.edu), Chetna Sureka, Harsh Garg, Manusarvathra, Mayank Kejriwal, Shikhar Gupta and Vivek Kapoor

Center for Cloud Computing and Big Data,
Dept. of Computer Science
PES University
Bangalore, India

Abstract – **In the present scenario, hybrid or multi-cloud environments are most suitable for Enterprises and Communities (like Government of India) for cloud bursting, disaster recovery, migration, and there is growing need for unified monitoring and management, however, it's challenging to setup a viable hybrid/multi-cloud environment. Currently, there are multiple solutions available in the market with limited success due to hidden drawbacks, for instance, vendor-lock-in, portability issues in migration, security threats and expensive in the long run and also, unfortunately, interoperability standardization is still work in progress. In this paper, we explore few hybrid/multi cloud use cases and demonstrate how these can be accomplished with our Federated Cloud Services Framework (middleware), which is built upon OpenStack [5], an open source cloud and by leveraging existing OpenStack functionalities.**

I. Introduction

Cloud Computing provides dynamic on the fly scale up/down of infrastructure resources (Compute, Network and Storage) as needed and is widely used by start-ups to enterprises in handling peak load requirements. The scale-up is limited to hardware resources available within the organization; however, there are no issues in scale-up while using public clouds.

Enterprises are unwilling to move their entire operations to public clouds due to security reasons, and they have opted for a hybrid cloud (private cloud + public cloud) infrastructure. Currently, there are proprietary hybrid cloud solutions which are offered by public cloud service providers between private cloud and their cloud, and in this scenario, it becomes very difficult to migrate from one public cloud to other for better services and cost savings.

On the other hand communities (like Govt. of India) plan to deploy multi vendor multiple clouds in geo-locations and are interested to know on how to optimize the infrastructure resource utilization across clouds. Currently, there are no single solutions to support this requirement.

Interoperability Standardization and federation between clouds is the solution for above problems.

In addition to infrastructure scale, there are few other challenges in the hybrid or multi-cloud which are discussed in the next section.

II. Challenges with Hybrid or Multi-Cloud – Use Cases

A. System and data backup and disaster recovery[3][4]

Currently, majority of the enterprises have setup dedicated environments for backup and recovery within geographical premises, which is a very expensive proposition. Instead, a geographically located hybrid or multi-cloud solution, which provides both rapid scaling with low cost at the same time a flexible environment, is preferable.

B. Cloud bursting or auto scale[3][4]

Enterprises are keen on using cloud bursting, which allows a cloud to leverage resources from other clouds when required, and to return resources when they are no longer required. This improves utilization efficiency and is very cost effective. However, it is not easy to setup with current solutions available in the market due to security reasons, vendor lock-in and interoperability issues. Currently, there are no non-proprietary solutions for federation between clouds without manual intervention.

C. System and data migration[4]

Migrating applications in testing or staging environments to the public cloud is a compelling business case and necessary for many enterprises, which helps in speeding up the development/deployment process. However, there could be compatibility issues while migrating environments from private to public cloud or vice versa, especially when proprietary tools are integrated/used within the application or for backup, this could be a painful exercise for every product release, or it could lead to vendor lock-in.

D. Resource monitoring and management [7]

Most cloud providers (including OpenStack) have their own web-based proprietary application for cloud infrastructure monitoring and management; there could be a few with SSO (Single Sign-On) to view multiple homogeneous cloud resources. However, there are none for hybrid or multi-cloud environment with unified monitoring and management. There is also no standardization for management and configurations of cloud services and resources across different providers.

E. Data protection and security[7]

Data security is probably the biggest challenge in hybrid or multi-cloud adoption, as enterprises opting for private cloud is to safeguard valuable and sensitive data from theft and hence

978-1-5386-9441-1/18/$31.00 ©2018 IEEE
DOI 10.1109/CCEM.2018.00018

data transfer between the private and hybrid/multi-cloud will be crucial.
These challenges and use cases are addressed by our framework solution which maps other clouds as sub-clouds to OpenStack Cloud by using Proxy Cloud Virtualization layer.

III. BACKGROUND – FEDERATED CLOUD SERVICES FRAMEWORK

OpenStack API interface is supported either directly or via third-party solutions in most of the clouds (AWS, Azure, VMware, and etc.). Our solution uses a Proxy Cloud Virtualization layer (refer Figure 2.) that maps other clouds as sub-clouds (or pseudo availability zone) to an OpenStack cloud, OpenStack Availability Zone (AZ) which in turn allows the grouping of resources and can be allocated based on the requirements (refer Figure 1).

User requests are intercepted by our framework and are either passed to be executed normally or are re-routed to the remote cloud based on the availability zone selected; responses are routed back appropriately to the client.

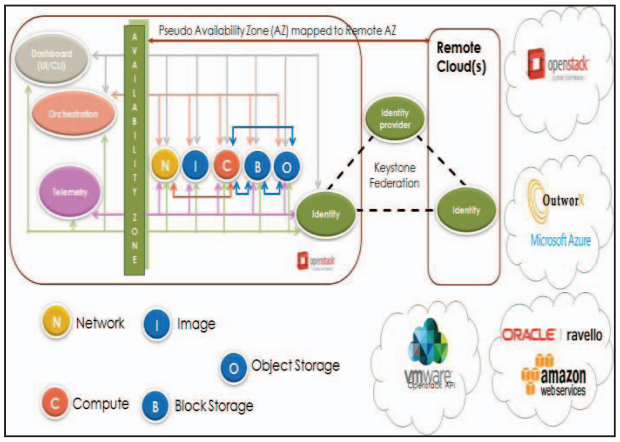


Figure 1: Federated Cloud Services Framework - Architecture

OpenStack Keystone federation (federated token) [6] is used for communication between multiple clouds. Refer Figure 1 for Federated Cloud Service Framework architecture and for more details refer [1][2]

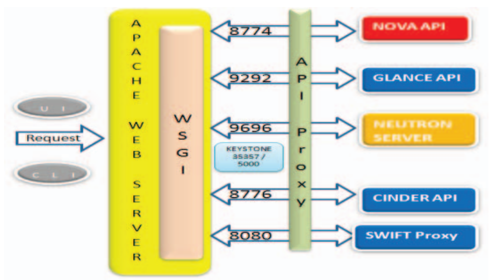


Figure 2: API Proxy (Proxy Cloud Virtualization Layer)

One of the key components in our framework solution is federated *nova* (compute.), which leverages existing OpenStack functionalities to support most of the hybrid or multi-cloud use cases that were mentioned earlier. Also, our solution has federated *cinder* for block storage, federated *swift* for container and object store, and federated *neutron* for the network related use cases in hybrid or multi-cloud environment.

Our framework supports *resource discovery*, *resource create*, *resource delete* and *resource detail* basic functionality in the hybrid/multi-cloud environment.

There are many third parties including major cloud vendors providing hybrid/multi-cloud service with broker architecture. In this case, enterprises/communities have to depend upon unified third-party proprietary tools for resource allocation, monitoring and management with an additional cost and there may be vendor lock-in with the broker. In contrast, our solution is built using OpenStack API for communicating across clouds.

In order to support features like disaster recovery, auto scale, migration and monitoring, we had to enhance our framework to include centralized/distributed *glance-cinder* store for backup and restore services, unified and integrated monitoring and a mix and match of resources in the hybrid/multi-cloud environment, which are discussed in subsequent sections.

IV. CENTRALIZED/DISTRIBUTED GLANCE-CINDER STORE

Glance Store
Our framework has been enhanced to support live backup and restore of VMs in a hybrid or multi-cloud environment, which in-turn uses OpenStack VM snapshot functionality. OpenStack APIs with keystone federation are used across clouds for communication.

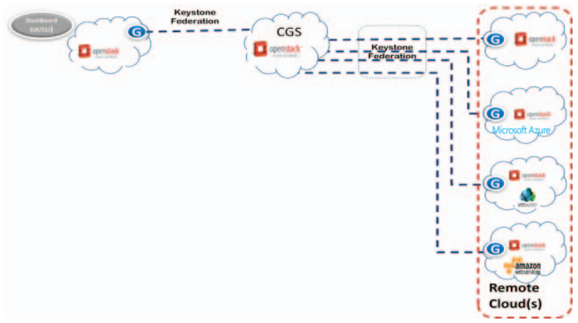


Figure 3: Centralized/Distributed *Glance* Store

Federated Cloud Service Framework supports the following image (*glance*) related functionalities (refer Figure 3) in a hybrid or multi-cloud environment:

- Image discovery, image detail, image creation, image delete.
- VM (*nova* instance) creation using the image from any of the registered clouds or central store with image caching enabled in the local cloud.
- VM snapshot - VM snapshot creation in a remote registered (or centralized *glance* store) cloud.

Cinder Store
Similarly, our framework is enhanced to support for backup and restore of *cinder*-volumes in a hybrid or multi-cloud environment, which uses OpenStack *cinder* snapshot/upload to

image functionality. OpenStack APIs with keystone federation are used across Clouds for communication.

Federated Cloud Service Framework supports following volume (*cinder*) related functionalities in hybrid or multi-cloud environment:

- Volume discovery, volume detail, volume create, volume delete.
- Remote instance (VM) remote volumes attach/detach
- Local instance (VM) remote volumes attach/detach
- Remote instance (VM) local volumes attach/detach, which allows us to leverage existing functionality in keeping the sensitive data volume within the private cloud and is visible and accessible to specific VMs in the hybrid or multi-cloud environment. This functionality will help in safeguarding the sensitive data within the private cloud firewall and also saves the data transfer time when the VM is migrated back to the private cloud from the hybrid or multi-cloud.
- Volume snapshot - volume snapshot creation in remote registered (or centralized *cinder* store) cloud.

Backup, disaster recovery and migration use case flow

1) Snapshot or backup requests by *nova* component are intercepted by our federated *nova* and federated *glance* and the requests are re-routed to the configured centralized/distributed *glance* store
2) Instances (VM's.) can be migrated/created in any registered cloud by choosing image stored in the centralized/distributed *glance* store.
3) Similarly *cinder* snapshot/upload to glance requests is processed by federated *cinder* and snapshots/glance image are stored in centralized/distributed *cinder/glance* store.

Python scripts implemented, to get all local instances ID and attached volumes for a project, and to initiate the periodic snapshot for all the listed instances and attached volumes using cron job and heat orchestration.

This implementation assumes that the instance is stateless and user data is stored in *cinder* volume.

With this solution, we support both (1) system and data migration and (2) system and data backup and disaster recovery use cases and compatibility, or vendor lock-in issues are addressed by standardization of OpenStack APIs across clouds.

V. Unified Monitoring and Management

OpenStack *ceilometer* polling agent poll OpenStack services, build meters and notification agents listen to notifications on the message queue, convert them to Events and Samples and apply pipelining actions.

Data normalized and collected by *ceilometer* can be sent to various targets. *Gnocchi* captures measurement data in a time series format to optimize storage and querying. *aodh* is the alarm service which can send alerts when user-defined rules are broken (refer to Figure 4).

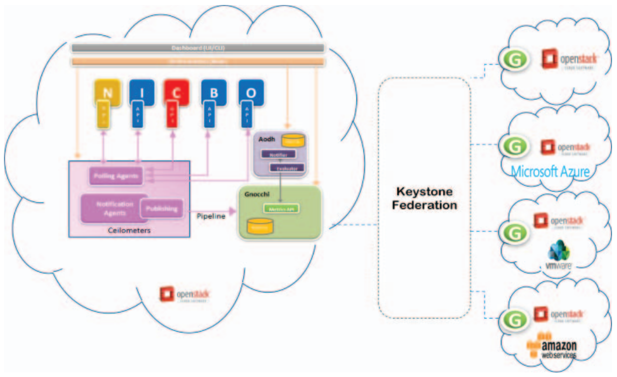


Figure 4: Unified Monitoring and Management

Implementation for unified monitoring and management is very tricky, since monitoring data collection and measure aggregation is performed on per cloud basis, and can be easily viewed using open source/commercial available products, but our requirement is to aggregate measures for resources across clouds. For example, in the case of cloud burst in a hybrid/multi-cloud environment, the VMs are created in multiple clouds and there is also a possibility of multiple VMs in one cloud depending on the auto scale policy and algorithm in the *heat* template, this can be solved by:

Developing a separate monitoring tool to collect data across cloud and aggregate measures and display, the management will have to be implemented separately. This will add overhead and as is OpenStack orchestration services can't be used. And same is the case with third-party monitoring tools.

Based on our investigations with very little documentation available, we found that there is a possibility of building federated *gnocchi* (refer Figure 5.) in line with other services in our framework to support integrated monitoring and management.

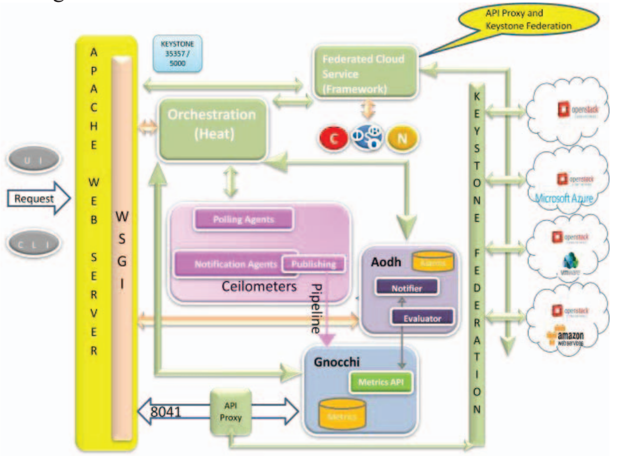


Figure 5: Federated *Gnocchi* Architecture

Gnocchi supports few aggregation methods (*mean, sum, last, max, min, std, median, first, count and Npct (with $0 < N < 100$))*. While these methods work perfectly within a cloud, but aggregation methods across clouds (aggregate of aggregates.) in hybrid/multi-cloud environment is not supported.

The aggregate of aggregates (mean of means) for *cpu_util* in a *server_group* across multiple clouds can be derived from the following formula:

$$\bar{X} = \frac{\sum_{k=1}^{nc} x_k n_k}{\sum_{k=1}^{nc} n_k}$$

x - Aggregate from k cloud
n - Instance count in a *server group* in k cloud
nc - Number of Clouds
$\bar{X}$ - Aggregate of aggregates (mean of means)

Similarly, formulas can be derived for other aggregation methods, as needed and necessary.

VI. Prediction with Dynamic Delta Correction Model

One of the challenges with enterprises is how efficiently private cloud resources are utilized and support for cost-effective auto scale or migration in a hybrid or multi-cloud environment with overall service assurance. Currently, broker architecture is widely used with limited number of public clouds for auto scale and or migration to public clouds, which involves cost of actual resource(s) + broker cost. In this paper, we are proposing a non-broker-based architecture, prediction and dynamic delta correction model with a look-ahead algorithm for efficient resource allocation/de-allocation for auto scale/migration in a hybrid/multi-cloud environment.

Dynamic and efficient Resource allocation

We have devised a generic mathematical model for resource allocation for auto-scale or migration based on a scaling factor, resource delta interference between virtual and physical resources and the cost of the resource. And delta interference and scaling factor are calculated as:

$$delta\_interference = f^{az}\left\{\left(\left[\sum_{i=1}^{n_{P_{res}}} P_{i_{res\_util}}\right] \div \left[n_{P_{res}}\right]\right) - \left(\left[\sum_{i=1}^{n_{V_{res}}} V_{i_{res\_util}}\right] \div \left[n_{V_{res}}\right]\right)\right\}$$

$$scaling\_factor = f^{az}\left\{\left(n_{P_{res}} \times P_{max_{res\_util}} - \sum_{i=1}^{n_{V_{res}}} V_{i_{res\_util}}\right) \div \left(\left[\sum_{i=1}^{n_{V_{res}}} V_{i_{res\_util}}\right] \div \left[n_{V_{res}}\right]\right)\right\}$$

f^{az} - Iterative function with availability zone(s)
P – Physical
V – Virtual
n – number
res – resource (compute/Network/storage)
res_util – resource utilization in %
max - maximum (for ex for CPU, it could be 80% resource utilization)

Our mathematical model suggests whether to auto scale or to migrate based on the following conditions (refer table 1 for sample resource availability calculations using *delta_interference* and *scaling_factor* formulas.)

$$\left(scaling\_factor - n_{V_{res}}\right) \div \left(2 \times n_{V_{res}}\right) \geq 1 \quad \text{.... Migration}$$

$$abs(scaling\_factor - delta\_interfernce) > 1 \quad \text{....Auto scaling}$$

$C_{R_{res}}$ Cost of remote cloud resource per hour is compared with multiple remote clouds for resource allocation in the remote cloud(s), prior to validating the scaling factor.

Sample resource availability calculations with a suggestion to possible solutions (migration and auto scale/auto scale) in an availability zone:

$n_{P_{res}}$	$n_{V_{res}}$	V_{res} %	V_{res}	V_{res}	V_{res}	V_{res}	$scaling\ factor$	Migration ≥1	*auto scale* >1
2	1	40	0	0	0	0	3	1	3
2	2	80	20	0	0	0	1.2	0.2	1.2
2	4	40	40	40	40	0	0	0.5	0
4	2	80	80	0	0	0	2	0	2
4	5	80	80	80	40	40	0	0.5	0
4	3	80	80	80	0	0	1	0.33333	1

$P_{max_{res\_util}} = 80\%$ Table 1

Sample Calculation: consider row number one in table1, number physical resource =2, Number Virtual resource =1 (40% utilized)

*scaling_factor = ((2 * 80) – 40) / (40/1) = 3, which is >1*
autocsale = scaling_factor =3 (assuming delta_interfernce is 0)
*migration = (3 -1)/ (2 *1) = 1, which is ≥1*

So, in this scenario either you can auto scale or migrate in selected availability zone.

In the Table 1, low numbers of resources are considered for simplicity in explanation.

Observations:

- Resource availability in AZ is directly proportional to the scaling factor.
- Scaling factor < 1, this availability zone doesn't have the required free resource for allocation.
- If migration is ≥ 1, this availability zone has sufficient free resources to migrate and scale.
- If the auto scale is > 1, this availability zone has sufficient resource to scale.

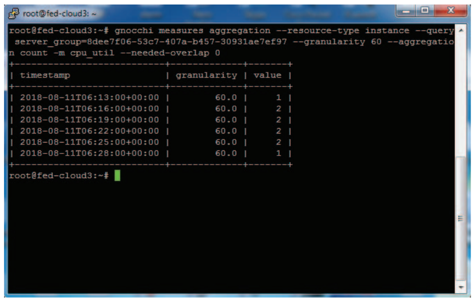
Prediction using time series data

Every day time series data is analyzed on varying data sample pocket size (P_{size}) and by averaging Peak Load resource Counts (PLCs) across multiple clouds to identify Peak Load Pockets (PLPs)

$$\begin{cases} i = 0 \\ \quad \mathrm{PLP}_{\mathrm{i_{st}}} = \mathrm{t}_{\mathrm{i} \times \mathrm{P_{size}}} \\ \quad \mathrm{PLP}_{\mathrm{i_{et}}} = \mathrm{t}_{(\mathrm{i} \times \mathrm{P_{size}}) + \mathrm{P_{size}}} \\ \quad \begin{cases} \Delta P_{size} = P_{size} \\ \qquad PLC_i = \left(\sum_{t=i\times P_{size},S_p}^{(i\times P_{size})+\Delta P_{size}} C_t\right) \Big/ (\Delta P_{size}) \\ \qquad PLC_i < 1, \quad \Delta P_{size} = ABS(\Delta P_{size}/2) \\ PLC_i \geq 1 \ or\ \Delta P_{size} \leq S_p\ , exit \end{cases} \\ \quad PLC_{i-1} \geq 1\ and\ PLC_i \geq 1\ , \ PLP_{\mathrm{i_{et}}} = \mathrm{t}_{(\mathrm{i} \times \mathrm{P_{size}}) + \Delta \mathrm{P_{size}}}, \text{exit} \\ \quad PLP_{\mathrm{i_{et}}} \geq t_{max}\ , exit \\ i++, i > (t_{max}/S_p)\ , exit \end{cases}$$

P_{time} $(Pocket\ time) = 60\ (min)$
S_p (Sample period) = 3 (min)
P_{size} $(pocket\ size) = P_{time}\ /S_p$
i(pocket number) = 0 and t_0(sample time) = 0
C_t *resource count at time t*
$\mathrm{PLP}_{\mathrm{i_{st}}}$(Peak Load Pocket start time) $= \mathrm{t}_{\mathrm{i} \times \mathrm{P_{size}}}$
$PLP_{\mathrm{i_{et}}}$(Peak Load Pocket end time) $= \mathrm{t}_{(\mathrm{i} \times \mathrm{P_{size}}) + \mathrm{P_{size}}}$
$\Delta P_{size}(partial\ pocket\ size) = P_{size}$

Federated *gnocchi* provides scaled-up resource count (refer the screenshot below.) for specific *server_group*, which is used as input to our prediction algorithm.



```
root@fed-cloud3: ~
root@fed-cloud3:~# gnocchi measures aggregation --resource-type instance --query
 server_group=8dee7f06-53c7-407a-b457-30931ae7ef97 --granularity 60 --aggregatio
n count -m cpu_util --needed-overlap 0
+---------------------------+-------------+-------+
| timestamp                 | granularity | value |
+---------------------------+-------------+-------+
| 2018-08-11T06:13:00+00:00 |        60.0 |     1 |
| 2018-08-11T06:16:00+00:00 |        60.0 |     2 |
| 2018-08-11T06:19:00+00:00 |        60.0 |     2 |
| 2018-08-11T06:22:00+00:00 |        60.0 |     2 |
| 2018-08-11T06:25:00+00:00 |        60.0 |     2 |
| 2018-08-11T06:28:00+00:00 |        60.0 |     1 |
+---------------------------+-------------+-------+
root@fed-cloud3:~#
```

Prediction with Dynamic Delta Correction model

Our model starts with traditional periodic monitoring with the threshold trigger and end of the day (or specified time period) time series data is analyzed and applied for subsequent day(s) and also periodic monitoring is used for delta correction.

```
Algorithm (auto scale and or migration)
Begin
Step 1: periodic monitoring
    Starts with traditional monitoring of multi cloud virtual
    resources cpu_util and or network_traffic and or
    storage_util threshold in a server_group or VM(s)
Step 2: action on threshold trigger
    On reaching the threshold, automatically on the fly one of
    the following operations is performed based on look-ahead
    resource availability matrix using mathematical model
    suggested earlier in this section
        • Attaches additional resources
        • Auto-scales within cloud AZs or remote cloud AZs
        • Migrates to new AZ (private or public cloud)
Step 3: End of the day (EoD) analysis
    End of Day analysis is performed based on predication
    using time series data
Step 4: Predictive/pro-active scale
    Based on the refined and corrected Peak Load Counts
    (PLCs) status ($PLC_i \geq 1$), Peak Load Pocket (PLP) start and
    end time ($\mathrm{PLP}_{i_{st}}$,$\mathrm{PLP}_{i_{et}}$), resources are automatically scaled up
    to $PLC_i$ at $\mathrm{PLP}_{i_{st}}$ till $\mathrm{PLP}_{i_{et}}$ and look-ahead resource availability
    matrix is used for resource allocation. Delta correction scale
    up/down is performed by step1 and step 2 periodically.
    Subsequently, either past EoD prediction can be applied or
    recalibrate prediction.
Repeat Step 1 – Step 4
```

Using OpenStack Telemetry services (*aodh*, *ceilometers* and federated *gnocchi*) and with incorporating *Dynamic Delta Correction model* in *heat* template, one can achieve very efficient auto scale/migration techniques in hybrid or multi-cloud environment.

VII. CONCLUSION

In the current context, more and more enterprises are looking for a quick turn-around for their business requirements by enabling hybrid or multi-cloud environment for cost-effective infrastructure at large scales, in addition to the existing private cloud. In this paper, we tried to articulate and address some of the hybrid/multi-cloud concerns and use cases with our framework solution. A major advantage of our framework solution is that it involves minimal changes to OpenStack and leverages the existing OpenStack base heavily. It also addresses hybrid / multi cloud challenges and issues, including security, effective management and migration complexities. We see a lot of traction from enterprises and especially communities (like Government of India). Once we complete development and testing, we intend to open source our solution or integrate with OpenStack releases.

Key Benefits:

- Centralized or multiple remote OpenStack image and volume (*glance* and *cinder*) store(s) for automated backup, disaster recovery and VM migration.
- Efficient and orchestration based cloud bursting in hybrid/multi-cloud environment.
- Provision to retain data volume in private cloud with secured access to VMs in hybrid/multi-clouds.
- Integrated single view for unified monitoring and management of distributed resources in private and hybrid/multi-clouds.
- In our solution - Federated Cloud Services Framework (middleware), OpenStack UI, monitoring software, workflow engine and other software can be reused without modifications.

Performance measurement was carried out in our environment with limited resources, and it's observed that there is a negligible delay (few seconds) in Auto scale, migration, and the procedure for disaster recovery on remote Cloud(s) as compared with the performance in the same or local cloud. For use case demonstrations refer [2].

Acknowledgement: Shreyas mm was involved in the initial stage of the project under the guidance of Dr. Dinkar Sitaram and Mr. Sudheendra Harwalkar from Cloud Computing and Big Data (CCBD) Center, PES University.

VIII. REFERENCES

[1] OpenStack Federated Cloud Services using API-Proxy and third party solutions. https://www.researchgate.net/publication/313822715_OpenStack_Federated_Cloud_Services_using_API-Proxy_and_third_party_solutions
[2] OpenStack Hybrid Cloud for Interoperability, Recorded presentation and demo at https://youtu.be/h7yyMS61MKo
[3] Blog: Hybrid Cloud 4 Top Use cases https://www.networkcomputing.com/data-centers/hybrid-cloud-4-top-use-cases/929963862
[4] Blog: Real Use cases: Why 50% of enterprise are choosing hybrid cloud https://www.cloudcomputing-news.net/news/2015/jun/25/why-are-50-of-enterprises-choosing-hybrid-cloud-real-use-cases
[5] OpenStack documentation from www.OpenStack.org
[6] Keystone federation setup http://docs.OpenStack.org/developer/keystone/federation/federated_identity.html
[7] Blog: What Are The Major Challenges Of Adopting A Hybrid Cloud http://www.micoresolutions.com/major-challenges-adopting-hybrid-cloud-approach/