EFFECTIVE CLOUD ORCHESTRATION APPROACH THROUGH XMPP

Vibha M B[1]
Assistant Professor, Department of MCA
Dayananda Sagar College of Engineering
Research Scholar, Bharathiar University,Coimbatore
Bangalore, India
Vishesh95@gmail.com

Dr. Raju R Gondkar[2]
Professor
CMR University
Bangalore
rrgondkar@gmail.com

Abstract— **Cloud computing enables extensive range of users to access distributed, scalable, virtualized hardware and/or software resources over the Internet. Multi-cloud deployments that are massive in scale are posing a challenge to manage the resources at the datacentres. Cloud resources can be managed efficiently through Orchestration. Orchestration enables us to combine the cloud resources and provides an access to the cloud architecture. In this paper, we propose a framework for cloud orchestration through Extensible Messaging and Presence Protocol (XMPP). XMPP is simple, dynamic and real time in nature. It is a middleware and powerful in user identification and group formation. In the current work, we recommend XMPP as a better alternative for orchestration in comparison with Advanced Message Queuing Protocol (AMQP). Orchestration can become a new industry standard.**

Keywords: Cloud Computing, Orchestration, XMPP messaging, Middleware, Datacentres.

1. Introduction

Cloud computing facilitates an extensive range of users to access distributed, scalable, virtualized hardware and software infrastructure over the Internet [1]. With well-defined service models, the cloud services are beneficial for both business and IT organisations. The elementary concept of cloud computing is virtualization. Virtualization is a process of creating multiple virtual machine instances for a single host machine in order to serve more number of users [2]. A VM has its own operating system, middleware, fundamental hardware assets and applications [3]. All the physical and the logical nodes are managed at the data-centres. The challenge at the datacentre is to optimize the resource allocation.

In the recent years, industry has made a remarkable impact on cloud computing. The modern society makes use of dynamic cloud-based administrations. The researchers and industries have set standards to procure the virtual assets in a dynamic and elastic manner. Cloud elasticity is a significant component for server implementation. NIST defines elasticity as the capacity for the clients to rapidly purchase on-demand and consequently release the required resources. The client gets a feeling of abundant cloud resources availability due to virtualization [4].The three unique methods in the implementation of cloud elasticity are replication, migration and resizing [5], which results in various methodologies to manage the resource allocation [6].

In the cloud provider's opinion, Infrastructure as a Service (IaaS) is the lowest level which is used to provide resources and storage to the users. Amazon, Google and Microsoft etc. are few of examples for resource provisioning. However, they are commercial cloud providers and are highly priced. In certain situations, it would be beneficial for the organizations to manage the resource provisioning on their own using open-source cloud platforms. With the emergence of several open-source technologies, it therefore becomes necessary to choose an effective approach to handle the client's request to procure VM.

Message Oriented Middleware (MOM), is a client/server infrastructure which makes it possible to distribute the application over dissimilar platforms. OpenStack is a popular open-source cloud platform operational on middleware [8]. It is a collection of softwares that enterprises or cloud providers can use to setup and run their cloud compute and infrastructure. It is popular platform among researchers, developers and enterprises [7]. Advanced Message Queuing Protocol (AMQP) is the existing messaging technology chosen by the OpenStack cloud [8]. In our study we compare AMQP with XMPP and we are able to suggest that XMPP protocol can be an alternative protocol for orchestrating.

XMPP is a network protocol and has seen popularity in chatting applications. Many successful researchers and developers have used XMPP at application level communication.

We know that XMPP protocol can reach at all levels of services and hence we make an attempt to adapt it for the communication at the middleware level.

In our work, we propose XMPP as the communication protocol which sets a new dimension for orchestration.

The main objective of this paper is to suggest an effective orchestration approach through XMPP to handle the requests of the end users for launching a new Virtual Machine (VM). XMPP server, being a middleware handles the request from the client to launch a new VM. The XMPP server exchanges

the message with the group nodes residing on the datacentre and responds back to the XMPP server with the available resources. The XMPP server based on the responses received will help the client to meet its request. The proposed work will suffice the requirement of resource allocation as well as load balancing. The work discussed in this paper, shows that it would be appropriate to include a middleware like XMPP to communicate between the user and the datacentre. The XMPP server is based on the message oriented model which has open standards and will be suitable to carry out the two-way communication.
The paper is structured as follows: Section 2 discusses the infrastructure management for dynamic provisioning of resources. Section 3, gives an overview of Cloud Orchestration, Message Oriented Middleware (MOM) and also highlights the need of XMPP for Cloud Orchestration in comparison with AMQP. In Section 4, the XMPP workflow is presented. Section 5 concludes the work with further scope.

2. Infrastructure Management for Dynamic Provisioning of Resources

Cloud Computing is the most popular computing in the recent past. The dynamic provisioning capability of hardware, software and services has made it the most popular. The cloud system utilizes the third party services over a network for dynamic provisioning. The elementary concept on which cloud works is Virtualization. This allows the cloud to create several logical virtual machines at the data-centres. The cloud system comprises three modules namely clients, data-centres and efficient distributed servers. The clients are the end users who intend to communicate with the cloud to manage information related to the cloud. The data-centre is a facility comprising of networked computers and storage that organizations use to organize, process, store and disseminate large amounts of data [9]. The distributed servers actively perform checks on their hosts to provide services and also to give an impression as a desktop application. The data-centres are responsible for effective task management and also to balance the load. The operations at the data-centres for resource allocation and load balancing have several approaches. The operations of mending resources like CPU, memory, OS, storage, network and databases to support an application requires a lot of configuration management. To fix the configuration management problems, different infrastructure management models/projects have been established.

3. Cloud Orchestration and Message Oriented Middleware.

Cloud services bear the operational complexities related to resource allocation, fault management, resource and service guarantees, image management, storage management etc. To execute all of the above, cloud services needs dynamic orchestration to obtain service abstractions [10]. Orchestration is to acquire an autonomic control or intelligent behavior [11]. In other words Orchestration is an automation performed to deploy the elements in a controlled manner [11]. The other functions of orchestration are to:

- To combine necessary architecture, tools and the processes to deliver a service.
- Stitching of software and hardware components together to deliver the defined service
- To connect and automate the workflows relevant to the defined service [12].

The Orchestration becomes necessary to scale-up arbitrarily and dynamically without or less human interventions. Hence it becomes necessary to define the appropriate workflow for smooth delivery of services.

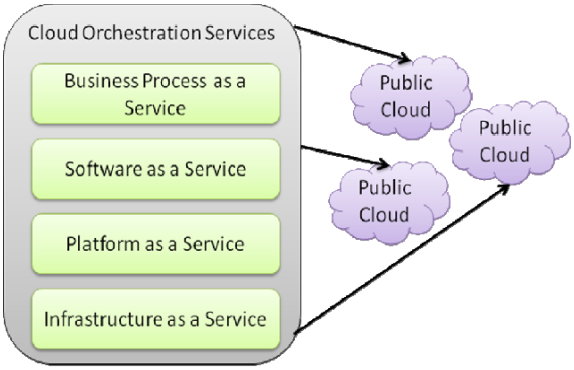

Fig 1: Cloud Orchestration Services [13]

Figure 1, gives a clear idea as to what we can obtain from orchestration services.
To orchestrate, effective communication is essential among the existing services. The technologies should ensure reliable and secured communication without affecting the underlying systems.
Middleware in the modern architecture enables to establish an efficient two way communication. In most of the distributed environment, middleware plays an important role in managing data, communication and interoperability. It supports and streamlines distributed applications including web servers, application servers, messaging servers, messaging and many more similar tools which are involved in application development and also resource delivery. Middleware is integrated with modern technology based on XML, SOAP, Web Services, and Service oriented Architecture. The middleware software is reliable and is easily available. It has the capability to scale. The middleware is mainly classified as

- Message Oriented Middleware (MOM)
- Remote Procedure Calls(RPC)
- Object Request Broker
- Transaction Processing Monitors

MOM is a client/server infrastructure which makes it possible to distribute the application over dissimilar platforms. It hides the complexity from the network protocols and the operating systems. MOM exchanges data either through message passing or message queuing. It is secured and has efficient administrative support. The reason for deciding on MOM in the current work is its asynchronous nature. It is loosely

coupled and is suitable for application integration which is the current need. Its delivery services are reliable as there is persistent queuing of messages. The messages can be filtered and transformed. An efficient network of message servers can be established and the database integration can be obtained. According to researchers and enterprisers, OpenStack is a popular cloud operating system that controls large pools of compute, storage and networking resources. It is a collection [15,16,17] of open-source software projects that enable to setup and run the cloud compute infrastructure. AMQP is the existing messaging technology chosen by the OpenStack cloud. The AMQP broker, Rabbitmq, placed between two Nova components allows communication in a loosely coupled fashion. They use RPC for communication and is built on top of the publish/subscribe model [8]. AMQP uses message queuing concept.

The drawbacks of AMQP are with respect to node identification, group formation and presence indication. These drawbacks can be fixed using XMPP protocol.

Table 1: Comparison between AMQP and XMPP Messaging Protocols

	AMQP	**XMPP**
Nature of the Protocol	Binary wire protocol	XML Based
Communication	Loosely coupled	Loosely coupled
Messaging mode	Message queuing	Message exchanging, real-time
Scalability	Scalable	Highly Scalable
Node Identification	Not Present	Present
Group Formation	Not Present	Present
Presence indication	Not Available	Available
Response Time	Moderate	Quick

The comparison study between AMQP and XMPP is presented in the above table after performing detailed study on both the mechanisms.

AMQP, adopts message queuing and it exchanges messages through RPC. Every time when a message has to be processed, it is through the RPC.

XMPP uses XML stream with a single XML parser instance. Due to its long-living nature, we need not initiate a new XML parser every time when a message has to be processed. This will improve the response time between message-exchanging significantly. Thus, in our work we propose to use XMPP as the method for orchestration. In the subsequent section we explain XMPP in detail and also highlight the features of XMPP.

3.1. XMPP for Cloud Orchestration

In the previous section, it was seen that middleware plays a popular role in distributed environment in provisioning the resources. The discussion also highlights the advantages of MOM. Also we were able to show that XMPP is powerful than AMQP. With this background, we consider XMPP as a better alternative for orchestration. **XMPP** is an acronym for Extensible Messaging and Presence Protocol. It is an open standard protocol, widely used as lightweight middleware. XMPP is well suited for distributed and complex applications. It can also be used for content syndication, collaboration and universal routing of XML data [18].

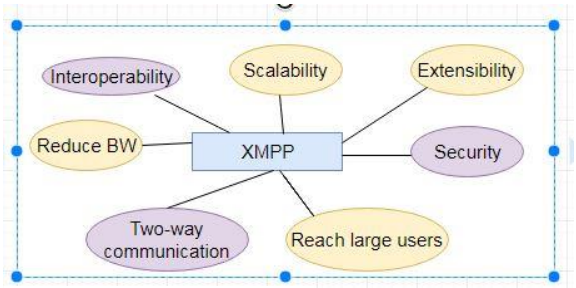

Fig 2: XMPP Features [18].

The Fig 2, depicts the features of XMPP.

3.2. Why XMPP??

- ✓ XMPP is an open standard protocol. IETF Standard RFC 3920.
- ✓ Stateful protocol – Two Way communication.
- ✓ Real-time communication.
- ✓ It can be used for all cloud resources (VMs/Storage).
- ✓ Scalability-XMPP is designed to be scaled. It overcomes the problem of long polling that is found in HTTP-based methods.
- ✓ Decentralised Network-DNS connection established in distributed manner to connect XMPP server with other networks.
- ✓ XMPP has strong user identification and grouping feature.
- ✓ Availability indication through <presence>.
- ✓ Secure and reliable- Improvised features of Transport Layer Security (TLS).
- ✓ Extensible- Responsibility in communicating XML messages from place to place.
- ✓ Flexible and Diverse.
- ✓ Large number of libraries.

XMPP is defined as a streaming protocol that makes it possible to exchange XML fragments between any two

network endpoints [19]. XMPP practices specific payloads in the manner of SOAP to achieve real-time communication [19]. It is an effective protocol for the cloud computing era. Cloud computing and storage systems depend on various stages and forms of communication. The cloud computing is just not about communication, but also deals with migration of larger units/objects such as storage or virtual machines. XMPP can be applied at various levels and is an ideal middleware protocol. It has a strong authentication feature. Applying compression on XML streams either at TLS layer or application layer will reduce the bandwidth usage significantly [26].XMPP involves less of human communication and more of machine communication. With such enormous benefits, XMPP can be considered as a promising protocol to achieve orchestration.

3.2. Cloud Orchestration Model through Instant Messaging – XMPP (IM-XMPP)

The IM-XMPP delivery model has several benefits. It is instantaneous and will include a built-in subscription mechanism and provides an indication of availability through presence. The XMPP architecture includes distributed, decentralized network, servers, JABBER IDs, structured XML data which helps in real time communication, presence information of networks. It is a stateful protocol, which communicates through XML streams. The clients focus on user experience. The servers withstand the complexities of the clients and ensure reliability and performance. The two-way communication helps in overcoming long polling, as the clients need not wait for immediate response of the XMPP server for processing the requests. The clients can send any number of requests to any other entities and get the response later. This mechanism is in contrast with http. Structured XML is used for communication in the form of message streams. JABBER IDENTIFIERS (IDs) are used for addressing. The identifiers can be used to indicate the location of the entity on the XMPP network. XMPP brings up a format for the purpose of data transmission between entities that are willing to communicate.

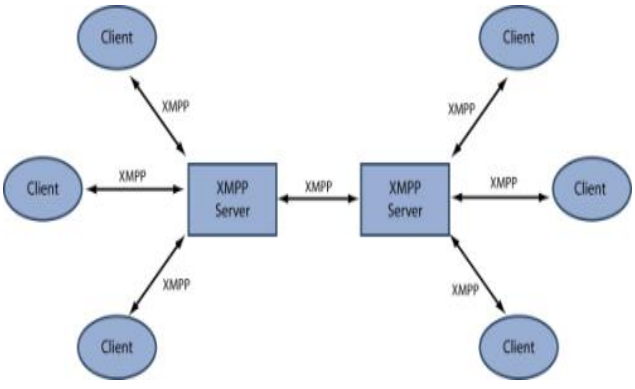

Fig 3: XMPP Architecture [21].

Figure 3, depicts that XMPP server is well connected with clients and also it represents communication between the XMPP servers.

XML stanza is to be considered as a basic unit of XMPP. XML stanza is structured information, which is being transmitted between communicating entities upon an XML stream. These XML stanzas are enclosed within an XML stream envelope. XMPP comprises of three kinds of stanzas namely <message>, <presence> and <iq>.

The <message/>stanza is used for speeding up the communication between entities. <presence> stanza is used for management and reporting. It broadcasts its presence information or availability to other entities. <presence> purpose is for publishing or advertising.

<iq>, Info/Query stanza is based on the request response. IQ stanza contains the following values: **get** or **set** for requesting entity, result or error for responding entity [19].

The above streams permit delivery of stanzas between the clients through the server. The servers' tasks include storing and managing XML data used by the clients and managing the delivery of XML streams to local XMPP clients. The local service policies and protocols take up the responsibility of foreign communication (HTTP). The stream is viewed as one structured XML document permitting the communication. The XMPP core has extensions too, which are useful in further extensibility.

4. Work-Flow Model for Cloud Orchestration through XMPP.

In this section, we propose our own work-flow model for cloud orchestration using XMPP. The user sends its resource requirements to the server through HTTP request. The requests are forwarded to the XMPP clients. The XMPP-Client which aims to launch a new Virtual machine sends the request to XMPP server. The XMPP server internally communicates through XMPP protocol with the resources at the cloud centre. The communication between the XMPP server and the nodes at the datacentre is through the XMPP protocol.

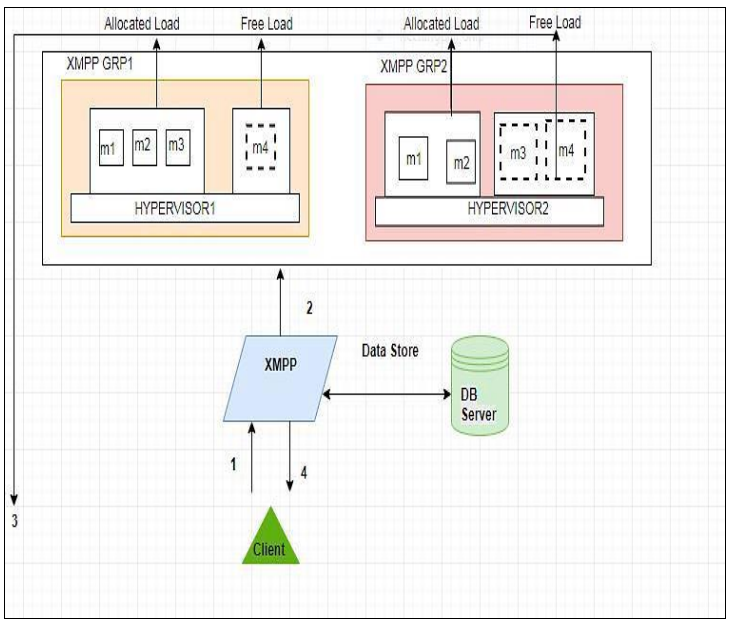

Fig 4: The proposed Cloud Orchestration model using XMPP.

The Cloud Orchestration Model working is shown in the above diagram. The steps involved in the Orchestration are illustrated in section 4.2 below:

4.1 Implementation using Ruby

The XMPP has varied collection of libraries useful for instant messaging. The XMPP libraries allow exploring the development. XMPP libraries can be functional for development among various languages like Ruby, Python, .Net etc. For simplicity and experiment purpose, in the current work Ruby is opted. The features of Ruby are:

- It is object oriented
- XMPP compliant
- Passed unit testing and has documented code.
- It uses software like REXML.
- It is extensible, which is an advantage.
- Dual-Licensing and compatible with GNU.

The installation and server connection establishment is shown below:

```
require 'xmpp4R/client'

#Create a very simple dictionary using a
hash
hash = {}
hash['ruby'] = 'Ruby Script'
hash['xmppr'] = 'XMPP library for ruby'
hash['xmpp'] = 'Extensible Messaging and
Presence Protocol'

#Connect to the server and authenticate
jid =
Jabber::JID::new('grp1@default.rs/Home')
al = Jabber::Client::new(jid)
al.connect
al.auth('password')

#Indicate the presence to the server
al.send Jabber::Presence::new

#Send a hello
= Jabber::Message::new( 'grp2@default.rs',
'grp1 ready' )
hello.set type(:chat).set id('1')
al.send hello

#Add a message callback to respond to peer
requests
al.add message callback do |inmsg|

    #Lookup the word in the dictionary
    resp = hashinmsg.body    if resp ==
nil
                    resp = "ok
```

Fig 5: Connection Establishment with the server.

The Instant Messenger [IM] will send the request through words and the agent will respond. The Instant Message initiation begins with a dictionary. The pattern requires an array or hash to hold key-value pairs. The hash is considered for indexing the next key in this demonstration. The XMPP4R library is selected, as it provides a framework to develop Jabber applications. The threads are event based and are easily extendible. The XMPP4R is installed from the command prompt.

"$ gem install xmpp4r"

After successful installation of XMPP library, the connection with the server is established. A new JID and a new client connection are established.
The connect method is used to connect with the server. With the establishment of the connection, "auth" method is called to authenticate. Once the authentication is valid, it is ready for messaging.
Once the authorisation is established, the subsequent step is to let the server know about the presence to the IM server. To perform this action, the presence stanza is sent to the server. The peer nodes can also be indicated, but this optional. A message stanza is created and initialized with the address and a message. Once the message is ready, it is sent using the *client* class instance and the *send* method is used to send the message.

4.2. Steps for the proposed work-flow represented in Fig 4.

Step 1: Client requests XMPP for new VM.

With the establishment of connection with the server, the client which aims to procure the Virtual Machine(VM), will send a request to XMPP server through the <iq> stanza indicated as"1" in Figure 4. The below stream illustrates <iq> stanza.

```
<iq>
<type>machine</type>
<size>medium</size>
<cpu>2</cpu>
```

Fig 6: <iq> Stanza indicating the requests.

The client will indicate its requirements to the XMPP server through <iq> stanza. Figure 6 <iq> stanza which indicates that the client requires three machines of medium size with two core CPUs for its launch of a new VM. The stanza uses simple and structured XML language for communication. The datacenters, which intend to procure the resources, establish the communication with the XMPP server.

4.3. Step 2: XMPP will send message to XMPP Group

The XMPP server receives the client's request and forwards them to its connected groups/peers. This is indicated as "2" in Figure 4. The XMPP group formation is mentioned in sections 4.3.1, 4.3.2, and 4.3.3. The XMPP server communicates with the groups that have been identified through JABBER IDENTIFIERS (IDs).

```
<message
group_id="grp1@cloudcenter.net>
request <iq>
</message>
```

Fig 7: <message> Stanza communicating with group.

The Hypervisors of the groups will identify their presence. The <iq> stanza is copied into the group hypervisors, which will hold the information requested by the client. The XMPP server initiates the group message. At this stage, replication happens through XMPP groups. Appropriate classification of the groups will reduce the burden on the XMPP server. This will also help in achieving the load balancing. The group is addressed through group-id. The proposed group classification is as follows:

4.3.1 Location based group formation

The XMPP server will communicate with the group based on the geographical region or location. Suppose an American client requests for the resource, the XMPP server will identify the group associated to that region. It will communicate only to the American group's datacenter and not the other group's datacentre. This will improve the response time. Smart machine learning algorithms will assist the XMPP server to identify the group and will reduce the delay in communication and thus speed up the response. The example of <message> stanza is shown in Figure 7. The hypervisors will respond through the <presence>.

4.3.2 Group on the basis of load

The next group classification is based on the load. There are many applications across the global networks which are in high demand. When the demand is high, the load is at its peak. The applications with high demand require constant provisioning of resources. Hence there are dedicated datacenters to provision their needs.

There are also certain applications which work with off-peak load applications. The hypervisors with fewer loads will respond quickly to the XMPP request for new VM and form a group. These off-peak groups can suffice the resources and hence the XMPP server will identify these groups and forward the <message> shown in Figure 7. The hypervisors will respond through the <presence>.

4.3.3 Group on the basis of type of instance

The groups are formed based on the type of instance. The type of instances generally comprises of memory, networking capacity, CPU, storage. These instances allow in selecting the combinations of resources for the applications. The instances can be general purpose, small, medium and large. By the pattern of the request made, the XMPP will decide to send the <iq>. If the desired resources are small, it will send the <iq> for smaller groups. Similarly, when the desired patterns of resources are huge it forms a group as large and sends the <iq> to those groups. The hypervisors will respond through the <presence>.

The full load instance types are deployed on clusters using c4.2X large with 9 nodes and c4.2 X large instances with 4 nodes [24]. In the real world scenario, in order to replicate extra nodes are provisioned.

4.4 Step 3: The XMPP Group will respond through its current status

The XMPP group hypervisor will perform its action in identifying the requested resource. It will respond back to the XMPP server through its <message> stanza. The response phase is indicated as "3" in Figure 4. It receives responses from all the hypervisors which reacted with <presence> stanza. The hypervisor's provide their current availability status.

Figure 8, indicates the Hypervisor1's response with its current availability status. It indicates that it has "one" machine available against the client's request for "three" machines.

```
<message>
<norem>1</norem>
<status>
<current status>
</status>
```

Fig 8: Status message from hypervisor1.

The Hypervisor2 also responds with its current availability status.

```
<message>
<norem>2</norem>
<status>
```

Fig 9: Status message from hypervisor2

This hypervisor's 2 status indicates that it has "two" machines currently available and is ready for allocation. Likewise all the hypervisors connected with the XMPP will respond with their respective statuses.

4.5. Step 4: Resource Allocation

The XMPP server accumulates all the responses received from the group hypervisors. It verifies and views the actual request from the client and allocates the required amount of resource to the client. It will store the complete information on the data storage. This step is indicated as "4" in Figure 4.

```
<iq>
<type>ok</type>
<status>
<machine>m</machine>
</iq>
```

Fig 10: Ready for allocation.

The <m> stanza will include the necessary information related to the machine as to its IP address, CPU-id, type of OS, Memory shared or unshared etc.

```
<m>
<ip>---</ip>
<cpu_id>---</cpu_id>
<os>---</os>
<mem>----</mem>
<shared>Y/N</shared>
</m>
```

Fig 11: Machine related information.

The final step includes all the necessary information related to the machine and XMPP will launch a new VM dynamically. The XMPP uses structured XML fragments as presented in the above steps for communication between any two nodes. The dynamic group formation according to the classification specified has to be achieved through the advanced Machine-Learning algorithms like Decision-Tree, K-Means Clustering etc. Since XMPP uses dynamic group messaging model, it will reduce unnecessary message exchanging between any two nodes. This will reduce the total number of CPU's required for processing at least by 30%. We can also see a significant improvisation in response time and resources can be allocated effectively.

5. Conclusion.

In our work we have suggested a conceptual framework for Cloud Orchestration through XMPP. This framework is a new dimension for cloud orchestration. The system is framed to meet the client's requirement to launch a new virtual machine (VM). AMQP and XMPP are communication standards between any two nodes. With our comparative study, we state that XMPP overpowers AMQP. With several benefits of XMPP we state that it can be a better alternative and can set a new dimension for Cloud Orchestration.

The framework has several benefits and is stated below:

- Orchestration through XMPP enables the enterprises to manage their own cloud datacentre efficiently without the involvement of commercial clouds.
- The three major features namely node identification, dynamic group formation and presence indication of XMPP, enables it to be a better approach for real-time communication.
- The traditional methods in requesting and responding were only HTTP based. In our framework, the request for resource allocation is through HTTP initially. Later the requests are managed by XMPP servers, which do not demand the client's presence continuously. The XMPP resolves the long polling as the clients need not wait for immediate response of the XMPP server for processing the requests.
- We have illustrated in figures 6 through 11 that XML fragments are capable of communicating with the group hypervisors. The workflow model has depicted proper group identification, which also enables to balance the workload instances. The XMPP server actively pushes messages, instead of client requesting and responding. This feature helps in overcoming the long polling.
- XMPP is less human interventional and thus can behave intelligently. The cloud orchestration will make the system more intelligent and extendible.
- The total number of CPU's required for processing is reduced by at least 30%, as the communication between the groups is well-defined. This will significantly improve system's performance.

Further, the framework has to be implemented in a real-time scenario and the results have to be observed and compared with our study.

Acknowledgements: I acknowledge **Bharathiar University, Coimbatore** for providing me an opportunity to carry out research in the field of computer science.

My Sincere gratitude to **Mr. Prashanth Raghu,** Technical Architect, Sen Sei Technologies, Bangalore, who gave valuable inputs to carry out this work. This framework conceived after several technical discussions.

References

[1] Michael Armbrust, Armando Fox, Gunho Lee, Ion Stoica "Above the Clouds: A Berkeley View of Cloud Computing" University of California at Berkeley Technical Report No. UCB/EECS2009-28, 2009.

[2] Bourguiba, M.; Haddadou, K.; EI Korbi, I.; Pujolle, G., "Improving Network VO Virtualization for Cloud computing," Parallel and Distributed Systems, IEEE Transactions on, vo1.25, no.3, pp.673- 681, March 2014.

[3] Weiwei Lin, BaoyunPeng, et al. " Novel Resource Allocation Model and Algorithms for Cloud", IEEE Xplore, Fourth International Conference on Emerging Intelligent Data and Web Technologies, DOI:10.1109, October , 2013

[4] L. Badger, R. Patt-Corner, J. Voas, Cloud Computing Synopsis and Recommendations, NIST Special Publication 800-146, May 2012.

[5] Gu. Galante, L.Carlos E. de Bona," Survey on Cloud Computing Elasticity", IEEE/ACM 5th International Conference on Utility and Cloud Computing, UCC, 2012

[6] Ficco, M., Esposito, C., Palmieri,, et.al, "Smart Cloud Storage Service Selection Based on Fuzzy Logic, Theory of Evidence and Game Theory", IEEE Transactions on Computers In Press, 2015

[7] Xiaolong Wen1 , Genqiang Gu1, et.al. "Comparison of Open-Source Cloud Management Platforms: OpenStack and OpenNebula", 9th International Conference on Fuzzy Systems and Knowledge Discovery, 2012.

[8] https://docs.openstack.org/nova/queens/reference/rpc.html

[9] https://searchdata-centre.techtarget

[10] Changbin Liu, Yun Mao, "Cloud Resource Orchestration: "A Data-

Centric Approach", Proceedings of the biennial, 2011.

[11] Thomas Erl. Service-Oriented Architecture: Concepts, Technology & Design. Prentice Hall, ISBN 0-13- 185858-0

[12] Wikipedia https://en.wikipedia.org/wiki/Orchestration.

[13] Bhaskar Prasad Rimal, Md. A. EJ-Refary, "A-Framework-of-Scientific-Workflow-Management", 19th IEEE International Workshops on Enabling Technologies, 2010.

[14] https://searchmicroservices.techtarget.com/definition/middleware

[15] OpenStack Home Page. http://www.openstack.org/

[16] I. Voras, B. Mihaljecić, M. Orlić, et al, "Evaluating Open-Source Cloud Computing Solutions", MIPRO, Proceedings of the 34th International Convention, pp. 209-214, 2011.

[17] M. Mahjoub, A. Mdhaffar, et al. "A Comparative Study of the Current Cloud Computing Technologies and Offers," IEEE First Symposium on Network, 2011

[18] XMPP: An overview, https://xmpp.org/about/technology-overview.html

[19] https://searchmicroservices.techtarget.com/tip/XMPP

[20] http://www.gyanvihar.org/centrallibrary

[21] https://www.isode.com/img/xmpp1.png

[22] https://developer.ibm.com/tutorials/

[23] https://cloud.google.com/about/locations/

[24] https://tools.ietf.org/html/rfc6121

[25] Roy T. Fielding, "Architectural Styles and the Design of Network-based Software Architecture", Doctoral dissertation, 2000

[26] https://stpeter.im/journal/1171.html