International Journal of Applied Engineering Research ISSN 0973-4562 Volume 12, Number 24 (2017) pp. 15723-15728

Cloud Micro Services Architecture for Portable Workloads using Container Technologies and Standards

Suresh Durairajan [1]
Research Scholar, Department of Computer Applications,
Dr. M.G.R. Educational & Research Institute University, Chennai, Tamilnadu, India.

Orcid Id: 0000-0001-9011-5589

Viji Vinod [2]
Professor, Department of Computer Applications,
Dr. M.G.R. Educational & Research Institute University, Chennai, Tamilnadu, India.

Abstract

Traditional software models build applications as a single and unified unit, where every feature and function is contained within one or other executable or active codebase. These applications by design are tightly integrated leading to components inter dependency and application centralization. Traditional applications are closely tightened with the resources. It limits the scalability, especially with large enterprise-class applications, whose capabilities are finite. This is due to the vertical scaling design of the application. Micro Services Architecture help the applications to remain consistent even with the increased load and the change in codes happen regularly. It should be able to discourse the scalability of the application and should be able to run on any platform irrespective of the infrastructure.

Keywords: TOSCA, ITIL, Micro Services, Cloud, Containers, Docker, PaaS, UniKernels, MiniOS.

INTRODUCTION

Traditional IT Architecture lies over three tier Monolithic model with three layers: Presentation, Application and Database layers. These traditional architectures are simple at first, single codebase or deploy unit, resource efficient at small scale, with in-process latencies. But as it grows the overhead increases, which leads to poor scaling, all-or-nothing deploy unit, and poor enforcement of modularity along with long build times.

The Client-server model used in 1980s was widely accepted as it was a cost-effective option for the organizations to spend on the systems as per the need. This is because the system chosen for server and for client could be configured separately as per the requirement. This helped in increasing the performance without much efforts and without affecting user applications. If the geographical area covered is small and need cost effective system, it is good to go for client/server computing.

From early 2000's, Service Oriented Architecture (SOA) started attraction in the market. SOA [1] is a strategic design pattern separating one big application into different application functionality as services, thereby allowing the enterprise to be more flexible. The pros of SOA are that it is reusable, platform independent, can provide service globally, can provide improved scalability and availability, and can increase productivity. It also includes increased overhead due to frequent service interactions with complete validation of the request. This leads to complex service management due to accurate message delivery and to overcome challenges with managing a huge population of services due to high data inflow as shown in figure 1.

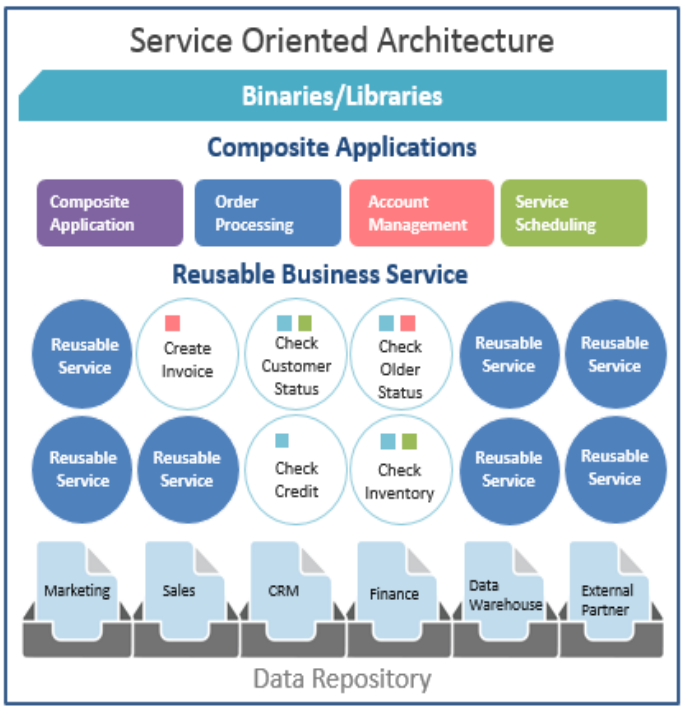

Figure 1: Service Oriented Architecture

Technology Transformation to Micro Services Architecture (MSA)

Micro Services Architecture [2] [3]is a distinctive method of developing software Applications as suites of independently deployable services. In this style, complex applications are composed of small, independent processes communicating with each other using API's. These services are small building blocks, highly decoupled and intensive on doing a small task, facilitating a modular approach to system building. So, in simple, MSA Applications are set of narrowly focused, independently deployable services as shown in figure 2.

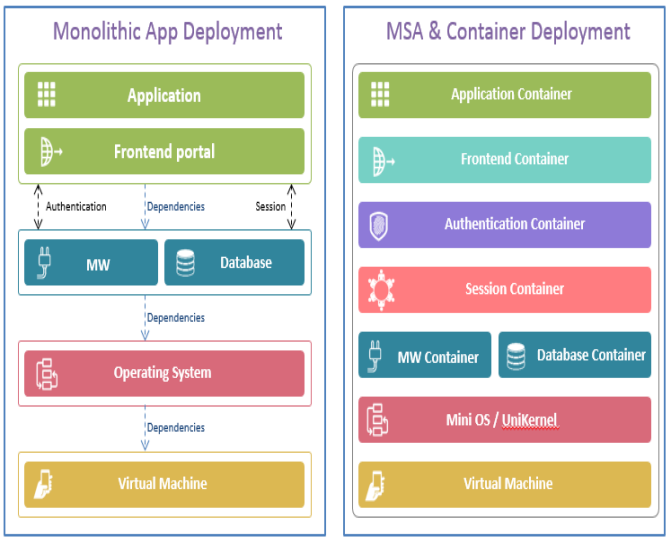

Figure 2: Technology helps Transformation

Companies which are having high volume portals started performing revamp of their IT strategy due to their legacy architectures. This was as it did not allow to add new or change old functionality rapidly. Hence these companies started adopting loosely coupled approach based on MSA. The major goal of these revamp is to eliminate dependency and to enable quick testing and deployment of code changes.

An MSA would also provide the services layer an emerging cloud-inspired, enterprise integration fabric. Once the application becomes platform independent, it will be able to run on any vendor platform or Cloud without effecting the application functionality. This is due to its small codes and its flexibility to enlarge based on the requirement. Emerging technologies like Containers help us to achieve the same.

CONTAINER TECHNOLOGY

Software containers are poised to do the same thing that the shipping containers does in logistics. They make it easier to move applications from one platform to another. In simple terms, container [4] consists of an entire runtime environment: An application plus all its dependencies like libraries, binaries, and configuration files required to run it are bundled into one package. By containerizing the application platform and its dependencies, differences in OS distributions and underlying infrastructure are abstracted away as shown in figure 3. The two revolutionary technologies which are most closely associated with container technology are: **Docker** and **Flocker**.

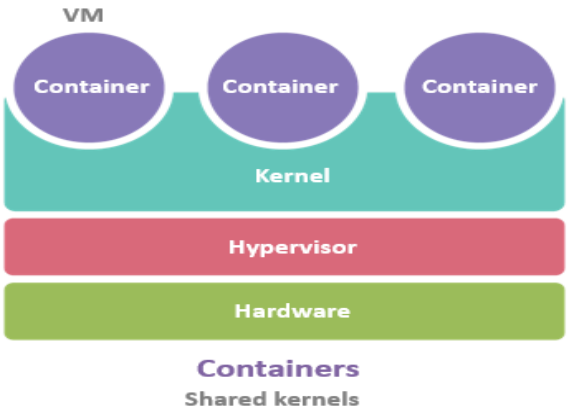

Figure 3: Containers

Docker: Docker [5] implements a high-level API to provide light weight containers that run processes in isolation. This relies on the kernel's functionality and uses resource isolation to isolate the application's view of the operating system, like CPU, memory, I/O, network, etc.

Flocker: Flocker is an open-source container data volume manager for the Docker based applications which supports block-based shared storage. Flocker manages Docker containers and data volumes together. Whenever a Docker application is shuttled between hosts on a given cluster ensure they move jointly for easy management of both application and data. Flocker helps us to deploy stateful services like databases, key-value stores and queues with containers.

The observation of Analysts like Gartner Hype cycle and Technology adoption lifecycle convey that, Containers are already in the peak status of expectation and the Micro Services is gaining the momentum of hype across the IT industry. IT Giants are already working closely on MSA models to transform their clients. Few of the clients like ING [6], Netflix already moved their applications onto MSA and taking those benefits.

Mini OS Revolution

Mini-OS is a tiny OS kernel distribution. These minimalist OS's are designed to host Docker applications and simplify the infrastructure. To have better scalability, flexibility, high availability, highly secured, good reliability its best to have Containers and MiniOS go together because it will become multi-host full replacement for enterprise type apps.

Nowadays, almost all modern Linux distributions have

officially supported container install paths. They come with thousands of pre-packaged software components that can run alongside the containers with minimal friction. The increased focus on stability and security of these Mini-OS systems is appealing, and the trade-off of losing traditional packaging systems will become less relevant as software is increasingly working on MSA's as shown in figure 4. OS releases like Core OS [7], Project Atomic, Snappy Ubantu, Rancher OS, Photon, Nano Servers, MicroVM comes under Mini-OS technologies.

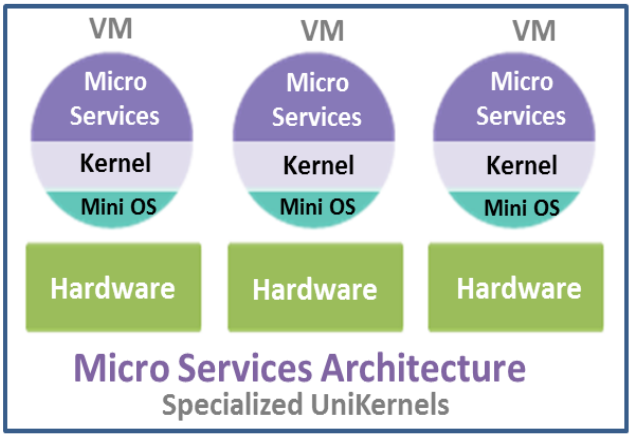

Figure 4: Uni Kernels

Boost of Uni Kernels

Uni Kernels [10] compile the source code into a custom operating system that includes only the functionality required by the application logic. That makes them small, fast, and improves efficiency. By reducing the amount of code deployed at OS level, Uni Kernels necessarily reduce the data or user attacks and therefore have improved security properties as shown in figure 5.

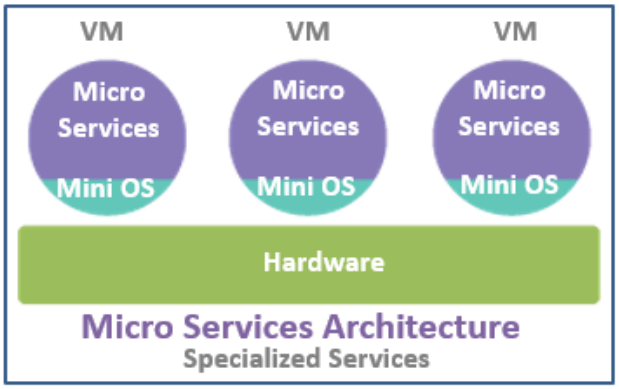

Figure 5: Mini OS

The small footprint of Uni Kernels helps in reducing the size which in turn helps in low boot times. These benefits lends in creating systems that follow the MSA's by creating relevance in developing the concept of immutable infrastructure, where modification in the machine isn't directly permitted whereas all changes are made to the source code itself. Uni Kernels are able to deliver easy deployment, impressive flexibility, speed and versatility for cross-platform environments, big data analytics and scale-out cloud computing. There are a number of new approaches in constructing Uni Kernels few of the listing are ClickOS, Clive, MirageOS, Runtime.js etc.,

ARCHITECTURE FOR NEXT GENERATION INFRASTRUCTURE

Containers, Mini-OS & Uni Kernels exist in a twilight zone somewhere between hosts and applications where neither application performance monitoring nor traditional infrastructure monitoring are effective [8]. This creates a blind spot in monitoring, where in the product based companies started adopting their development in them. In the present world, we are monitoring Operating System and its libraries but tomorrow our focus would move towards Containers, Mini-OS, Uni Kernels etc., So it's important to define the metrics for Containers so that the operational readiness for support will be in-place. There are few tools in the market like CAdvisor, Scout, Data dog, Sensu monitoring, etc., which can help in monitoring Containers.

An MSA provides well-defined loosely coupled approach by eliminating most of the dependencies and providing software component as a suite for independent deployment modules. In order to make Micro Services to work independently, functional module dependencies and Operating System Kernel are to be wrapped with the functional components of the architecture [3]. In a way, any software application and its components, which wants to reduce complexity, increase reliability, and increase efficiency should be designed as a Micro Service.

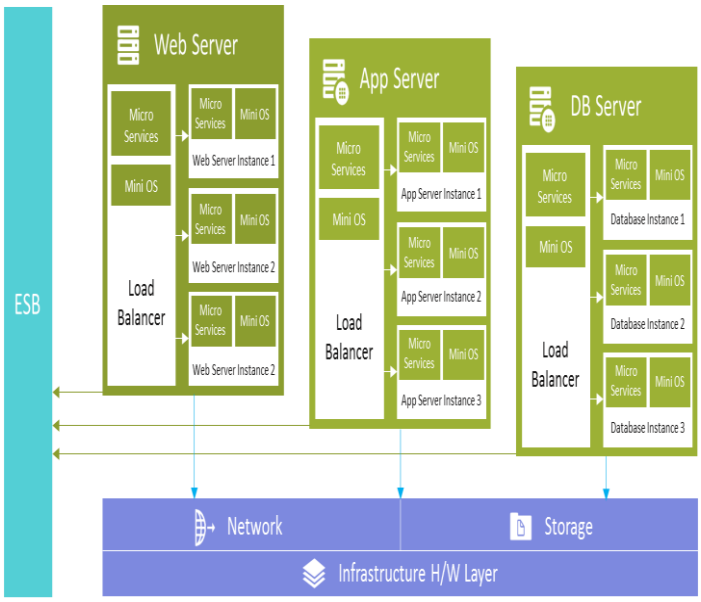

Figure 6: Inner Architecture of MSA

Figure 6, is a sample for how an inner architecture of Micro Service looks like in common multi-tier application environment. In Webserver and Appserver tiers one can achieve important aspect of scalability by spinning multiple instance in horizontal way. This can be done by imparting Micro Service feature to its components like Load Balance, Web server instances and Appserver instances. An MSA helps you to design functional components for high availability and manage flexibility by making component totally independent. Necessary security features can be achieved by OS functions which are wrapped with that Micro Services.

Unlike Web and App tiers, DB instance requires state full session connection in order to make data persistent, which will be accomplished using Database containerization technology like Flocker, to communicate between Micro Services and DB services. By this approach, data volume also can be modularized and moved between multiple hosts without much difficulty to achieve high availability and to manage independently as shown in figure 7.

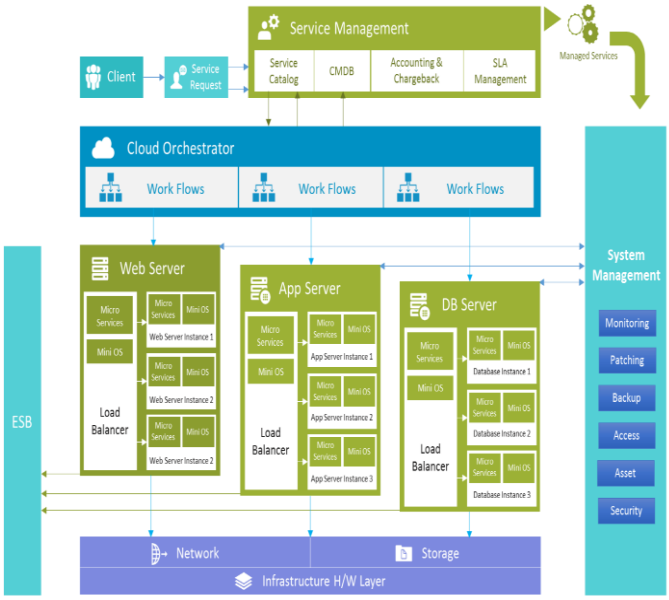

Figure 7: Outer Architecture of MSA

As Containers and Mini-OS include only those dependencies that the application requires there is a decrease in the vulnerability surface of the application's environment and it makes it easier to lock it down. The smaller footprint also decreases the number of components that need to be patched with security updates.

- Security patches are installed on the system independently of the application.
- Containers integrate the app with dependencies more tightly and allow for the container's image to be patched as part of the application deployment process.
- Rebuilding the container's image allows the application's dependencies to be automatically updated. [5]
- The container ecosystem changes the work that ops might traditionally perform, but that is not necessarily a bad thing.

The security management of the containers is handled at Mini-OS level for which monitoring is also necessary and the tools like Clair etc., inspects containers layer-by-layer for known security flaws.

Using containers to keep systems up to date makes the process of patching a server nearly obsolete. Of course, the base system will need to be kept up to date as well, but since it should be inaccessible from the Internet, the attack vector is much smaller. The clustering option makes them to patch the base system without effecting the containers and without any downtime or outages.

Cloud Orchestration and Service Management

The future of Service Management [11] is being able to manage the services and their dependencies directly, and not by inference. The more these are self-contained, the more the job of management is simpler. Time spent messing with kernel configurations and installing software has dropped sharply as the systems started to abstract at a higher level and to use smaller, reusable bits.

Infrastructure as a code: The future

YAML Coding Prototype:

DBServer_Data:
ImageName=LabServer_Data
ContainerName=LabServer_Data
DBServer:
Image_Name=DBServer:Latest
Container_Name=DBServer:Demo_Server
HostName=DB_Server
Port:
- *Source_PortNo:Target_PortNo*

Environment:
- *Variable_Account=Service_Account*

Volume_From:
- *FileSystem:WriteMode*

Application_Server:
Image_Name=Application:Latest
Container_Name=Application:Demo_Server
HostName=Application_Server
Port:
- *Source_PortNo:Target_PortNo*
- *Source_PortNo:Target_PortNo*

Environment:

- *Variable_Account=Service_Account*
- *Variable_Password=Service_Password*

Volume_From:

- *FileSystem:WriteMode*

Links:

- *Refer_Tag_Defined*

Web_Server:

Image_Name=WebServer:Latest

Container_Name=WebServer:Demo_Server

HostName=Web_Server

Port:

- *Source_PortNo:Target_PortNo*

Environment:

- *Variable_Account=Service_Account*
- *Variable_Password=Service_Password*

Volume_From:

- *FileSystem:WriteMode*

Links:

- *Refer_Tag_Defined*

Because of the architectural change on the application design and the management, the service management model should be equipped to adapt the Micro Services technology. It is very necessary to start with modeling the services and manage those as an early concern, backing it with adequate tooling in place. The pressure of the dynamic nature of the Micro Services has meant that actual service dependencies have to be modeled and the respective tools [12] should also be ready for such customizations.

Cost Analysis for Micro services against Server based workloads

On-Premises hosting is done locally on the company's own computers and servers. On-Premises hosting involves large initial capital expenditure in the form setting up servers, storage and network. Recurring expenses include fees for support, training and updates.

Cloud based hosting is done on vendor's servers and accessed through a web browser. Expenditure for cloud based hosting is subscription based either monthly or annual. Due to low entry cost of cloud based hosting [9] compared to hefty initial expenses for on-premises hosting cloud based hosting has been adopted throughout the world. Container hosting is the future technology when compared to cloud based hosting. Containers make use of Operating system virtualization concept. Containers can make isolated systems run on a single server or host OS. When compared to cloud based hosting we can save anywhere between 35-40% in container hosting as shown in figure 8.

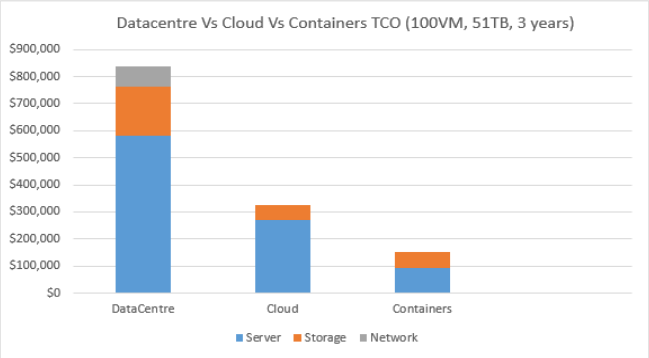

Figure 8: Cost Analysis of Data Centre vs Cloud vs Containers

RELATED WORK AND SUMMARY

Researchers [8], [11 - 13] present a discussion of automating ITIL service management processes in a cloud environment to support managed production workloads. The shift away from operating system-level virtualization and toward application virtualization is one of the most disruptive developments to emerge as shown in figure 9.

Figure 9: Micro Services for LAMP Stack

Micro Services and Container frameworks are up-and-coming technologies that can greatly improve software delivery for large companies and regulated enterprises, provided they are well managed and implemented with enterprise requirements in mind. The real power of this container approach is that containers are scalable; if a blockage occurs in one Container, development team can deploy more iterations of that Container alone, to handle the load by effectively utilizing the resources with less infrastructure resources. Also because of this blockage other containers will not be effected nor the total application as shown in figure 10.

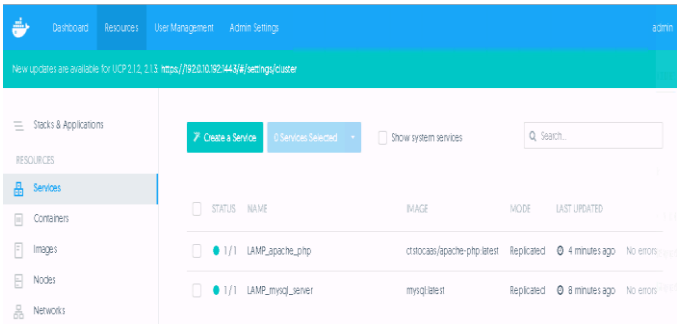

Figure 10: Micro Services on Cloud

Hence, Organizations should deploy a loosely coupled Micro Services Architecture which helps in moving from monolithic app to multiple smaller apps to avoid the complexity and interdependency. This gives the freedom for each component to run on its own without any dependency on others and the freedom to communicate with other component whenever required. It would decrease the complexity of the application management with efficiency and consistency.

The truth is that building an app using a Micro Services Architecture would need a lot of design changes and doing things in diverse ways. The application needs an MSA for application decentralization, efficiency, stability, scalability and for being platform independent.

This reduces dependency on the platform and enables resources to concentrate on other productive jobs. The products like Docker, CoreOS Server, Micro VM, Mesos etc., are already in market to support MSA based application models so it is very important for the organizations to start utilizing this for their larger growth of business and better management of their applications and infrastructure.

REFERENCES

[1] Barry&Associates, 2000-2017 “Service Oriented Architecture” http://www.service-architecture.com/articles/web-services/service-oriented_architecture_soa_definition.html,

[2] Sam Newman, 2015 “Micro Services Architecture“ O’Reilly media http://www.oreilly.com/catalog/errala.csp?isbn=9781491950357, ISBN 971491950357,

[3] Dmitry Nomoit, Manfred Sneps-Sneppe, 2014 “On Micro Services Architecture” International Journal of Open Information Technologies ISSN:2307-8162 vol. 2, No 9,

[4] Mohamed Mohamed, Belaid Samir Tata, 2013 “ Self-Managed Micro-Containers for Service-Based Applications in the Cloud” IEEE WETICE 22nd International Workshop, ISBN 978-1-4799-0405-1,

[5] David Bernstien, 2014 “Containers and Cloud: From LXC to Docker to Kubernetes” In IEEE Cloud Computing http://ieeexplore.ieee.org/abstract/document/7036275/, IEEE

[6] Henk Kolk, 2015 “CS_ING “ https://www.docker.com/sites/default/files/CS_ING_01.25.2015_1.pdf, Docker.com,

[7] J Ellingwood, 2014 “An Introduction to CoreOS Systems and Components” https://www.digitalocean.com/community/tutorials/an-introduction-to-coreos-system-components,

[8] Stephen Watt, 2005"Enhance IT Infrastructure Library service management capabilities" http://www.ibm.com/developerworks/autonomic/library/wsitil/index.html. IBM Corp,

[9] Amazon TCO Calculator, http:// https://awstcocalculator.com/

[10] A Braterud, AA Walla, H Hugirud Paal E Engulsad, Kyrre Begnum, 2015 “IncludeOS: A minimal, resource efficent UniKernal for cloud services, http://ieeexplore.ieee.org/abstract/document/7396164/ Cloud Computing Technology and Science (CloudCom), 2015 IEEE 7TH International Conference, ISBN 978-1-4673-9560-1,

[11] Chieu, Trieu C., Manas Singh, Chunqiang Tang, Mahesh Viswanathan, and Ashu Gupta. 2012 "Automation System for Validation of Configuration and Security Compliance in Managed Cloud Services." In *e-Business Engineering (ICEBE), 2012 IEEE Ninth International Conference on*, pp. 285-291. IEEE,

[12] Head, Michael R., Anca Sailer, Hidayatullah Shaikh, and Mahesh Viswanathan. 2009, "Taking it management services to a cloud." In Cloud Computing, 2009. CLOUD'09. IEEE International Conference on, pp. 175-182. IEEE

[13] Rajkumar, N., Viji Vinod 2015, Integrated Educational Information Systems for Disabled Schools via a Service Bus using SOA, Indian Journal of Science and Technology, Vol. 8. No.13, pp. 1-7