ISSN:0975-9646

Service-Oriented Architecture for Cloud Computing

V.E.Unnamalai, J.R.Thresphine

Department of Computer Science and Engineering,
PRIST University Pondicherry, India.

***Abstract*----- Cloud computing is a significant advancement in the delivery of information technology and services. Cloud computing builds off a foundation of technologies such as grid computing, which includes clustering, server virtualization and dynamic provisioning, as well as SOA shared services and large-scale management automation. The technological improvement believe that clouds, service grids, and service oriented architectures having an Outside-In architecture style are technologies that will be fundamental to successfully making such corporate transformations.**

***Keywords*--- Cloud Computing, Service-oriented Architecture.**

I. INTRODUCTION

Cloud computing has gained a lot of hype in the current world of Information Technology. Cloud computing is said to be the next big thing in the computer world after the internet. While cloud computing is currently a term without a single consensus meaning in the marketplace, it describes a broad movement toward the use of wide area networks, such as the Internet, to enable interaction between IT service providers of many types and consumers.

II. CLOUD COMPUTING

A Cloud is a type of parallel and distributed system consisting of a collection of interconnected and virtualized computers that are dynamically provisioned and presented as one or more unified computing resources based on service-level agreements established through negotiation between the service provider and consumers in [2].

Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.This definition from the National Institute of Standards1 has gained broad support from the industry.

The NIST definition of cloud computing describes five essential characteristics, three service models and four deployment models

A. Five Essential Characteristics

1) *On-demand self service:* Users are able to provision, monitor and manage computing resources as needed without the help of human administrators
2) *Broad network access:* Computing services are delivered over standard networks and heterogeneous devices
3) *Rapid elasticity:* IT resources are able to scale out and in quickly and on an as needed basis
4) *Resource pooling:* IT resources are shared across multiple applications and tenants in a non-dedicated manner
5) *Measured service:* IT resource utilization is tracked for each application and tenant, typically public cloud billing.

B. Three Service Models

1) *Software as a Service (SaaS)***:** Applications delivered as a service to end-users typically through a Web browser. There are hundreds of SaaS service offerings available today, ranging from horizontal enterprise applications to specialized applications for specific industries, and also consumer applications such as Web-based email.
2) *Platform as a Service (PaaS)***:** An application development and deployment platform delivered as a service to developers who use the platform to build, deploy and manage SaaS applications. A virtualized and clustered grid computing architecture is often the basis for PaaS offerings, because grid provides the necessary elastic scalability and resource pooling.
3) *Infrastructure as a Service (IaaS)***:** Computer servers, storage, and networking hardware delivered as a service. This infrastructure hardware is often virtualized, so virtualization, management and operating system software are also part of IaaS as well.

C. Four Deployment Models

1) *Public:* Services and resources are reachable to the public by using the internet. This environment emphasizes the advantages of rationalization (as a user has the ability to utilize only the needed services and pay only for their use), operational simplicity (as the system is organized and hosted by a third party) and scalability. The main concern in this type of cloud environment is the security; since this environment is accessible to the public and user data in one stage is hosted by a third party.

2) *Private:* Services and resources are reachable within a private institute. This environment emphasizes the advantages of integration, optimization of hardware deals and scalability. The main concern is the complexity, as this environment is organized and hosted by internal resources. Security is not a main issue compared to the public cloud as the services are reachable only through private and internal networks.

3) *Community:* Services and resources of this type are shared by various institutes with a common aim. It may be organized by one of the institutes or a third party.

4) *Hybrid:* This type combines the methods from the private and public clouds, where resources can be used either in a public or a private cloud environment. The advantages and the concerns are a mixture of the earlier type.

III. HIGH-LEVEL MARKET-ORIENTED CLOUD ARCHITECTURE.

The high-level architecture for supporting market-oriented resource allocation in Data Centers and Clouds[1]. There are basically four main entities involved are

A. Users/Brokers

Users or brokers acting on their behalf submit service requests from anywhere in the world to the Data Center and Cloud to be processed.

B. SLA Resource Allocator

The SLA Resource Allocator acts as the interface between the Data Center/Cloud service provider and external users/brokers. It requires the interaction of the following mechanisms to support SLA-oriented resource management are

1) *Service Request Examiner and Admission Control*: When a service request is first submitted, the Service Request Examiner and Admission Control mechanism interprets the submitted request for QoS requirements before determining whether to accept or reject the request. Thus, it ensures that there is no overloading of resources whereby many service requests cannot be fulfilled successfully due to limited resources available. It also needs the latest status information regarding resource availability (from VM Monitor mechanism) and workload processing (from Service Request Monitor mechanism) in order to make resource allocation decisions effectively. Then, it assigns requests to VMs and determines resource entitlements for allocated VMs.*Pricing*: The Pricing mechanism decides how service requests are charged. For instance, requests can be charged based on submission time (peak/off-peak), pricing rates (fixed/changing) or availability of resources (supply/demand).
2) *Pricing:* serves as a basis for managing the supply and demand of computing resources within the Data Center and facilitates in prioritizing resource allocations effectively.
3) *Accounting:* The Accounting mechanism maintains the actual usage of resources by requests so that the final cost can be computed and charged to the users. In addition, the maintained historical usage information can be utilized by the Service Request Examiner and Admission Control mechanism to improve resource allocation decisions.*VM Monitor*: The VM Monitor mechanism keeps track of the availability of VMs and their resource entitlements.
4) *Dispatcher*: The Dispatcher mechanism starts the execution of accepted service requests on allocated VMs.
5) *Service Request Monitor*: The Service Request Monitor mechanism keeps track of the execution progress of service requests.

C. VMs

Multiple VMs can be started and stopped dynamically on a single physical machine to meet accepted service requests, hence providing maximum flexibility to configure various partitions of resources on the same physical machine to different specific requirements of service requests. In addition, multiple VMs can concurrently run applications based on different operating system environments on a single physical machine since every VM is completely isolated from one another on the same physical machine.

D. Physical Machines

The Data Center comprises multiple computing servers that provide resources to meet service demands.

In the case of a Cloud as a commercial offering to enable crucial business operations of companies, there are critical QoS parameters to consider in a service request, such as time, cost, reliability and trust/security. In particular, QoS requirements cannot be static and need to be dynamically updated over time due to continuing changes in business operations and operating environments. In short, there should be greater importance on customers since they pay for accessing services in Clouds. In addition, the state-of-the-art in Cloud computing has no or limited support for dynamic negotiation of SLAs between participants and mechanisms for automatic allocation of resources to multiple competing requests. Recently, we have developed negotiation mechanisms based on alternate offers protocol for establishing SLAs [8]. These have high potential for their adoption in Cloud computing systems built using VMs.

Commercial offerings of market-oriented Clouds must be able to:

- support customer-driven service management based on customer profiles and requested service requirements,
- define computational risk management tactics to identify, assess, and manage risks involved in the execution of applications with regards to service requirements and customer needs,
- incorporate autonomic resource management models that effectively self-manage changes in service requirements to satisfy both new service demands and existing service obligations, and
- Leverage VM technology to dynamically assign resource shares according to service requirements.
-

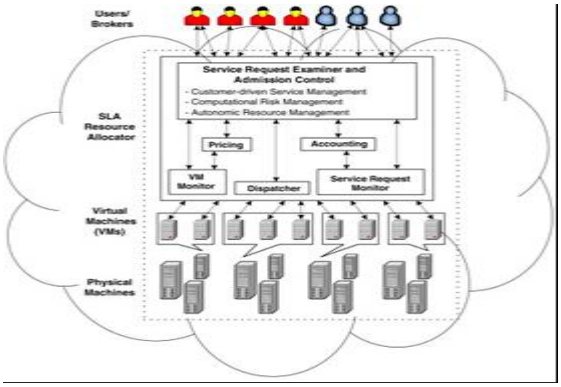

Figure 1. High-level market-oriented Cloud architecture

IV. SERVICE-ORIENTED ARCHITECTURE

A service-oriented architecture service exposes a clearly defined activity—like credit card validation—to consuming business applications that might need to perform that function (such as an order processing application). At the core of the service-oriented architecture philosophy is the modularization of business functions for greater flexibility, manageability, and reusability.

With thoughtful engineering and an enterprise point of view, SOA offers positive benefits which are as follow.

A. *Language-Neutral Integration*

The foundational contemporary Web Services standards use eXtensible Markup Language, which is focused on the creation and consumption of delimited text. Regardless of the development language used, these systems can offer and invoke services through a common mechanism. Programming language neutrality is a key differentiator from past integration approaches.

B. *Component reuse*

Given current Web Service technology, once an organization has built a soft ware component and offered it as a service, the rest of the organization can then utilize that service. With proper service governance, emphasizing topics such as service provider trust, service security, and reliability, Web Services offer the potential for aiding the more effective management of an enterprise portfolio, allowing a capability to be built well once and then shared. Multiple components can be combined to offer greater capabilities in what is often termed “orchestration.”

C. *Organizational agility*

SOA defines building blocks of soft ware capability in terms of offered services that meet some portion of the organization’s requirements. These building blocks, once defined and reliably operated, can be recombined and integrated rapidly.

D. Leveraging existing systems

One common use of SOA is to define elements or functions of existing application systems and make them available to the enterprise in a standard agreed-upon way, leveraging the substantial investment already made in existing applications. The most compelling business case for SOA is oft en made regarding leveraging this legacy investment, enabling integration between new and old systems components.

V. TRANSFORMING AN EXISTING ARCHITECTURE

The fact that an Inside-Out architecture typically is *not* service-oriented — even though it might be possible to access application functionality *using* Web services — suggests that just using the wrapper strategy will not yield the benefits of a full Outside-In architecture implementation, and compensation for Inside-Out architecture limits may even be more costly than taking an alternative approach.

The process of converting an Inside-Out architecture to an Outside-In one, we consider how a typical Web application platform could be converted to an Outside-In architecture in which some Web application accesses all critical business functionality through a Web services layer, and Web services are hosted in a cloud, a service grid, or internally.

From a layered perspective, a Web application usually can be described by a graphic of a three-tiered architecture like the Figure 2.

At the top of the graphic we see a user interface layer, which usually is implemented using some Web server (like Microsoft’s IIS or Apache’s HTTP Web server) and scripting languages or servlet-like technologies that they support.

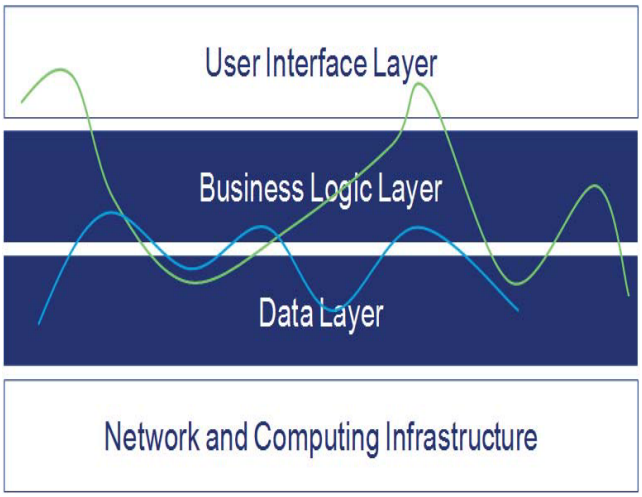

Figure 2.Existing Architecture

The second layer, the business logic layer, is where all business logic programmed in Java, C#, Visual Basic, and php/python/perl/tcl (or pick your favorite programming language that can be used to code libraries of business functionality) is put. The data layer is where code that manipulates basic data structures goes, and this usually is constructed using object and/or relational database technologies. All of these layers are deployed on a server configured with an operating system and network infrastructure enabling an application user to access Web application functionality from a browser or rich internet client application. The blue and red lines illustrate that business and data logic sometimes are commingled with code in other layers of the architecture, making it difficult to modify and manage the application over time (code that is spread out and copied all over the architecture is hard to maintain). Ideally, the red and blue lines would not exist at all in this diagram, so it is here where we start in the process of converting this Inside-Out architecture to an Outside-In one.

A. Addressing Architecture Layering and Partitioning

The first step of transitioning from one architecture style to another is to correct mistakes relating to layering wherever possible. This requires code to be cleaned and commented, refactored, and consolidated so that it is packaged for reuse and orderly deployment, and so that cross-layer violations (e.g., database specifics and business logic are removed from the UI layer, or business logic is removed from the data layer) are eliminated.

Assuming layering violations are addressed, it makes sense then to introduce a service application programming interface (API) between the User Interface Layer and the Business Logic Layer as in Figure 3.
The service layer illustrated here is positioned between the User Interface and lower architecture layers as the only means of accessing lower level functionality. This means that the concerns of one architecture layer do not become

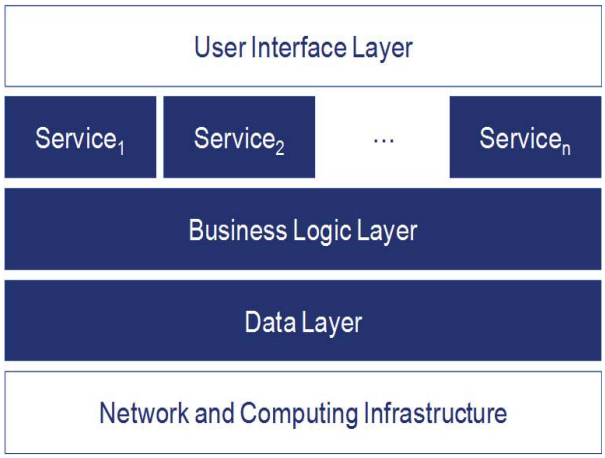

Figure 3.building services in the layer

or complicate the at other levels. But while we may have cleaned up layering architecture violations, we may not have cleaned up *partitioning* violations. Partitioning refers to the "componentizing" or "modularizing" of business functionality such that a component in one business functional domain (e.g., order management) accesses functionality in another such domain (e.g., inventory management) through a single interface (ideally using the appropriate service API). Partitioning also may be referred to as factoring. When transitioning to a new architecture style, the first stage of partitioning often is implemented at the Business Logic Layer, resulting in a modified architecture depicted as Figure 4.

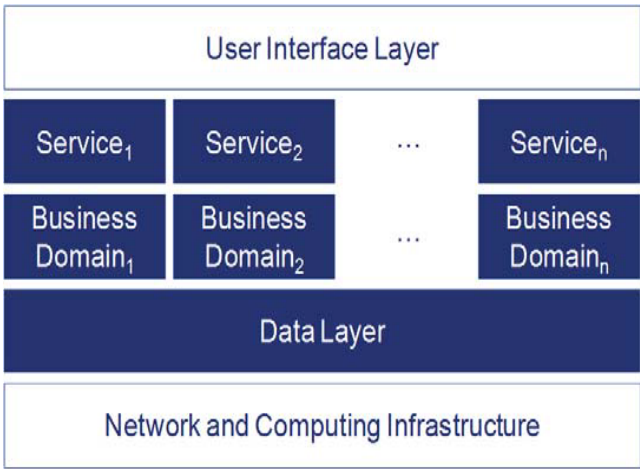

Figure 4.Partition of Business Logic Layer

The next phase of transformation focuses attention on partitioning functionality in the database. Because it is possible to transition the architecture in Figure 2 to become likes one of the depictions below in Figure 5 illustrates a well-organized platform that might be centrally hosted.

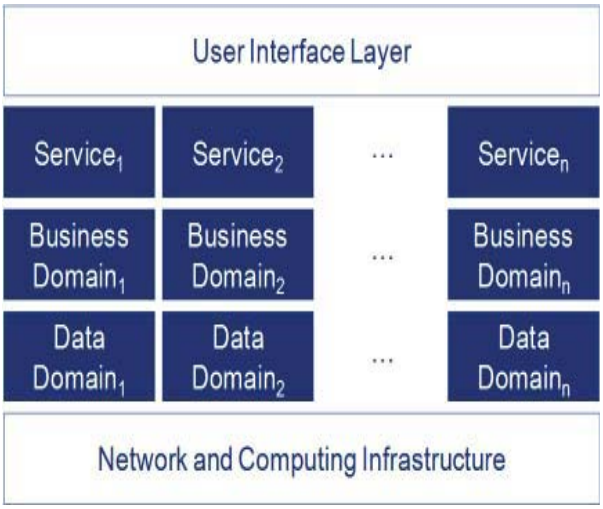

Figure 5.Well-Organized Layer

Figure 6 illustrates a well organized platform that could be hosted in a service grid or even many service grids.

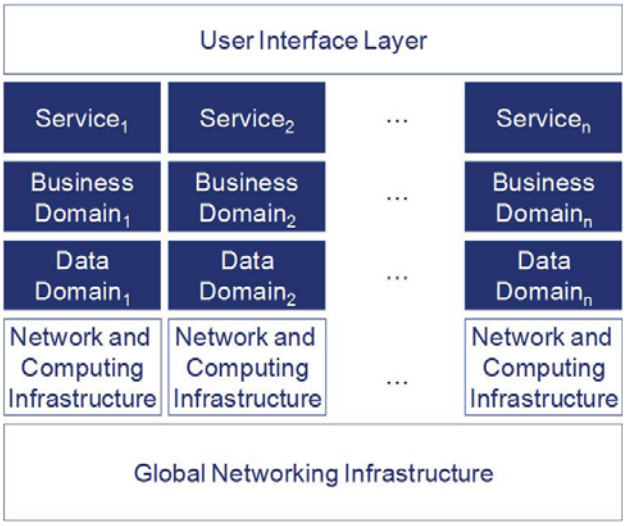

Figure 6. Layers with service grids

Figures 5 and 6 make it simple to see that services and their supporting business logic and data functionality could be replaced easily with an alternative service implementation without negatively impacting other areas of the architecture, provided that functionality in one service domain is accessed by another service domain only through the service interface.

B. Externalizing Policy

The next step toward implementing an Outside-In architecture is to external both business and infrastructure policies from any of the functionality provisioning services illustrated in the figures above. Our use of the word policy connotes constraints placed upon the business functionality of a system, harmonized with constraints on the infrastructure (hardware and software) that provisions that functionality. Policy extension points provide the means by which policy constraints are exposed to business and corresponding infrastructural functionality and incorporated into their execution. Externalizing policy highlights a significant distinction between Inside-Out and Outside-In

architecture styles. To illustrate the problem of scaling systems where policy is distributed throughout it, consider the system illustrated in Figure 7.

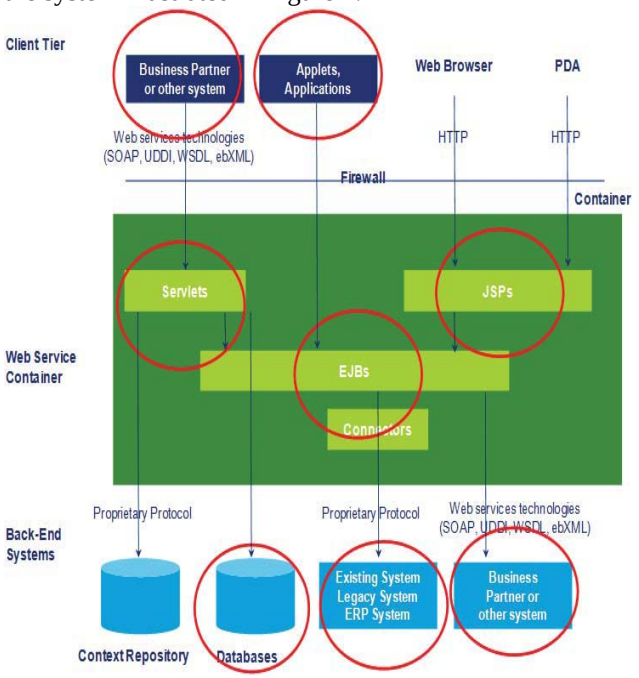

Figure 7. Service Architecture Example

Figure 7 illustrates a system where business policy exists in multiple locations of the architecture as indicated by areas outlined in red. Scaling this architecture would be disastrous because policy would be distributed as copies (or, worst case, as different code bases) over a very complex deployment environment.

VI. CONCLUSION

Cloud computing is an emerging computing paradigm that is increasingly popular. Leaders in the industry, such as Microsoft, Google, and IBM, have provided their initiatives in promoting cloud computing. However, the public literature that discusses the research issues in cloud computing are still inadequate.

Transforming Inside-Out architecture to an Outside-In architecture can be a lengthy process — it is a function of existing system complexity, size, and age. The foresight to recognize the company's need to create a platform. The corresponding need to make architecture changes to support more rapid development and simpler deployment of new services.

VII. FUTURE ENHANCEMENT

One of the most important challenges ahead is that clouds will always be compared to local machine in the time of usage. It's important for the user to know what he gains of shifting to the cloud. Obviously using services on local machines, the user needs more resources but at least he knows that he has access to his data all the time and he has the data he owns on his local machine. But who is in charge of restoring his data if something happens to the cloud and the fact that the user is not aware of the physical place which his data is stored makes cloud more unreliable one.

REFERENCES

[1] Kirsten ferguson-Boucher, Aberystwyth University, Walez, "Cloud Computing a Records and Information Management Perspective", IEEE cloud computing, 2012.

[2] Rajkumar Buyya, Chee Shin Yeo, and Srikumar Venugopal, "Market-Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities", (DIISR), 2012.

[3] Hosam AlHakami, Hamza Aldabbas, and Tariq Alwada, De Montfort University,Leicester, United Kingdom, "Comparison Between Cloud And Grid Computing: Review Paper", International Journal on Cloud Computing: Services and Architecture (IJCCSA),Vol.2, No.4, August 2012.

[4] Abu Sarwar Zamani , Md. Mobin Akhtar and Sultan Ahmad, " Emerging Cloud Computing Paradigm", IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 4, No 2, July 2011.

[5] M.A. Vouk, "Virtualization of Information Technology Resources", in Electronic Commerce: A Managerial Perspective 2008, 5th Edition y Turban, Prentice-Hall Business Publishing.

[6] [5] A. Weiss. Computing in the Clouds. *netWorker*,11(4):16-25, Dec. 2007.

[7] Twenty Experts Define Cloud Computing,http://cloudcomputing.syscon.com/read/612375_p.htm [18 July 2008].

[8] R. Buyya, D. Abramson, and S. Venugopal. The Grid Economy. *Proceedings of the IEEE*, 93(3): 698-714, IEEE Press, USA, March 2005.

[9] R. Buyya, D. Abramson, and S. Venugopal. The Grid Economy. *Proceedings of the IEEE*, 93(3): 698-714, IEEE Press, USA, March 2005.

[10] S. Venugopal, X. Chu, and R. Buyya. A Negotiation Mechanism for Advance Resource Reservation using the Alternate Offers Protocol. In *Proceedings of the 16th International Workshop on Quality of Service (IWQoS 2008)*, Twente, The Netherlands, June 2008.

[11] D. Hamilton. 'Cloud computing' seen as next wave for technology investors. *Financial Post*, 04 June 2008. http://www.financialpost.com/money/story.html

[12] Mike Ricciuti, "Stallman: Cloud computing is 'stupidity'", http://news.cnet.com/8301-1001_3-10054253-92.html

[13] K. Keahey, I. Foster, T. Freeman, and X. Zhang. Virtual workspaces: Achieving quality of service and quality of life in the Grid. *Scientific Programming*, 13(4):265-275, October 2005.

[14] I. Llorente, OpenNebula Project. http://www.opennebula.org/ [23 July 2008]

[15] Amazon Elastic Compute Cloud (EC2), http://www.amazon.com/ec2/ [18 July 2008]

AUTHOR PROFILE

Ms.V.E.UNNAMALAI, Received The B.E(Computer hardware and software Engineering) From Avinashilingam University, Coimbatore, Tamilnadu, India and Presently Pursuing Final Year M.TECH CSE, In PRIST University, Puducherry Campus, Puducherry, India in 2012 and 2014.

Ms.J.R.THRESPHINE**,** Received The M.Tech In Computer Science And Engineering. Presently she is a Working Assistant Professor in Computer Science and Engineering at PRIST University, Puducherry Campus, and Puducherry, India.