Architectural System Analysis in Cloud Computing

Praveen Kumar Shrivastava
Research Scholar
Department of Computer Application
Dr. C.V. Raman University, Bilaspur, Chhattisgarh, India

Dr. S. M. Ghosh
Professor
Department of Computer Science & Engineering
Rungta College of Engineering & Technology, Bhilai, Chhattisgarh, India

Vibha Sahu
Research Scholar
Department of Computer Application
Dr. C.V. Raman University, Bilaspur, Chhattisgarh, India

Dipti Singh Galav
Research Scholar
Department of Computer Application
Dr. C.V. Raman University, Bilaspur, Chhattisgarh, India

Abstract

Cloud Architectures are designs of software applications that use 'Internet-Accessible on-Demand Services'. Applications built on Cloud Architectures are such that the underlying computing infrastructure is used as per need, to perform a specific task. Ones the task is completed, the un used or un needed recourses are either relinquished or often disposed. Based on the needs of the client, the applications scales either upwards or downwards while operating. This paper shows the architecture of cloud computing in practical approach, means what type of front end or back end it uses and also what type of infrastructure it requires for that, etc.

Keywords: Cloud Computing, On Demand Services, SOAP.

I. INTRODUCTION

Cloud computing is a natural extension of many of the design principles, protocols, plumbing, and systems that have been developed over the past 20 years. However, cloud computing describes some new capabilities that are architected into an application stack and are responsible for the programmability, scalability, and virtualization of resources. One property that differentiates cloud computing is referred to as composability, which is the ability to build applications from component parts. A platform is a cloud computing service that is both hardware and software. Platforms are used to create more complex software. Virtual appliances are an important example of a platform, and they are becoming a very important standard cloud computing deployment object. Cloud computing requires some standard protocols with which different layers of hardware, software, and clients can communicate with one another. Many of these protocols are standard Internet protocols. Cloud computing relies on a set of protocols needed to manage inter process communications that have been developed over the years. The most commonly used set of protocols uses XML as the messaging format, the Simple Object Access Protocol (SOAP), protocol as the object model, and a set of discovery and description protocols based on the Web Services Description Language (WSDL) to manage transactions.

Some completely new clients are under development that is specifically meant to connect to the cloud. These clients have as their focus cloud applications and services, and are often hardened and more securely connected. Two examples presented are Jolicloud and Google Chrome OS. They represent a new client model that is likely to have considerable impact.

II. CLOUD COMPUTING STACK

Cloud computing builds on the architecture developed for staging large distributed network applications on the Internet over the last 20 years. To these standard networking protocols, cloud computing adds the advances in system virtualization that became available over the last decade. The cloud creates a system where resources can be pooled and partitioned as needed. Cloud architecture can couple software running on virtualized hardware in multiple locations to provide an on-demand service to user-facing hardware and software. It is this unique combination of abstraction and metered service that separates the architectural requirements of cloud computing systems from the general description given for an entire Internet application.

Many descriptions of cloud computing describes it in terms of two architectural layers:

- A client as a front end
- The "cloud" as a backend

This is a very simplistic description because each of these two components is composed of several component layers, complementary functionalities, and a mixture of standard and proprietary protocols. Cloud computing may be differentiated from older models by describing an encapsulated information technology service that is often controlled through an Application Programming Interface (API), thus modifying the services that are delivered over the network.

A cloud can be created within an organization's own infrastructure or outsourced to another data center.While resources in a cloud can be real physical resources, more often they are virtualized resources because virtualized resources are easier to modify and optimize. A compute cloud requires virtualized storage to support the staging and storage of data. From a user's perspective, it is important that the resources appear to be infinitely scalable, that the service be measurable, and that the pricing be metered.

III. Composability

Applications built in the cloud often have the property of being built from a collection of components, a feature referred to as composability. A compos able system uses components to assemble services that can be tailored for a specific purpose using standard parts. A composable component must be:

- Modular:

It is a self-contained and independent unit that is cooperative, reusable, and replaceable.

- Stateless:

A transaction is executed without regard to other transactions or requests. It isn't an absolute requirement that transactions be stateless, some cloud computing applications provide managed states through brokers, transaction monitors, and service buses. In rarer cases, full transactional systems are deployed in the clouds, but these systems are harder to architect in a distributed architecture. Although cloud computing doesn't require that hardware and software be composable, it is a highly desirable characteristic from a developer or user's standpoint, because it makes system design easier to implement and solutions more portable and interoperable.

There is a tendency for cloud computing systems to become less composable for users as the services incorporate more of the cloud computing stack. From the standpoint of an IaaS (Infrastructure as a Service) vendor such as Amazon Web Services, Go Grid, or Rack space, it makes no sense to offer non-standard machine instances to customers, because those customers are almost certainly deploying applications built on standard operating systems such as Linux, Windows, Solaris, or some other well-known operating system. In the next step up the cloud computing stack, PaaS (Platform as a Service) vendors such as Windows Azure or Google AppEngine may narrow the definition of standard parts to standard parts that work with their own platforms, but at least from the standpoint of the individual platform service provider, the intent is to be modular for their own developers.

When you move to the highest degree of integration in cloud computing, which is SaaS (Software as a Service), the notion of composability for users may completely disappear. An SaaS vendor such as Quicken.com or Salesforce.com is delivering an application as a service to a customer, and there's no particular benefit from the standpoint of the service provider that the customer be able to compose its own custom applications. A service provider reselling an SaaS may have the option to offer one module or another, to customize the information contained in the module for a client, to sell the service under their own brand, or to perform some other limited kind of customization, but modifications are generally severely limited. This idea that composability diminishes going up the cloud computing stack is from the user's point of view. If you are a PaaS or SaaS service provider and your task is to create the platform or service presented to the developer, reseller, or user, the notion of working with a composable system is still a very powerful one. A PaaS or SaaS service provider gets the same benefits from a composable system that a user does—these things, among others:

- Easier to assemble systems
- Cheaper system development
- More reliable operation
- A larger pool of qualified developers
- A logical design methodology

You encounter the trend towards designing composable systems in cloud computing in the widespread adoption of what has come to be called the Service Oriented Architecture (SOA). The essence of a service oriented design is that services are constructed from a set of modules using standard communications and service interfaces. An example of a set of widely used standards describes the services themselves in terms of the Web Services Description Language (WSDL), data exchange between services using some form of XML, and the communications between the services using the SOAP protocol. There are, of course, alternative sets of standards.

IV. Infrastructure

Most large Infrastructure as a Service (IaaS) providers rely on virtual machine technology to deliver servers that can run applications. Virtual servers described in terms of a machine image or instance have characteristics that often can be described in term s of real servers delivering a certain number of microprocessor (CPU) cycles, memory access, and network bandwidth to customers. Virtual machines are containers that are assigned specific resources.The software that runs in the virtual machines is what defines the utility of the cloud computing system.

Figure I shows the portion of the cloud computing stack that is defined as the "server." In the diagram, the API is shown shaded in gray because it is an optional component that isn't always delivered with the server. The VMM component is the

Virtual Machine Monitor, also called a hyper visor. This is the low-level software that allows different operating systems to run in their own memory space and manages I/O for the virtual machines.

The notion of a virtual server presents to an application developer a new way of thinking about and programming applications. For example, when a programmer is creating software that requires several different tasks to be performed in parallel, he might write an application that creates additional threads of execution that must be managed by the application. When a developer creates an application that uses a cloud service, the developer can attach to the appropriate service(s) and allow the application itself to scale the program execution. Thus, an application such as a three-dimensional rendering that might take a long time for a single server to accomplish can be scaled in the cloud to many servers at once for a short period of time, accomplishing the task at a similar or lower price but at a much faster rate.

In future applications, developers will need to balance the architectural needs of their programs so their applications create new threads when it is appropriate or create new virtual machines. Applications will also need to be mindful of how they use cloud resources, when it is appropriate to scale execution to the cloud, how to monitor the instances they are running, and when not to expand their application's usage of the cloud. This will require a new way of thinking about application development, and the ability to scale correctly is something that will have to be architected into applications from the ground up.

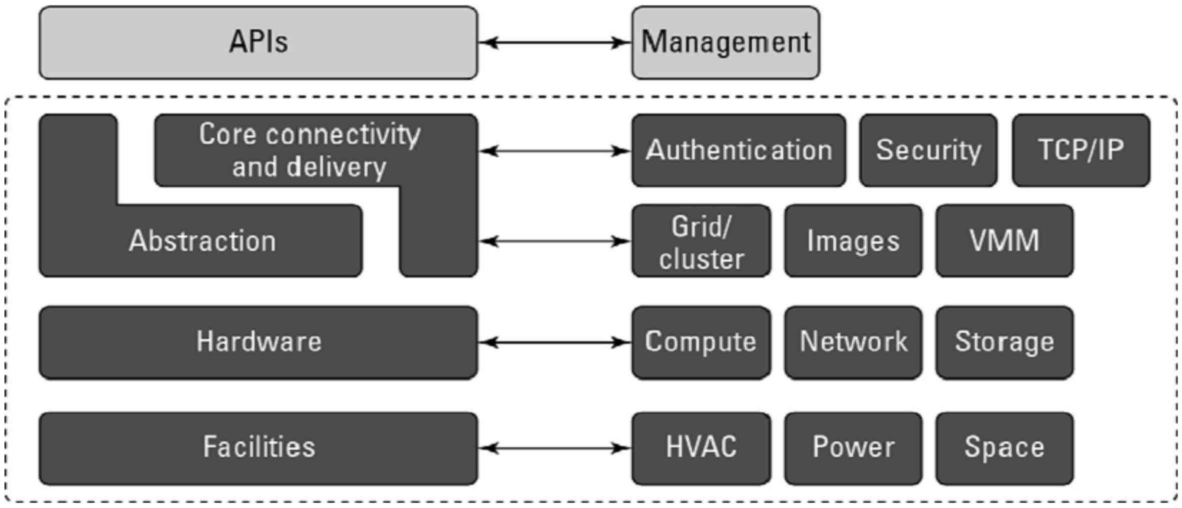

Fig. 1: Application Interface

V. PLATFORMS

A platform in the cloud is a software layer that is used to create higher levels of service. Many different Platform as a Service (PaaS) providers offer services meant to provide developers with different capabilities. PaaS is explored more thoroughly, but for now it is useful to cite three of the major examples are :

- Salesforce.com's Force.com Platform
- Windows Azure Platform
- Google Apps and the Google AppEngine

These three services offer all the hosted hardware and software needed to build and deploy Web applications or services that are custom built by the developer within the context and range of capabilities that the platform allows. Platforms represent nearly the full cloud software stack, missing only the presentation layer that represents the user interface. This is the same portion of the cloud computing stack that is a virtual appliance. What separates a platform from a virtual appliance is that the software that is installed is constructed from components and services and controlled through the API that the platform provider publishes.

It makes sense for operating system vendors to move their development environments into the cloud with the same technologies that have been successfully used to create Web applications. Thus, you might find a platform based on a Sun xVM hyper visor virtual machine that includes a Net Beans, Integrated Development Environment (IDE) and that supports the Sun Glass Fish Web, stack programmable using Pearl or Ruby. For Windows, Microsoft would be similarly interested in providing a platform that allowed Windows developers to run on a Hyper-V VM, use the ASP.NET application framework, support one of its enterprise applications such as SQL Server, and be programmable within Visual Studio—which is essentially what the Azure Platform does. This approach allows someone to develop a program in the cloud that can be used by others.

VI. CONCLUSION

In this paper we learned about the various architectural layers that make up the cloud computing stack. Cloud computing may be seen to be an extension of older Internet standards, but it includes some new architecture. In particular, infrastructure components were described, as were platform components. Platform services are software-based and can include a class of applications referred to as virtual appliances.

The focus of the paper is, how all the different parts of cloud computing work together? Many of the important standards are aimed at managing transactions in the cloud and making components work together. In this paper we took an initial look at SOAP, XML, and the various WSDL services.

REFERENCES

[1] Cloud Computing Bible Pp 62-66, Barre Sosinsky
[2] http://ijltet.org/wp-content/uploads/2014/04/33.pdf
[3] http:// is.muni.cz/th/387126/fi_m/387126_DimeDimovksi_DaaS_For_SMO Final. pdf pp12