Research on the Architecture of Cloud Computing

Jun Wu, Jing Yang, Xisi Tu, and Guangling Yi

School of Economics and Management
Beijing University of Posts and Telecommunications
Beijing, China
junwu2011@sina.com

Abstract. Cloud computing is a paradigm that focuses on sharing data and computations over a scalable network of nodes. In this paper, we propose five-layer architecture to illustrate their interrelations as well as their inter-dependency on preceding technologies.

Keywords: Cloud computing, five layer architecture.

1 Introduction

Cloud computing can be considered a new computing paradigm that allows users to temporary utilize computing infrastructure over the network, supplied as a service by the cloud-provider at possibly one or more levels of abstraction. Consequently, several business models rapidly evolved to harness this technology by providing software applications, programming platforms, data-storage, computing infrastructure and hardware as services. While they refer to the core cloud computing services, their inter-relations have been ambiguous and the feasibility of enabling their inter-operability has been debatable.

At the same time, the current cloud computing research lacks the understanding of the classification of the cloud systems, their correlation and inter-dependency. In turn, this obscurity is hindering the advancement of this research field. As a result, the introduction of an organization of the cloud computing knowledge domain, its components and their relations is necessary to help the research community achieve a better understanding of this novel technology.

In this paper, we present a five-layer architecture abstraction of cloud computing, with a classification of its components, and their relationships as well as their dependency on some of the prior concepts from other fields in computing. In addition, we highlight some of the technical challenges involved in building cloud components in each layer of our proposed architecture, and how these challenges were addressed in the predecessor areas of computing research. We also underline the constraints associated with specifying a user interface for each layer.

The rest of this paper is organized as follows. Section 2 discusses the research status quo in light of the limitations and imperfections of current cloud systems. Section 3 examines each layer of the architecture, elaborating on its composition, limitations, and correlation with other layers. We conclude our paper with a discussion on several challenges for cloud computing research, and the various hinders delaying its wide adoption in application area.

2 Research Motivation

The recent evolution of cloud computing has borrowed its basics from several other computing areas and systems engineering concepts. Cluster and Grid Computing on one hand, and virtualization on the other hand are perhaps the most obvious predecessor technologies that enabled the inception of cloud computing. However, several other computing concepts have indirectly shaped today's cloud computing technology, including peer-to-peer (P2P) computing [1], SOA [2] and autonomic computing [3].

In order to comprehend the impact of those various concepts on cloud computing, our approach is to determine the different layers and components that make the cloud, and study their characteristics in light of their dependency on the other computing fields and models. Therefore, it was important to define the constituents of cloud computing in order to define its strengths and imperfections with respect of its adoption of previous technological concepts.

Besides, an architecture of cloud computing will allow better understanding of the inter-relations between the different cloud components, enabling the composition of new systems from existing components and further re-composition of current systems from other cloud components for desirable features like extensibility, flexibility, availability or merely optimization and better cost efficiency.

3 The Cloud Architecture

Cloud computing systems fall into one of five layers: applications, software environments, software infrastructure, software kernel, and hardware. Obviously, at the bottom of the cloud stack is the hardware layer which is the actual physical components of the system. Some cloud computing offerings have built their system on subleasing the hardware in this layer as a service, as we discuss in subsection III-E. At the top of the stack is the cloud application layer, which is the interface of the cloud to the common computer users through web browsers and thin computing terminals. We closely examine the characteristics and limitations of each of the layers in the next five subsections.

A. Cloud Application Layer

The cloud application layer is the most visible layer to the end-users of the cloud. Normally, the users access the services provided by this layer through web-portals, and are sometimes required to pay fees to use them. This model has recently proven to be attractive to many users, as it alleviates the burden of software maintenance and the ongoing operation and support costs. Furthermore, it exports the computational work from the users' terminal to data centers where the cloud applications are deployed. This in turn lessens the restrictions on the hardware requirements needed at the users' end, and allows them to obtain superb performance to some of their cpu-intensive and memory-intensive workloads without necessitating huge capital investments in their local machines.

As for the providers of the cloud applications, this model even simplifies their work with respect to upgrading and testing the code, while protecting their intellectual property. Since a cloud application is deployed at the provider's computing infrastructure (rather than at the users' desktop machines), the developers of the application are able to roll smaller patches to the system and add new features without disturbing the users with requests to install major updates or service packs. Configuration and testing of the application in this model is arguably less complicated, since the deployment environment becomes restricted, i.e., the provider's data center. Even with respect to the provider's margin of profit, this model supplies the software provider with a continuous flow of revenue, which might be even more profitable on the long run. This model conveys several favorable benefits for the users and providers of cloud applications, and is normally referred to as Software as a Service (SaaS). Salesforce Customer Relationships Management (CRM) system [4] and Google Apps [5] are two examples of SaaS. As such, the body of research on SOA has numerous studies on composable IT services which have direct application to providing and composing SaaS.

Despite all the advantageous benefits of this model, several deployment issues hinder its wide adoption. Specifically, the security and availability of the cloud applications are two of the major issues in this model, and they are currently avoided by the use of lenient service level agreements (SLA). Furthermore, coping with outages is a realm that users and providers of SaaS have to tackle, especially with possible network outage and system failures. Additionally, the integration of legacy applications and the migration of the users' data to the cloud is another matter that is also slowing the adoption of SaaS. Before they can persuade users to migrate from desktop applications to cloud applications, cloud applications' providers need to address end-users' concerns about security and safety of storing confidential data on the cloud, users authentication and authorization, up-time and performance, as well as data backup and disaster recovery and provide reliable SLAs for their cloud applications.

B. Cloud Software Environment Layer

The second layer in our proposed cloud architecture is the cloud software environment layer (also dubbed the software platform layer). The users of this layer are cloud applications' developers, implementing their applications for and deploying them on the cloud. The providers of the cloud software environments supply the developers with a programming-language-level environment with a set of well-defined APIs to facilitate the interaction between the environments and the cloud applications, as well as to accelerate the deployment and support the scalability needed of those cloud applications. The service provided by cloud systems in this layer is commonly referred to as Platform as a Service (PaaS). One example of systems in this category is Google's App Engine [5], which provides a python runtime environment and APIs for applications to interact with Google's cloud runtime environment. Another example is SalesForce Apex language [6] that allows the developers of the cloud applications to design, along with their applications' logic, their page layout, workflow, and customer reports.

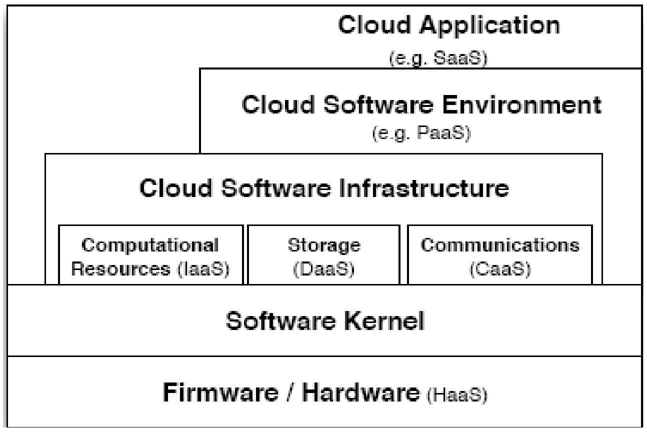

Cloud Application
(e.g. SaaS)
Cloud Software Environment
(e.g. PaaS)
Cloud Software Infrastructure
Computational Resources (IaaS)
Storage (DaaS)
Communications (CaaS)
Software Kernel
Firmware / Hardware (HaaS)

Fig. 1. Our Proposed Cloud Computing Architecture

Developers reap several benefits from developing their cloud application for a cloud programming environment, including automatic scaling and load balancing, as well as integration with other services (e.g. authentication services, email services, user interface) provided to them through the PaaS-provider. In such a way, much of the overhead of developing cloud applications is alleviated and is handled at the environment level. Furthermore, developers have the ability to integrate other services to their applications on-demand. This in turn makes the cloud application development a less complicated task, accelerates the deployment time and minimizes the logic faults in the application. As such, cloud software environments facilitate the process of the development of cloud applications.

C. Cloud Software Infrastructure Layer

The cloud software infrastructure layer provides fundamental resources to other higher-level layers, which in turn can be used to construct new cloud software environments or cloud applications. Our proposed architecture reflects the fact that the two highest levels in the cloud stack can bypass the cloud infrastructure layer in building their system. Although this bypass can enhance the efficiency of the system, it comes at the cost of simplicity and development efforts.

Cloud services offered in this layer can be categorized into: computational resources, data storage, and communications

1) Computational Resources:

Virtual machines (VMs) are the most common form for providing computational resources to cloud users at this layer, where the users get finer-granularity flexibility since they normally get super-user access to their VMs, and can use it to customize the software stack on their VM for performance and efficiency. Often, such services are dubbed Infrastructure as a Service (IaaS). Virtualization is the enabler technology for this cloud component, which allows the users unprecedented flexibility in configuring their settings while protecting the physical infrastructure of the provider's data center. Recent advances in OS Virtualization have made the concept of IaaS plausible. This was specifically enabled by two virtualization technologies: paravirtualization and hardware-assisted virtualization. Although both virtualization technologies have addressed performance isolation between virtual machines contending on common resources, performance interference between VMs sharing the same cache and TLB hierarchy cannot yet be avoided [7]. Further, the emergence of multicore machines into mainstream servers exacerbate this performance interference problem. In turn, the lack of strict performance isolation between VMs sharing the same physical node has resulted in the inability of cloud providers to give strong guarantees for performance to their clients. Instead, they offer them unsatisfactory SLAs in order to provide a competitive pricing for the service. Such weak guarantees, unfortunately, can inject themselves up the layers of the cloud stack, and affect the SLAs of the cloud systems built above the IaaS's SLAs.

2) Data Storage:

The second infrastructure resource is data storage, which allows users to store their data at remote disks and access them anytime from any place. This service is commonly known as Data-Storage as a Service (DaaS), and it facilitates cloud applications to scale beyond their limited servers. Data storage systems are expected to meet several rigorous requirements for maintaining users' data and information, including high availability, reliability, performance, replication and data consistency; but because of the conflicting nature of these requirements, no one system implements all of them together. For example, availability, scalability and data consistency can be regarded as three conflicting goals. While those features are hard to be met with general data storage systems, DaaS-providers have taken the liberty of implementing their system to favor one feature over the others, while indicating their choice through their SLA. These implementations have borrowed their fundamental ideas from proceeding research and production systems. Some examples of data storage systems are: distributed file systems (e.g., GFS [8]), replicated relational databases (RDBMS) (e.g., Bayou [9]) and key value stores (e.g., Dynamo [10]). RDBMS, for example opt to present a stricter consistency model at the cost of the availability of the data, while key-value stores have placed more importance on the availability of the data while relaxing the consistency model for the storage. In this respect, the cloud DaaS has inherited the different characteristics of today's data storage systems. Example of commercial DaaS-systems are Amazon's S3 [11] and EMC Storage Managed Service [12].

3) Communication:

As the need for a guaranteed quality of service (QoS) for network communication grows for cloud systems, communication becomes a vital component of the cloud infrastructure. Consequently, cloud systems are obliged to provide some communication capability that is service oriented, configurable, schedulable, predictable, and reliable. Towards this goal, the concept of Communication as a Service (CaaS) emerged to support such requirements, as well as network security, dynamic provisioning of virtual overlays for traffic isolation or dedicated bandwidth, guaranteed message delay, communication encryption, and network monitoring. Although this model is the least discussed and adopted cloud service in the commercial cloud systems, several research papers and articles [11], [12], [13] have investigated the various architectural design decisions, protocols and solutions needed to provide QoS communications as a service. One recent example of systems that belong to CaaS is Microsoft Connected Service Framework (CSF) [14]. VoIP telephone systems, audio and video conferencing as well as instant messaging are candidate cloud applications that can be composed of CaaS and can in turn provide composable cloud solutions to other common applications.

D. Software Kernel

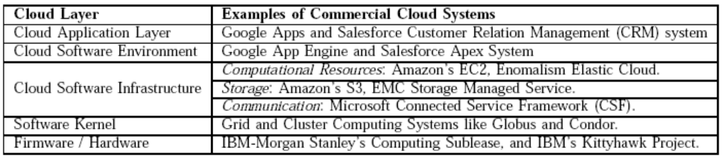
This cloud layer provides the basic software management for the physical servers that compose the cloud. Software kernels at this level can be implemented as an OS kernel, hypervisor, virtual machine monitor and/or clustering middleware. Customarily, grid computing applications were deployed and run on this layer on several interconnected clusters of machines. However, due to the absence of a virtualization abstraction in grid computing, jobs were closely tied to the actual hardware infrastructure and providing migration, checkpointing and load balancing to the applications at this level was always a complicated task.

E. Hardware and Firmware

The bottom layer of the cloud stack in our proposed architecture is the actual physical hardware and switches that form the backbone of the cloud. In this regard, users of this layer of the cloud are normally big enterprises with huge IT requirements in need of subleasing Hardware as a Service (HaaS). For that, the HaaS provider operates, manages and upgrades the hardware on behalf of its consumers, for the life-time of the sublease. This model is advantageous to the enterprise users, since they do not need to invest in building and managing data centers. Meanwhile, HaaS providers have the technical expertise as well as the cost-effective infrastructure to host the systems. One of the early examples HaaS is Morgan Stanley's sublease contract with IBM in 2004 [14]. SLAs in this model are more strict, since enterprise users have predefined business workloads whose characteristics impose strict performance requirements. The margin benefit for HaaS providers materialize from the economy of scale of building huge data centers infrastructures with gigantic floor space, power, cooling costs as well as operation and management expertise.

HaaS providers have to address a number of technical challenges in operating and managing their services. Efficiency, ease and speed of provisioning such large scale systems, for example is a major challenge. Remote scriptable boot-loaders are one solution to remotely boot and deploy complete software stacks on the data centers. PXE [15] and UBoot [16] are examples of remote bootstrap execution environments that allow the system administrator to stream a binary image to multiple remote machines at boot-time. One example of such systems is IBM Kittyhawk [17], a

Table 1. Summary of examples of some commercial cloud computing systems in our proposed cloud computing Architecture



Cloud Layer	Examples of Commercial Cloud Systems
Cloud Application Layer	Google Apps and Salesforce Customer Relation Management (CRM) system
Cloud Software Environment	Google App Engine and Salesforce Apex System
Cloud Software Infrastructure	*Computational Resources*: Amazon's EC2, Enomalism Elastic Cloud.
	Storage: Amazon's S3, EMC Storage Managed Service.
	Communication: Microsoft Connected Service Framework (CSF).
Software Kernel	Grid and Cluster Computing Systems like Globus and Condor.
Firmware / Hardware	IBM-Morgan Stanley's Computing Sublease, and IBM's Kittyhawk Project.

research project that uses UBoot to script the boot sequence of thousands of remote Bluegene/P nodes over the network. Other examples of challenges that arise at this cloud layer include data center management, scheduling, and power-consumption optimizations.

Table 1 exemplify some of the commercial cloud systems discussed earlier in our proposed architecture.

4 Conclusions

In this paper, we proposed a detailed architecture for the cloud in an attempt to establish the knowledge domain of the area of cloud computing and its relevant components. We used composability as our methodology in constructing our cloud architecture which allowed us to capture the inter-relations between the different cloud components. We presented our proposed architecture as a stack of cloud layers, and discussed each layer's strengths, limitations as well as dependence on preceding computing concepts. Few recent online articles attempt to establish a similar structure for cloud computing and its components [16], [17], [18]. Although they attain some valuable understanding of several cloud services and components, they tend to be more general classifications. They neither went to the level of detail in the analysis as we did, nor they included all the cloud layers we captured in our model. Their main end-goal is to classify the commercial cloud offerings in order to analyze the cloud computing market opportunities. As such, they do not address the specific potentials or limitations of the several cloud layers, nor the research opportunities associated with each cloud layer. We believe our proposed cloud architecture is more comprehensive, and encompass more detailed analysis of the cloud computing knowledge domain.

References

1. Androutsellis-Theotokis, S., Spinellis, D.: A survey of peerto- peer content distribution technologies. ACM Comput. Surv. 36(4), 335–371 (2004)
2. Papazoglou, M.P., Heuvel, W.-J.: Service oriented architectures: approaches, technologies and research issues. The VLDB Journal 16(3), 389–415 (2007)
3. Jin, X., Liu, J.: From individual based modeling to autonomy oriented computation. In: Nickles, M., Rovatsos, M., Weiss, G. (eds.) AUTONOMY 2003. LNCS (LNAI), vol. 2969, pp. 151–169. Springer, Heidelberg (2004)
4. Olston, C., Reed, B., Srivastava, U., Kumar, R., Tomkins, A.: Pig latin: a not-so-foreign language for data processing. In: SIGMOD 2008: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, pp. 1099–1110. ACM, New York (2008), http://dx.doi.org/10.1145/1376616.1376726
5. Koh, Y., Knauerhase, R.C., Brett, P., Bowman, M., Wen, Z., Pu, C.: An analysis of performance interference effects in virtual environments. In: ISPASS, pp. 200–209. IEEE Computer Society, Los Alamitos (2007)
6. Ghemawat, S., Gobioff, H., Leung, S.-T.: The google file system. SIGOPS Oper. Syst. Rev. 37(5), 29–43 (2003)

7. Petersen, K., Spreitzer, M., Terry, D., Theimer, M.: Bayou: replicated database services for world-wide applications. In: EW 7: Proceedings of the 7th Workshop on ACM SIGOPS European Workshop, pp. 275–280. ACM, New York (1996)
8. DeCandia, G., Hastorun, D., Jampani, M., Kakulapati, G., Lakshman, A., Pilchin, A., Sivasubramanian, S., Vosshall, P., Vogels, W.: Dynamo: amazon's highly available key-value store. In: SOSP 2007, pp. 205–220. ACM, New York (2007)
9. Hanemann, A., Boote, J.W., Boyd, E.L., Durand, J., Kudarimoti, L., Łapacz, R., Swany, D.M., Trocha, S., Zurawski, J.: PerfSONAR: A service oriented architecture for multi-domain network monitoring. In: Benatallah, B., Casati, F., Traverso, P. (eds.) ICSOC 2005. LNCS, vol. 3826, pp. 241–254. Springer, Heidelberg (2005)
10. Smr, P., Novek, V.: Architecture acquisition for automatic building of scientific portals. In: SOFSEM 2006: 32nd Conference on Current Trends in Theory and Practice of Computer Science, pp. 493–500. Springer, Heidelberg (2006), `http://www.fit.vutbr.cz/research/viewpub.php?id=8004`
11. Chau, M., Huang, Z., Qin, J., Zhou, Y., Chen, H.: Building a scientific knowledge web portal: the nanoport experience. Decis. Support Syst. 42(2), 1216–1238 (2006)
12. Christie, M., Marru, S.: The lead portal: a teragrid gateway and application service architecture: Research articles. Concurr. Comput.: Pract. Exper. 19(6), 767–781 (2007)
13. Plale, B., Rossi, A., Simmhan, Y., Sarangi, A., Slominski, A., Shirasauna, S., Thomas, T.: Building grid portal applications from a webservice component architecture. Proceedings of the IEEE (Special issue on Grid Computing) 93(3), 551–563 (2005)
14. Stanley, M.: IBM ink utility computing deal, `http://news.cnet.com/2100-7339-5200970.html`
15. Appavoo, J., Uhlig, V., Waterland, A.: Project Kittyhawk: building a global-scale computer: Blue Gene/P as a generic computing platform. SIGOPS Oper. Syst. Rev. 42(1), 77–84 (2008)
16. Deelman, E., Singh, G., Livny, M., Berriman, B., Good, J.: The cost of doing science on the cloud: the montage example. In: SC 2008: Proceedings of the 2008 ACM/IEEE Conference on Supercomputing pp. 1–12. IEEE Press, Piscataway (2008)
17. Defogging Cloud Computing: A Taxonomy (June 16, 2008), `http://refresh.gigaom.com/2008/06/16/defogging-cloud-computing-a-taxonomy/`
18. Cloud Services Continuum (July 3, 2008), `http://et.cairenene.net/2008/07/03/cloud-services-ontinuum/`