Cloud computing and its key techniques

Xu Wang
Software School of Xiamen University
Xiamen, China
E-mail: wangxu047@gmail.com

Beizhan Wang
Software School of Xiamen University
Xiamen, China
E-mail: wangbeizhan@gmail.com

Jing Huang
Department of Land and Resources of Jiangxi Province
Nanchang, China
E-mail: huang_jing@126.com

Abstract—**With the development of parallel computing, distributed computing, grid computing, a new computing model appeared, called cloud computing. It aims to share data, calculations, and services transparently among users of a massive grid. It became a hot issue for its advantages such as "reduce costs", "increase business flexibility" and/or "provide business continuity". In this paper, we described what is cloud computing and took Google's cloud computing techniques as an example, summed up key techniques, such as data storage technology (Google File System), data management technology (BigTable), as well as programming model and task scheduling model (Map–Reduce), used in cloud computing, and then some example of cloud computing vendors were illustrated and compared.**

Key Words—Cloud Computing; SaaS; Google App Engine

I. INTRODUCTION

In recent 10 years, Internet has been developing very quickly. The cost of storage, the power consumed by computer and hardware is increasing. The storage space in data center can't meet our needs and the system and service of original internet can't solve above questions, so we need new solutions. At the same time, large enterprises have to study data source fully to support its business [1]. The collection and analysis must be built on a new platform. So we need a new computing model to utilize the vacant resources of computer, increase the economic efficiency through improving utilization rate, decrease the equipment energy consumption.

Cloud computing, a new kind of computing model, is coming. This word is a new word that appears at the fourth season, 2007 [2], which is the long-held dream of computing as a utility, has the potential to transform a large part of the IT industry, making software even more attractive as a service and shaping the way IT hardware is designed and purchased. Developers with innovative ideas for new Internet services no longer require the large capital outlays in hardware to deploy their service or the human expense to operate it.

This paper Section 2 describes the definition, styles and characters of cloud computing. In Section 3 takes Google's cloud computing techniques as an example, summed up key techniques. Some example of cloud computing vendors were illustrated and compared in Section 4. Finally, Section 5 concludes the paper.

II. WHAT IS CLOUD COMPUTING

A. The Concept of Cloud Computing

A number of computing researchers and practitioners have attempted to define Clouds in various ways [3]. Based the observation of the essence of what Clouds are promising to be, this paper follows the definition of cloud computing proposed in [4]:

- "A Cloud is a type of parallel and distributed system consisting of a collection of inter-connected 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."

At a cursory glance, Clouds appear to be a combination of clusters and Grids. However, this is not the case. Clouds are clearly next-generation data centers with nodes "virtualized" through hypervisor technologies such as VMs, dynamically "provisioned" on demand as a personalized resource collection to meet a specific service-level agreement, which is established through a "negotiation" and accessible as a compostable service via "Web 2.0" technologies.

Cloud Computing refers to both the applications delivered as services over the Internet and the hardware and systems software in the datacenters that provide those services. The services themselves have long been referred to as Software as a Service (SaaS). The datacenter hardware and software is what we will call a Cloud. When a Cloud is made available in a pay-as-you-go manner to the general public, which called a Public Cloud; the service being sold is Utility Computing. The term Private Cloud is used to refer to internal data centers of a business or other organization not made available to the general public. Thus, Cloud Computing is the sum of SaaS and Utility Computing, but does not include Private Clouds. People can be users or providers of SaaS, or users or providers of Utility Computing. We focus on SaaS Providers (Cloud Users) and Cloud Providers, which have received less attention than SaaS Users. Fig.1 shows the roles of the people as users or providers of these layers of Cloud Computing [5].

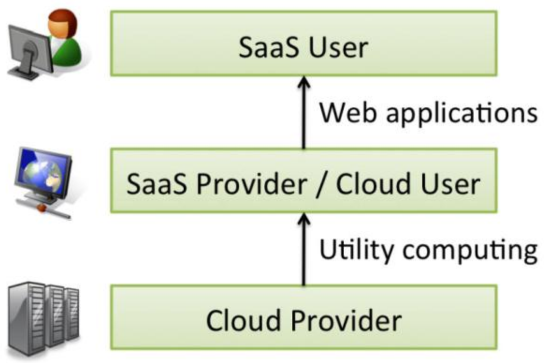

Figure 1. Users and Providers of Cloud Computing.

From a hardware point of view, three aspects are new in Cloud Computing [5].

a) The illusion of infinite computing resources available on demand, thereby eliminating the need for Cloud Computing users to plan far ahead for provisioning.

b) The elimination of an up-front commitment by Cloud users, thereby allowing companies to start small and increase hardware resources only when there is an increase in their needs.

c) The ability to pay for use of computing resources on a short-term basis as needed (e.g., processors by the hour and storage by the day) and release them as needed, thereby rewarding conservation by letting machines and storage go when they are no longer useful.

B. *Cloud Computing Style*

Though people have different views on the cloud computing, they have already reached an agreement on the basic style on it. Its style is as follows [6]:

a) SAAS(Software as a service)

This kind of cloud computing transfer programs to millions of users through browser. In the user's views, this can save some cost on servers and software. In the provider's views, they only need to maintain one program, this can also save cost. Salesforce.com is so far the most famous company that provides this kind of service. SAAS is commonly used in human resource management system and ERP (Enterprise Resource Planning). Google Apps and Zoho Office are also providing this kind of service.

b) Utility Computing

Recently Amazon.com, Sun, IBM and other companies that provide storage services and virtual services are appearing. Cloud computing is creating virtual data center for IT industry to make it can provide service for the whole net through collecting memory, IO equipment, storage and computing power to a virtual resource pool.

c) Network service

Net service has a close relation with SAAS. The service providers can help programmers develop applications based on internet instead of providing single machine procedure through providing API (Application Programming Interface).

d) PAAS(Platform as a service)

Platform as a service, another SAAS, this kind of cloud computing providing development environment as a service. You can use the middleman's equipment to develop your own program and transfer it to the users through internet and servers.

e) MSP (management service provider)

This is one of the ancient applications of cloud computing. This application mostly serves the IT industry instead of end users. It is often used in mail virus scanning and program monitoring.

f) Commercial service platform

The commercial service platform is the mixture of SAAS and MSP (Mixed signal Processor), this kind of computing provides a platform for the interaction between users and service provider. For instance, the user individual expense management system can manage user's expense according user's setting and coordinate all the services that users purchased.

7. Integrating internet

It can integrate all the companies that provide similar services, so that users can compare and select their service provider.

C. *The Characters of Cloud Computing*

a) Ultra large-scale

The scale of cloud is large. The cloud of Google has owned more than one million servers. Even in Amazon, IBM, Microsoft, Yahoo, they have more than hundreds of thousands servers. There are hundreds of servers in an enterprise. Cloud enlarge the user's computing power.

b) Virtualization

Cloud computing makes user get service anywhere, through any kind of terminal. The resources it required come from cloud instead of visible entity. You can complete all you want through net service using a laptop or a mobile phone. Users can attain or share it safely through an easy way, anytime, anywhere. Users can complete a task that can't be completed in a single computer.

c) High reliability

Cloud uses data multi-transcript fault tolerant, the computation node isomorphism exchangeable and so on to ensure the high reliability of the service. Using cloud computing is more reliable than local computer.

d) Versatility

Cloud computing doesn't aim at certain special application. It can produce various applications supported by cloud, and one cloud can support different applications running it at the same time.

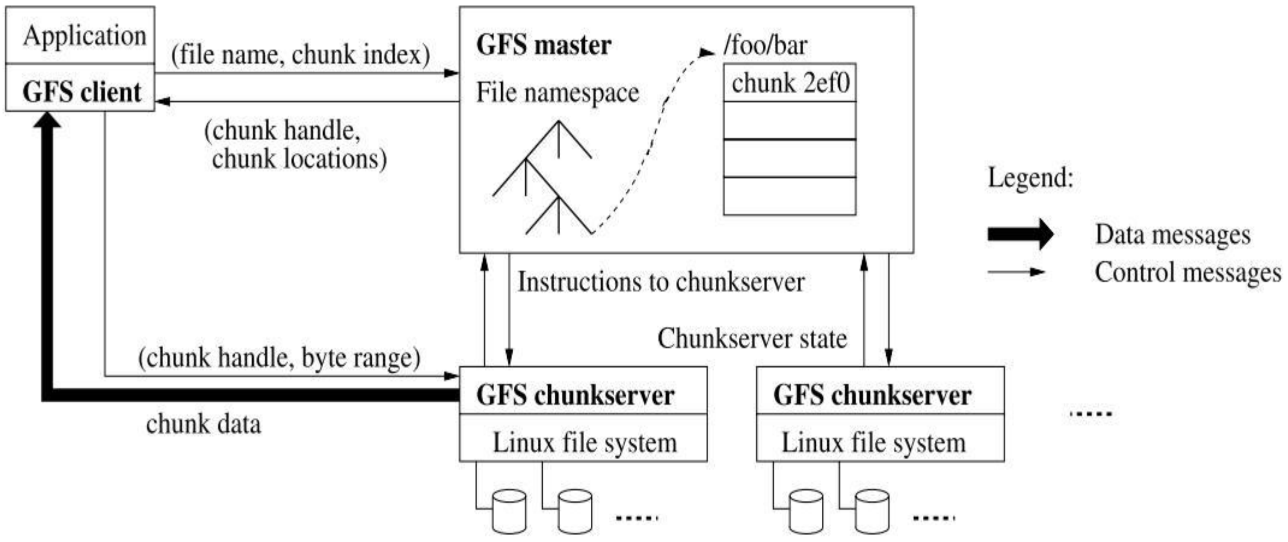

Figure 2. GFS architecture [7]

e) High extendibility

The scale of cloud can extend dynamically to meet the increasingly requirement.

f) On demand service

Cloud is a large resource pool that you can buy according to your need; cloud is just like running water, electric, and gas that can be charged by the amount that you used.

g) Extremely inexpensive

Because the cloud's special fault tolerance can be built by very inexpensive nodes, the centered management of cloud make the enterprise needn't undertake the management cost of data center that increase very fast. The versatility can increase the utilization rate of the available resources compared with traditional system, so users can fully enjoy the low cost advantage. You can spend only a few hundred dollars and a few days to accomplish a task that you must do it spending thousands of dollars and several months before.

III. KEY TECHNIQUES OF CLOUD COMPUTING

In this section, We would take Google's cloud computing techniques as an example, summed up some key techniques, such as data storage technology (Google File System [7]), data management technology (BigTable [8]), as well as programming model and task scheduling model (Map-Reduce [9]), used in cloud computing.

A. Google File System (GFS)

Google File System (GFS) is a proprietary distributed file system developed by Google Inc. for its own use. It is designed to provide efficient, reliable access to data using large clusters of commodity hardware.

GFS is optimized for Google's core data storage and usage needs (primarily the search engine), which can generate enormous amounts of data that needs to be retained; Google File System grew out of an earlier Google effort, "Big Files", developed by Larry Page and Sergey Brin in the early days of Google, while it was still located in Stanford. Files are divided into chunks of 64 megabytes, which are only extremely rarely overwritten, or shrunk; files are usually appended to or read. It is also designed and optimized to run on Google's computing clusters, the nodes of which consist of cheap, "commodity" computers, which means precautions must be taken against the high failure rate of individual nodes and the subsequent data loss. Other design decisions select for high data throughputs, even when it comes at the cost of latency. Fig.2 shows the architecture of GFS.

The nodes are divided into two types: one Master node and a large number of Chunkservers. Chunkservers store the data files, with each individual file broken up into fixed size chunks (hence the name) of about 64 megabytes, similar to clusters or sectors in regular file systems. Each chunk is assigned a unique 64-bit label, and logical mappings of files to constituent chunks are maintained. Each chunk is replicated several times throughout the network, with the minimum being three, but even more for files that have high demand or need more redundancy.

The Master server doesn't usually store the actual chunks, but rather all the metadata associated with the chunks, such as the tables mapping the 64-bit labels to chunk locations and the files they make up, the locations of the copies of the chunks, what processes are reading or writing to a particular chunk, or taking a "snapshot" of the chunk pursuant to replicating it (usually at the instigation of the Master server, when, due to node failures, the number of copies of a chunk has fallen beneath the set number). All this metadata is kept current by the Master server periodically receiving updates from each chunk server ("Heart-beat messages").

Permissions for modifications are handled by a system of time-limited, expiring "leases", where the Master server grants permission to a process for a finite period of time during which no other process will be granted permission by the Master server to modify the chunk. The modifying chunkserver, which is always the primary chunk holder, then propagates the changes to the chunkservers with the backup copies. The changes are not saved until all chunkservers acknowledge, thus guaranteeing the completion and atomicity of the operation.

Programs access the chunks by first querying the Master server for the locations of the desired chunks; if the chunks are not being operated on (if there are no outstanding leases), the Master replies with the locations, and the program then contacts and receives the data from the chunkserver directly (similar to Kazaa and its supernodes).

As opposed to many file systems, GFS is not implemented in the kernel of an operating system, but is instead provided as a user space library.

B. BigTable

BigTable development began in 2004 and is now used by a number of Google applications, such as MapReduce, which is often used for generating and modifying data stored in BigTable, Google Reader, Google Maps, Google Book Search, "My Search History", Google Earth, Blogger.com, Google Code hosting, Orkut, YouTube, and Gmail. Google's reasons for developing its own database include scalability, and better control of performance characteristics.

A Bigtable is a sparse, distributed, persistent multi-dimensional sorted map. The map is indexed by a row key, column key, and a timestamp; each value in the map is an uninterpreted array of bytes. The data model of Google BigTable shown in Fig.3.

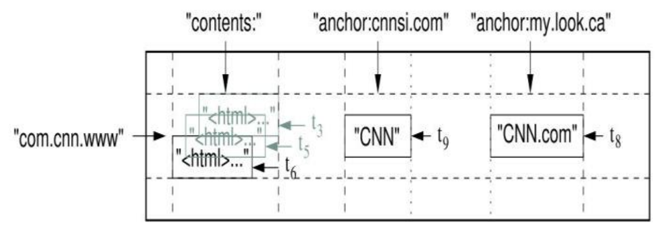

Figure 3. Data model of Google BigTable [8]

Bigtable relies on a highly available and persistent distributed lock service called Chubby [10]. A Chubby service consists of five active replicas, one of which is elected to be the master and actively serve requests. The service is live when a majority of the replicas are running and can communicate with each other. Bigtable uses Chubby for a variety of tasks: to ensure that there is at most one active master at any time; to store the bootstrap location of Bigtable data; to discover tablet servers and finalize tablet server deaths; to store Bigtable schema information (the column family information for each table); and to store access control lists.

Each table has multiple dimensions (one of which is a field for time, allowing for versioning and garbage collection). Tables are optimized for GFS by being split into multiple tablets-segments of the table as split along a row chosen such that the tablet will be 200 megabytes in size. It use a three-level hierarchy analogous to that of a BTree to store tablet location information (Fig.4).

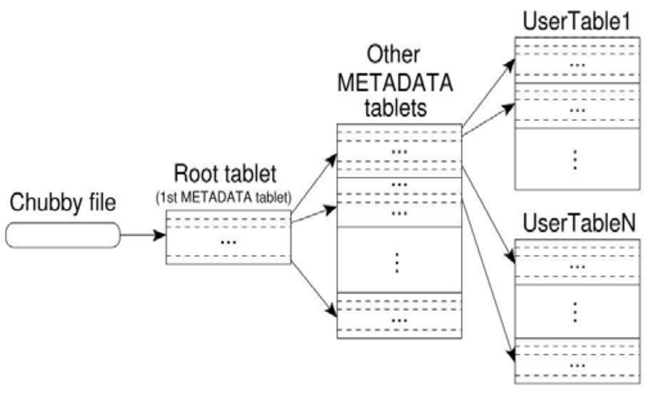

Figure 4. Tablet location hierarchy

The first level is a file stored in Chubby that contains the location of the root tablet. The root tablet contains the location of all tablets in a special METADATA table. Each METADATA tablet contains the location of a set of user tablets. The root tablet is just the first tablet in the METADATA table, but is treated specially—it is never split—to ensure that the tablet location hierarchy has no more than three levels.

C. Map-Reduce Programming Model

Map-Reduce is a patented software framework introduced by Google to support distributed computing on large data sets on clusters of computers. This framework is inspired by map and reduce functions commonly used in functional programming, although their purpose in the Map-Reduce framework is not the same as their original forms. Map-Reduce libraries have been written in C++, C#, Erlang, Java, Python, Ruby, F#, R and other programming languages.

Map-Reduce is a framework for processing huge datasets on certain kinds of distributable problems using a large number of computers (nodes), collectively referred to as a cluster. Computational processing can occur on data stored either in a filesystem (unstructured) or within a database (structured).

"Map" step: The master node takes the input, chops it up into smaller sub-problems, and distributes those to worker nodes. A worker node may do this again in turn, leading to a multi-level tree structure. The worker node processes that smaller problem, and passes the answer back to its master node.

"Reduce" step: The master node then takes the answers to all the sub-problems and combines them in a way to get the output - the answer to the problem it was originally trying to solve.

For example, consider the problem of counting the number of occurrences of each word in a large collection of documents.

The user would write code similar to the following pseudo-code:

```
map(String key, String value):
// key: document name
// value: document contents
for each word w in value:
EmitIntermediate(w, "1");

reduce(String key, Iterator values):
// key: a word
// values: a list of counts
int result = 0;
for each v in values:
result += ParseInt(v);
Emit(AsString(result));
```

The map function emits each word plus an associated count of occurrences (just '1' in this simple example). The reduce function sums together all counts emitted for a particular word. Fig.5 shows the overall flow of a Map-Reduce operation.

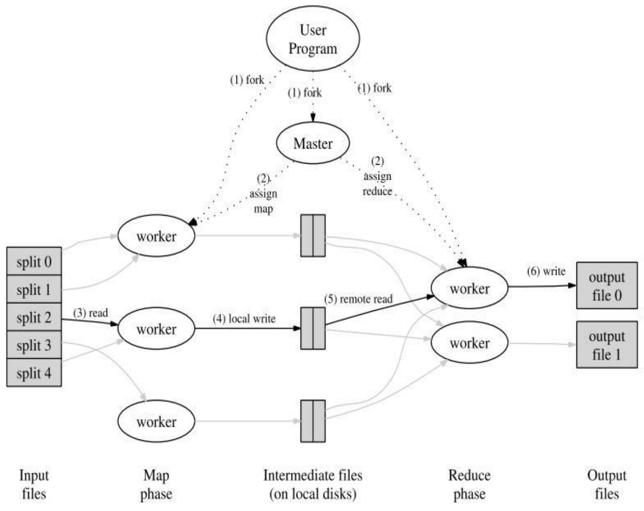

Figure 5. Overall flow of a Map-Reduce operation

The advantage of Map-Reduce is that it allows for distributed processing of the map and reduction operations. Provided each mapping operation is independent of the other, all maps can be performed in parallel - though in practice it is limited by the data source and/or the number of CPUs near that data. Similarly, a set of 'reducers' can perform the reduction phase - all that is required is that all outputs of the map operation which share the same key are presented to the same reducer, at the same time. While this process can often appear inefficient compared to algorithms that are more sequential, Map-Reduce can be applied to significantly larger datasets than "commodity" servers can handle - a large server farm can use Map-Reduce to sort a petabyte of data in only a few hours. The parallelism also offers some possibility of recovering from partial failure of servers or storage during the operation: if one mapper or reducer fails, the work can be rescheduled, assuming the input data is still available.

IV. SOME CLOUD COMPUTING VENDORS

Any application needs a model of computation, a model of storage and, assuming the application is even trivially distributed, a model of communication. The statistical multiplexing necessary to achieve elasticity and the illusion of infinite capacity requires resources to be virtualized, so that the implementation of how they are multiplexed and shared can be hidden from the programmer. Our view is that different utility computing offerings will be distinguished based on the level of abstraction presented to the programmer and the level of management of the resources.

Amazon EC2 [11] is at one end of the spectrum. An EC2 instance looks much like physical hardware, and users can control nearly the entire software stack, from the kernel upwards. The API exposed is "thin": a few dozen API calls to request and configure the virtualized hardware. There is no a priori limit on the kinds of applications that can be hosted; the low level of virtualization—raw CPU cycles, block-device storage, IP-level connectivity—allow developers to code whatever they want. On the other hand, this makes it inherently difficult for Amazon to offer automatic scalability and failover, because the semantics associated with replication and other state management issues are highly application-dependent.

AWS does offer a number of higher-level managed services, including several different managed storage services for use in conjunction with EC2, such as SimpleDB. However, these offerings have higher latency and nonstandard API's, and our understanding is that they are not as widely used as other parts of AWS.

At the other extreme of the spectrum are application domain-specific platforms such as Google App Engine [12] and Force.com, the SalesForce business software development platform. App Engine is targeted exclusively at traditional web applications, enforcing an application structure of clean separation between a stateless computation tier and a stateful storage tier. Furthermore, App Engine applications are expected to be request-reply based, and as such they are severely rationed in how much CPU time they can use in servicing a particular request. App Engine's impressive automatic scaling and high-availability mechanisms, and the proprietary MegaStore (based on BigTable) data storage available to App Engine applications, all rely on these constraints. Thus, App Engine is not suitable for general-purpose computing.

Microsoft's Azure [13] is an intermediate point on this spectrum of flexibility vs. programmer convenience. Azure applications are written using the .NET libraries, and compiled to the Common Language Runtime, a language-independent managed environment. The system supports general-purpose computing, rather than a single category of application. Users get a choice of language, but cannot control the underlying operating system or runtime. The libraries provide a degree of automatic network configuration and failover/scalability, but require the developer to declaratively specify some application properties in order to do so. Thus, Azure is intermediate between complete application frameworks like App Engine on the one hand, and hardware virtual machines like EC2 on the other.

Table I summarizes how these three classes virtualize computation, storage, and networking.

TABLE I. EXAMPLES OF CLOUD COMPUTING VENDORS

	Amazon Web Services	Microsoft Azure	Google App Engine
Computation model (VM)	•x86 Instruction Set Architecture(ISA) via Xen VM •Computation elasticity allows scalability, but developer must build the machinery, or third party VAR such as RightScale must provide it	•Microsoft Common Language Runtime (CLR) VM; common intermediate form executed in managed environment •Machines are provisioned based on declarative descriptions (e.g. which "roles" can be replicated); automatic load balancing	•Predefined application structure and framework; programmer-provided "handlers" written in Python, all persistent state stored in MegaStore (outside Python code) •Automatic scaling up and down of computation and storage; network and server failover; all consistent with 3-tier Web app structure
Storage model	•Range of models from block store (EBS) to augmented key/blob store (SimpleDB) •Automatic scaling varies from no scaling or sharing (EBS) to fully automatic (SimpleDB, S3), depending on which model used •Consistency guarantees vary widely depending on which model used •APIs vary from standardized (EBS) to proprietary	•SQL Data Services (restricted view of SQL Server) •Azure storage service	•MegaStore/BigTable
Networking model	•Declarative specification of IP-level topology; internal placement details concealed •Security Groups enable restricting which nodes may communicate •Availability zones provide abstraction of independent network failure •Elastic IP addresses provide persistently routable network name	•Automatic based on programmer's declarative descriptions of app components (roles)	•Fixed topology to accommodate 3-tier Web app structure •Scaling up and down is automatic and programmer-invisible

V. CONCLUSIONS

Cloud computing is a new technology widely studied in recent years. Now there are many cloud platforms both in industry and in academic circle. How to understand and use these platforms is a big issue. In this paper, we not also described the definition, styles and characters of cloud computing, but also took Google's cloud computing techniques as an example, summed up key techniques, and then some example of cloud computing vendors were illustrated and compared.

Though each cloud computing platform has its own strength, one thing should be noticed is that no matter what kind of platform there is lots unsolved issues. For example, continuously high availability, dealt mechanisms of cluster failure in cloud environment, consistency guaranty, synchronization in different clusters in cloud platform, inter-operation and standardization, the security of cloud platform. These issues mentioned above will be the research hotspot of cloud computing. There is no doubt that cloud computing has a bright future.

REFERENCES

[1] http://searchcloudcomputing.techtarget.com/sDefinition/0,,sid201_gci1287881,00.html

[2] Sims K. "IBM introduces ready–to–use cloud computing collaboration services getclients started with cloud computing," 2007. http://www-03.ibm.com/press/us/en/pressrelease/22613.wss.

[3] Twenty Experts Define Cloud Computing, http://cloudcomputing.sys-con.com/read/612375_p.htm.

[4] R.Buyya, C.S.Yeo, S.Venugopal. "Market-Oriented Cloud Computing: Vison, Hype, and Reality for Delivering IT Services as Computing Utilites," The 10th IEEE International Conference on High Performance Computing and Communications.

[5] UC Berkeley Reliable Adaptive Distributed Systems Laboratory. "Above the Clouds: A Berkeley View of Cloud Computing."

[6] http://baike.baidu.com/view/1316082.htm , 2010.

[7] Ghemawat S, Gobioff H, Leung ST. "The Google file system," In:Proc. Of the 19th ACM Symp. on Operating Systems Principles. New York: ACM Press. 2003. 29-43.

[8] Chang F, Dean J, Ghemawat S, Hsieh WC, Wallach DA, Burrows M, Chandra T, Fikes A, Gruber RE. "Bigtable: A distributed storage system for structured data," In: Proc. of the 7th USENIX Syrup. on Operating Systems Design and Implementation. Berkeley: USENIX Association, 2006. 205-218.

[9] Dean J, Ghemawat S. "MapReduce: Simplified dataprocessing on large clusters," In: Proc. of the 6th Symp. on Operating System Design and Implementation. Berkeley: USENIX Association, 2004. 137-150.

[10] Brrows M. "The chubby lock service for loosely-coupled distributed systems," In: Proc. Of the 7th USENIX Symp. on Operating Systems Design and Implementation. Berkeley: USENIX Association, 2006. 335-350.

[11] Amazon Elastic Compute Cloud, http://aws.amazon.com/ec2/

[12] Google App Engine, https://appengine.google.com/

[13] Microsoft Azure, http://www.microsoft.com/azure/windowsazure.mspx