A Metadata-driven Cloud Computing Application Virtualization Model

Yunpeng Xiao[1,2,*]
1. Chongqing Engineering Laboratory of Internet and Information Security, Chongqing University of Posts and Telecommunications (CQUPT), Chongqing, China
2. Beijing Key Laboratory of Intelligent Telecommunications Software and Multimedia, Beijing University of Posts and Telecommunications (BUPT), Beijing, China
Email: shineagle2005@hotmail.com

Guangxia Xu[1], Yanbing Liu[1] and Bai Wang[2]
1. Chongqing Engineering Laboratory of Internet and Information Security, Chongqing University of Posts and Telecommunications (CQUPT), Chongqing, China
2. Beijing Key Laboratory of Intelligent Telecommunications Software and Multimedia, Beijing University of Posts and Telecommunications (BUPT), Beijing, China
Email: {xugx, liuyb}@cqupt.edu.cn, wangbai@bupt..edu.cn

Abstract—**In order to meet the requirements of standardization of virtualization in cloud computing platform, improve the flexibility and expansibility of the system and enhance the capability of management-control of the platform, by means of introducing the features of decoupling and semantic of metadata, a Metadata-driven Cloud Computing Application Virtualization Model(MCCAVM) in software level is proposed in the paper based on Turing machine model and Von Neumann computer architecture. The model achieves the complete life cycle management of the capabilities and services. Based on the formal definition, analyzing the hierarchical structure with multi-role and multi-dimensional view, the paper proposes a Metadata-driven Cloud Computing Application Virtualization System(MCCAVS). Taking the production of virtual cloud storage service as example, this paper gives formal analysis of system running and compares with other relating work. The results show that the model presents good reference on the construction of cloud computing application virtualization platform.**

Index Terms—**MCCAVM, metadata, cloud computing, application virtualization, software architecture**

I. Introduction

The internet is gradually becoming a kind of computing platform in peace with the rapid expansion and popularization of computer communication technology. As a new computing mode, cloud computing describes a mode of increment, use and delivering for a new type of IT services based on internet. It usually means to apply dynamic scalable and virtual resources through internet[1, 2]. Wikipedia defined cloud computing scenario as follow: Users or clients can submit a task, such as word processing, to the service provider, without actually possessing the software or hardware[3]. This description shows that a core issue of cloud computing research is how to achieve virtualization and large-scale application scalability and availability in the virtual environment.

A broadly understood of virtualization is that computing elements run on the virtual basis. That is a kind of solution to simplify management and optimize resources. The key question highlights platform standardization, improvement of the platform flexibility and dynamic scalability, reduction of the degree of coupling of platform components and other aspects. There are many virtualization technology researches and explorations: Research [4] and [5] put forward virtualization platform architecture through researches based on service-oriented architecture (SOA), [6] and [7] focus on platform flexibility and dynamic capacity expansion, [8, 9, 10] study on virtualization from storage virtualization, virtual device, network virtualization, and other aspects.

Application virtualization uncouples the applications from operating systems, provides a virtual operating environment for the applications. In this environment, not only includes the application executable file, but also includes the runtime environment it requires. In essence, application virtualization is abstracted dependent between low-level application systems and hardware. It can solve the problem of version incompatibility, the limitation of terminal capacity, application system hosting mass, real-time deployment of application, disaster recovery and so on.

Metadata is descriptive information about the data. It is semantics on the basic concepts, basic relationships and basic constraints of data model. The metadata can solve problems that model layer can not resolve, such as fuzzy semantic of data model, model integration and sharing of information. By using metadata we can translate

Manuscript received August 20, 2012; revised October 8, 2012; accepted October 14, 2012.
Corresponding author: Yunpeng Xiao (xiaoyp@cqupt.edu.cn).

functional strong coupling relationships into data type weak coupling relationships. Metadata research is widely used in data-driving system such as file system, information system and so on [11, 12]. On the other hand, the abstract computational model: Turing machine, which can simulate any human computing process, is equivalent to any finite mathematics of logic process. The Turing machine is also a general-purpose computer and an ideal model of universal definition. Its abstract definition is a kind of mathematical logic machine [13]. Von Neumann implemented the ideal model, designed store-based computer architecture [14]. The Von's thinking is inherited by modern computer architectures, since its clear structure and feasibility [15].

Taking into account the needs of application virtualization mentioned above, and by means of metadata, Turing machine and Von Neumann architecture, a Metadata-driven Cloud Computing Application Virtualization Model(MCCAVM) in software level is proposed in this paper. A Metadata-driven Cloud Computing Application Virtualization System (MCCAVS) is implemented based on MCCAVM by then. We make the following three major contributions: 1) Metadata is used to drive the whole system, so that the description of the system is standard and uniform. The decoupling purpose is well done besides. 2) We propose an application virtualization model in software level by refering Von Neumann architecture. All kind of applications and services can be managed by software bus. 3) Not only implemented engineering, the entire model and system are defined and verified formally by using Turing machine.

The rest of this paper is organized as follows: after the introduction in Section 1, Section 2 describes the formal definition and gives the system model; Section 3 designs system and explains each module in detail; Section 4 presents our MCCAVS and verifies it engineering and formally respectively; Section 5 concludes this paper.

II. Model

A. Formal Definition

Before giving model description formally, the definition of capability and Service in the model is described firstly.

Definition 1. *Capability*. Any underlying hardware and software resource in the cloud server side.

Definition 2. *Service*. Product generated by a variety of capabilities, through assembling and reprocessing pattern.

In fact, a virtual application in the cloud server side is combined of *Capability* and *Service*. According to Von Neumann architecture, computers must have five basic components: input data and devices, memory program and data memory, data processing computing device, control program execution controller, output device. Drawing on the thinking of the Von Neumann computer architecture, model regards *Capability* and *Service* as an external device. Various external devices mounted to the model via a software bus and driven by model controlled components. The work of the model is based on predefined tasks, makes multiple types of *Capability* and *Service* work together, and provides virtualization technology by using the basis of hardware, software resources and the upper applications supplied by these devices. Regarding *Capability* and *Service* as an external device, the collaborative process can be considered as the calculation of the finite number of steps. Based on formal definition, the paper proposed a Metadata-driven Cloud Computing Application Virtualization Model(MCCAVM) considering Turing machine computation model and the von Neumann computer system structure of these devices.

Definition 3. Formal definition of the model. A Metadata-driven cloud computing application virtualization model can be formalized as a ten-tuple: $T = (Q,\Sigma,\Gamma_1,\Gamma_2,\Gamma_3,\Gamma_4,\delta,q_0,q_a,q_r)$

Q is the set of states;

Σ is the input alphabet, which does not contain a special blank symbol B;

Γ_1 is the capability to take the alphabet, where $B\in\Gamma_1$ and $\Sigma\in\Gamma_1$;

Γ_2 is the service with the alphabet, where $B\in\Gamma_2$ and $\Sigma\in\Gamma_2$;

Γ_3 is the result with the alphabet, where $B\in\Gamma_3$ and $\Sigma\in\Gamma_3$;

Γ_4 is a task with an alphabet, where $B\in\Gamma_4$ and $\Sigma\in\Gamma_4$;

$\delta : Q\times\Gamma^4\rightarrow Q\times\Gamma^4\times\{L,R\}^4$ is the transfer function, where L, R indicates that the read-write head to the left or to the right;

$q_0\in Q$ is the initial state;

$q_a\in Q$ is an accepting state;

$q_r\in Q$ is the denial of state and $q_r\neq q_a$;

Capability alphabet and service alphabet need to be processed were recorded on work tapeΓ_1 and Γ_2; Γ_3 records the results; Γ_4 records work processes alphabet of different virtualization tasks.

Theorem 1. MCCAVM is a general computing model which is equivalent to the universal Turing machine.

Proof: Firstly, in Definition 3, depending on the differences of storage function, defining a number of working tapes. Obviously, MCCAVM is multi-tape Turing machine, and multi-tape Turing machine is equivalent with Turing machine, so MCCAVM is equivalent with Turing machine.

Secondly, we set the capability alphabet, service alphabet, results alphabet and tasks alphabet as T1,T2,T3,T4 in T's each computing step. Turing machine M is a seven-tuple. The current state, the current tape content and the location of the read-write head constitute the pattern of M. The specific calculation process of M is conversion from one pattern to another, based on the conversion rules described in the transition function δ. The essence of the Turing machine is an algorithm or function, given data x, a mapping rule according to the function f, calculating the corresponding f(x), that is, M is equivalent to a dedicated machine for a particular calculation. For a specific task, it completes a specific calculation or mapping process by MCCAVM according to task process.

For a particular task, we can fix an algorithm Mi from process of tasks alphabet T4. The implementation process: Given T1, T2, transferring function δ is first fixed according to algorithm Mi, making T from one pattern to another. Each task corresponds to a process and each process corresponds to a Turing machine. Therefore, the algorithm is equivalent to a Turing machine. The calculation of (T1, T2) inputted on Turing machine Mi and array (T1, T2, T4) on MCCAVM are equivalent. That is, to any input of Turing machine Mi, MCCAVM can emulate the calculations of Mi, MCCAVM equivalent to interpreter for Mi. Therefore, MCCAVM is equivalent to a universal Turing machine.

B. *Role View of Model*

From the perspective of model development and management control, the model role is divided into developers and system operators. Among them, the developers are divided into capability developers and service developers. Developers produce capabilities or services by using interface language. The system operators use the system language to complete the control of model. Interface instruction set is related to interface language, system instruction set is related to system language. We will explain separately bellowed.

Definition 4. Interface language. It is a set of grammar rules facing capability and service developers. Following this rule, developers can operate MCCAVM directly and complete the operating tasks accurately. Interface language enabled developers to manage full life-cycle of capability and service. In order to facilitate developer, interface language uses simple instruction.

Definition 5. Interface command set. This is a set of commands which make MCCAVM to complete all kinds of basic operating actions according to interface language syntax framework. To complete a certain capability or service development tasks, a number of interface commands are combined together according to the workflow. And each command can also be used to carry out specified action. The commands are made up of parameters and interface functions. Interface functions instructs MCCAVM to complete the basic operating actions. Parameter stands for the executing target of operating instructions and the association attributes of operational objectives.

Definition 6. System language. A set of grammar rules which can be discerned and read directly by the central processing unit of MCCAVM. Under the rules of grammar in the system language, each interface command corresponds to a number of system commands.

Definition 7. System commands. Basic system commands set which meet the system language syntax rules. System commands directly relate to a variety of metadata operating, and they are significant minimum driving force of model. Model task input will be turned into system instructions sequence finally.

In the MCCAVM, the interface language is source language and the system language is target language. The interface language is developer-oriented, which presents a simple way and shields complex logic operating involved with metadata in the model. The system language is the model executable language, which can complete internal behaviors with the core of metadata-driven.

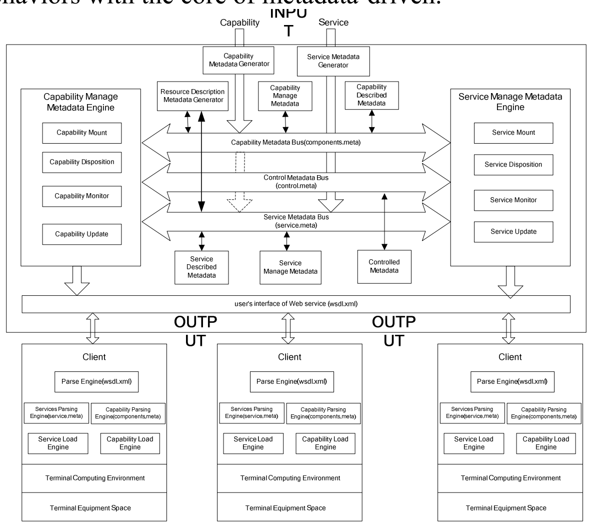

INPU
T
Capability
Service
Capability Metadata Generator
Service Metadata Generator
Resource Description Metadata Generator
Capability Manage Metadata
Capability Described Metadata
Capability Manage Metadata Engine
Capability Mount
Capability Disposition
Capability Monitor
Capability Update
Service Manage Metadata Engine
Service Mount
Service Disposition
Service Monitor
Service Update
Capability Metadata Bus(components.meta)
Control Metadata Bus (control.meta)
Service Metadata Bus (service.meta)
Service Described Metadata
Service Manage Metadata
Controlled Metadata
user's interface of Web service (wsdl.xml)
OUTP
UT
OUTP
UT
Client
Parse Engine(wsdl.xml)
Services Parsing Engine(service.meta)
Capability Parsing Engine(components.meta)
Service Load Engine
Capability Load Engine
Terminal Computing Environment
Terminal Equipment Space

Figure 1. MCCAVM architecture

C. *Model Architecture*

According to the formal definition of the model, model architecture is shown in Fig.1. There are five parts: metadata entities, metadata management engine, bus architecture, the input-output system and the client interface. Referencing to the Von Neumann computer architecture, capability and service in MCCAVM are equivalent to the "peripheral" in computer hardware; the metadata entity is memory and driver of peripheral; Metadata management engines are belonged to the central controller unit. Capability and service are mounted to the corresponding bus through the metadata entity drive and in form of peripheral, completing interaction with metadata management engine. Capability and service are managed and controlled by metadata management engine. The central controller will eventually register service and capability to the user interface and release in a standardized form of web service. The form will be transferred by client, implements transparent access to cloud resources through clients.

1) Metadata entity

The metadata entity is the core part of the model basis. It is generated by metadata generator when Capability or Service enters the MCCAVM. The metadata entity is divided into two categories: descriptive metadata and administrative metadata. Descriptive metadata includes resource description metadata, capability description metadata and service description metadata. Resource description metadata is a summary list of capability and service of the model, describing available resources of the entire model. Administrative metadata includes the capability management metadata, the service management metadata and the control metadata. Metadata management engine controls the capability and service through the administrative metadata.

2) Metadata management engine

Metadata management engine is institution of control and scheduling of model, completes unified monitoring, management and coordination work of capability and service and other resources. Firstly, after capability (or service) enters the model metadata generator. Metadata generator requests metadata management engine to process by using interruption. Then engine mounts the corresponding resource on model bus by using metadata entity, completes the registration task of service and capability, informs MCCAVM that the capability (or service) has been in a state of readiness, then accomplishes work of further assembly and configuration of resource by deploying the capability (or service), coordinates with other related equipments and components in order to make the resource executable; realizes the management and control of model resource through capability (or service) monitoring. For any update of the capability(or service), metadata generator requests engine to process by using the interrupt mechanism similarly. The simple mechanism makes the MCCAVM model have favorable agile characteristic and dynamic extension property.

3) Bus architecture

Important ligament and prominent feature of MCCAVM is bus architecture. In the structure of computer hardware, rational task division of data bus, address bus and control bus promotes module production which is suitable for computer components, boosts the popularity of computers. The bus architecture of MCCAVM is constituted by capability metadata bus, service metadata bus and control metadata bus. Capability metadata bus and service metadata bus finish the carry of service and capability. Control metadata bus transmits control signals, communicate capability management metadata and service management metadata at the same time, makes MCCAVM unified.

4) Input and output interface system of server side

Server-side provides the capability and service for the system by using metadata, through the capability metadata generator and the metadata generator. Capability and service are provided to MCCAVM model in the form of peripheral through description and expansion of metadata. On the output side, further assembly and deployment of the model are provided to the client system in the unified form of web service.

5) Client interface

Client gets kinds of cloud applications from server side by using virtual desktop. Firstly, client virtual desktop gets service list from cloud side. User orders the apps which he/she likes subsequently. The load engine completes the transparent access to capability and service of cloud side at last.

III. System Design

An open, dynamic scalable and data loosely coupled MCCAVS is designed based on MCCAVM in this section.

A. System Architecture

Based on the model design, MCCAVS architecture is shown in Fig.2. Corresponding to the model, the whole system includes metadata entities, metadata management engine, and bus structure, the input and output interface system of server side and client interface. Moreover, capability pool and service stores are implemented, used for storing capability and service.

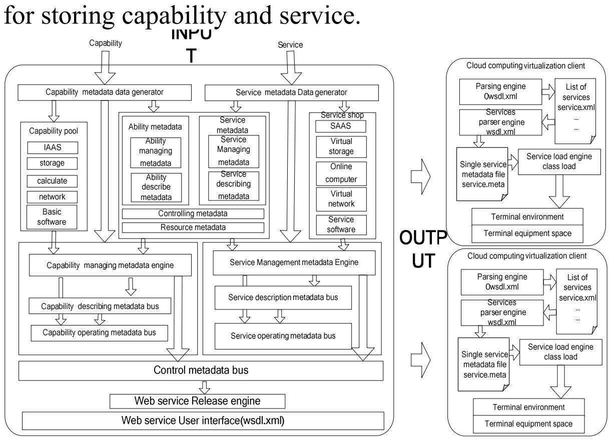

INPUT
Capability
Service
Capability metadata data generator
Service metadata Data generator
Capability pool
IAAS
storage
calculate
network
Basic software
Ability metadata
Ability managing metadata
Ability describe metadata
Service metadata
Service Managing metadata
Service describing metadata
Controlling metadata
Resource metadata
Service shop
SAAS
Virtual storage
Online computer
Virtual network
Service software
Capability managing metadata engine
Capability describing metadata bus
Capability operating metadata bus
Service Management metadata Engine
Service description metadata bus
Service operating metadata bus
Control metadata bus
Web service Release engine
Web service User interface(wsdl.xml)
OUTPUT
Cloud computing virtualization client
Parsing engine 0wsdl.xml
List of services service.xml
Services parser engine wsdl.xml
Single service metadata file service.meta
Service load engine class load
Terminal environment
Terminal equipment space
Cloud computing virtualization client
Parsing engine 0wsdl.xml
List of services service.xml
Services parser engine wsdl.xml
Single service metadata file service.meta
Service load engine class load
Terminal environment
Terminal equipment space

Figure 2. MCCAVS architecture

B. Metadata Entity

In MCCAVS, metadata entity components are series files of system capability and service for describing, controlling and managing (.meta). As the extensible markup language (eXtensible Markup Language, XML) provides standard methods of metadata information exchange methods, we use XML-based metadata file format. Resource metadata files (resource.meta) record all available capability and service in system; Capability description metadata file(capability_description.meta) and capability manager meta data files (capability_manager.meta) record detail information of specific capability; Similarly, service description metadata file (service_description.meta) and service managing data files (service_manager.meta) record detail information of specific service; Control metadata files record permissions related to capability and service, information of roles and life-cycle state control and so on. The resource metadata file (resource.meta) is given as followed for example:

```
<resource_meta>
  <Capability_list>
  <Capability>
    <id>***</id>
    <name>***</name>
    <meta_location>***</meta_location>
  </Capability>
  ......
  </Capability_list>
  <service_list>
  <service>
    <id>***</id>
    <name>***</name>
    <meta_location>***</meta_location>
  </service>
  ......
  </service_list>
</resource_meta>
```

From above we can conclude that resource metadata file includes all the current capability and service list. Each capability or service has a system unique id identifier, which is assigned by the system when this capability or service enters the system and registers to the metadata engine. Based on Von Neumann architecture, we regard capability and service as “peripheral”, thus capability or service id is these “peripherals” address. Metadata management engine, which is the CPU of MCCAVS, can locate any capability or service according to the "peripheral” id through meta-data bus. Individual capability or service metadata description file and path to the file metadata management are stored in the tag meta_location. We take capability described metadata file as an example to expound the entity format of individual capability metadata in the following

```
<Capability_description_meta>
  <attrs>
    <id>***</id>
    <name>***</name>
    <author>***</author>
    <version>***</version>
    <location>***</location>
    <description>***</description>
    <loadclass>***</loadclass>
    <dependentCapability>**</dependentCapability>
    ......
  </attrs>
</Capability_description_meta>
```

Besides containing the capability id and other basic information, capability description metadata file also contains version tag used for controlling capability version information, location tag indicates the real location of the capability products in the capability pool, loadclass tag indicates the entrance classes of the capability part; dependentCapability tag indicates dependency relationship with other capability components.

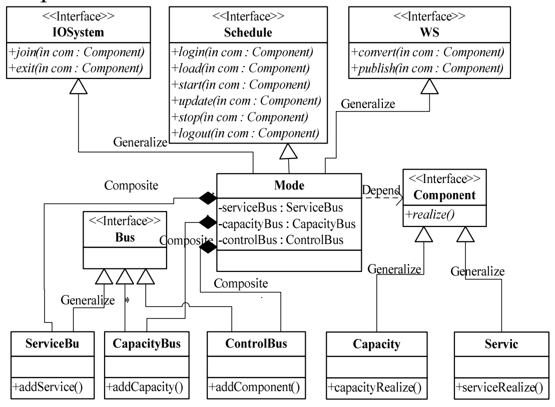


Figure 3. Interface language UML static structure

C. Metadata Management Engine

Metadata management engine, which takes responsible for parsing the interface command and translates it into system commands, is the "central processor" and the core component of the system. We define a set of interface commands based on object-oriented language JAVA to facilitate developers. As shown in fig.3, interface language is divided into three categories: the system input and output (IOSystem), the scheduling interface (Schedule), and the web service interface(WS). Interface command is the abstraction method provided by these interfaces and the command parameter is the method parameter. Table 1 shows a typical interface and the interface commands.

D. Bus Structure

According to the design of the bus structure in the model, three types of buses in the MCCVAS system are defined: capability bus, service bus and control bus. For the purpose of quick addressing and maintaining resources of storage efficiently, HashMap, which can complete key-value mapping and time complexity is O(1), is used to organize system capability and service. Capability or service id is the key and capability or service object instance is the value. Three bus declarations are followed:

protected class ***ControlBus*** *extends HashMap<ID, Component> implements Bus*
protected class ***CapabilityBus*** *extends HashMap <ID, Capability> implements Bus*
protected class ***ServiceBus*** *extends HashMap<ID, Service> implements Bus*

TABLE I.
TYPICAL INTERFACE AND INTERFACE COMMAND

Interface	Interface commands	Parameter	Function Declaration
IOSystem	join()	Component: the Parent interface of Capability and Service	To Generate metadata entity when Capability or Service enters.
	exit()	ditto	To make Capability or Service exits the system.
Schedule	login()	ditto	To allocate component ID when register to the system Capability or Service.
	load()	ditto	To load Capability or Service on the system, which requests metadata management engine mounted components to the corresponding meta- data bus through interrupt mode.
	start()	ditto	To start-up Capability or Service.
	update()	ditto	To update Capability or Service.
	stop()	ditto	To stop Capability or Service.
	logout()	ditto	To log out Capability or Service, log out components, exit the bus system.
WS	convert()	ditto	To generate service.xml and wsdl.xml for Capability or Service.
	publish()	ditto	To publish Capability or Service as a web service.

E. Input and Output Interface in Server Side

Server-side input interface includes capability metadata generator and service metadata generator. It shields metadata manipulation for outside of the system and generates metadata entity as described in Section 3.2. Output interface releases system capability and service to meet the invoking of terminals by using standardization web service interface. As shown in fig.2, we use AXIS2 as release engine.

F. Client Access

As the server-side uses web service technology to provide resources, the system supports transparent access heterogeneous multi-platform capability and service of cloud side. Client interface work steps are as follows:

a. Regarding all cloud resources (Capability and service) in cloud side as the services, and getting the list of services through parsing engine.

b. Users order corresponding services when enter the list of services.

c. User ordering events will trigger service parsing engine and get metadata file of their subscription services.

d. Service load engine loads the service of client components according to service metadata file, achieving transparent accessing to virtual resources of the cloud.

G. Capability Pool and Service Store

Capability pool and service stores are components which storage system peripherals (Capability and service). System bus manages peripheral by using HashMap. Capability or service id is the key and capability or service object instance is the value. Correspondingly, the capability pool and service store are stockpiles of the value. So, here we use two instances of simple data structures (class Set) to implement the two components separately.

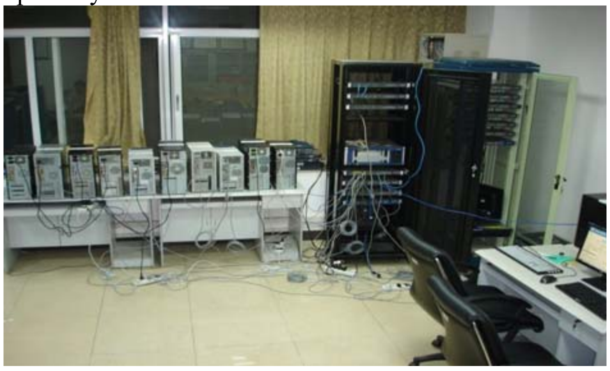


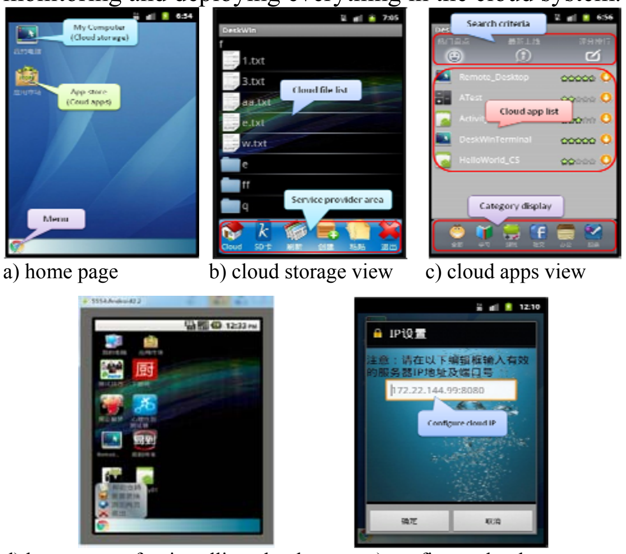
Figure 4. Snapshot of experimental environment

IV. System Implementation and Verification

A. System Implementation

The experimental environment of the system is as follows: The cloud cluster is made up of 14 PCs in which master node uses memory bank of 4G, Intel(r) core(tm)2 duo 2.93GHz, hard disk of 500G and 13 slave nodes are of the same configuration of Pentium(r) dual-core 3.20GHz,use memory bank of 2G,hard disk of 250G. MCCVS prototype system hosted by the master node, as shown in fig.4. The IaaS layer resources and the environment are made up of 13 slave nodes, which have installed Hadoop0.20, choose one as NameNode from the 13 slave nodes. In MCCAVS, as IaaA basic resources made up of 13 slave nodes are regarded as a common "capability" enter system and MCCAVS manages IaaS resources through NameNode, IaaS layer can expand arbitrary amount of nodes at any time according to demand while there is not effect on MCCAVS. The inner bandwidth of the cluster is 100Mbit/s, outlet bandwidth of the server is 10Mbit/s; The operating system is ubuntu11.04, the version of Java virtual machine is Java SE6; The Web container is Tomcat 5.5.17; Client test platform is Android2.2, working in China Mobile EDGE network and using Google nexus s, Samsung and HTC etc. as test termination.

The MCCAVS includes three subsystems, corresponding three user roles: 1) Virtual desktop subsystem, which is client software, corresponding to end user. As shown in fig.5, end user can enjoy cloud storage, browse cloud app list and virtual install apps which he/she likes. 2) Developer subsystem, which is a platform in the cloud side for the developers. As shown in fig.6, developers can upload, submit for review, and release apps. 3) Administrator subsystem, which is a platform in the cloud side for administrator. As shown in fig.7, administrator can use it for checking, configuring, monitoring and deploying everything in the cloud system.



a) home page b) cloud storage view c) cloud apps view
d) home page after installing cloud apps e) configure cloud server

Figure 5. Virtual desktop subsystem

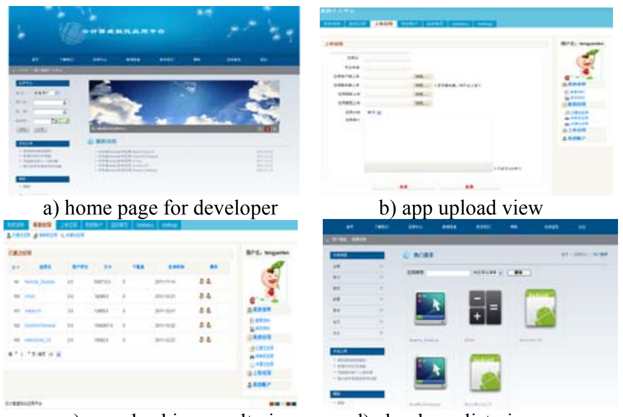

a) home page for developer b) app upload view
c) app checking result view d) cloud app list view

Figure 6. Developer subsystem

a) login page for administrator b) home page

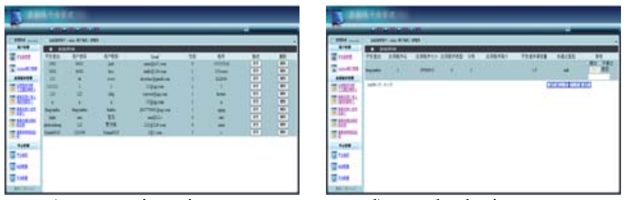


Figure 7. Administrator subsystem

B. System Verification

Due to lack of space, we elaborate formal description and experimental analysis based on Turing machine by using an example of virtual cloud storage services in this section.

1) Service Description

Virtual cloud storage is a destination service system, which makes numerous different kinds of storage equipments in network co-operate and provides data storage and service access function jointly. To the movable termination which is resource-constrained system, virtual cloud storage can extend the storage capability of mobile terminals. In MCCAVS, virtual cloud storage implementation involves four steps: First, the system IaaS layer underlying storage resource (Hadoop HDFS) enters the system as a capability, as shown in fig.2; Then development storage services log-in the system; Moreover, system transfers store service to standard web service and releases it; Finally, the terminal device finds and loads the service, to provide users with a virtual storage service. Using the system interface instruction defined in the 3.3 section, the service specific implementation steps are as follows:

a. In IaaS layer, HDFS file system as a capability to enter the system and invokes interface IOSystem join() method.

b. Based on HDFS, we develop cloud storage control components as service, invoking join() method of IOSystem.

c. Storage capability registers to the system, invokes login() method of Schedule interface.

d. Storage service registers to the system, invokes login() method of Schedule interface

e. Invoke load() method of interface Schedule, load the storage capability.

f. Invoke load() method of interface Schedule, load the storage service.

g. Invoke start() method of interface Schedule to start the storage capability.

h. Invoke start() method of interface Schedule to start the storage service.

i. Invoke convert() method of interface Schedule; make Storage service as standard web service.

j. Invoke publish() method of interface Schedule, release storage web service.

In this section, the order of step a, b is fixed. Step c, e and d, f is the registration and loading of the capability and service, order cannot be changed. The order of process g, h, i, j is also fixed.

2) Formal Verification

According to the description of the services, virtual cloud storage model of the Turing machine can decode the symbol string S=(a b N g h i j), in which N stands for symbol string sequence combination of the fixed alphabetical order c, e and d, f. S is the symbol sequence after the combination of these symbols. Symbols come from a finite alphabet Σ and all the sequences of symbols constitute a language L. Therefore, the problem is transformed to a Turing machine T which can identify the language L. The Turing machine formal description is given in the following:

$T = (Q,\Sigma,\Gamma,\delta,q_0,q_a,q_r)$

$Q = \{q_1,q_2,\ldots,q_8,q_a,q_r\}$

$\Sigma = \{a, b, c, d, e, f, g, h, i, j\}$

$\Gamma = \{a, b, c, d, e, f, g, h, i, j, B\}$

δ：$Q\times\Gamma\rightarrow Q\times\Gamma\times\{L,R\}$ is the transfer function.

$q_0 \in Q$ is the initial state;

$q_a \in Q$ is the accepting state;

$q_r \in Q$ is the reject state and $q_r \neq q_a$;

The initial state of Turing machine is q_0, write symbol sequence S= (a b N g h i j), which needed to be read, on the work tape. Read-write head is scanned from left to right, Each reading of a symbol will trigger a process of metadata management engine, transferring to a new state. That is, transferring from q_0 to q_1 is a→b, R, its state transition function is $\delta(q_0 , a) = (q_1, b, R)$. The service production of the state transition is shown in fig.8.

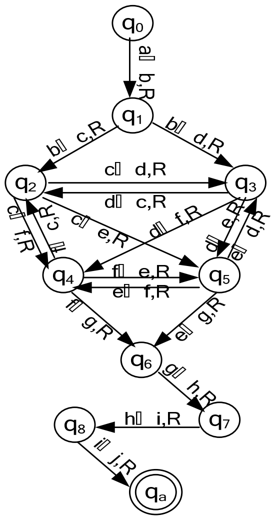


Figure 8. State diagram of virtual storage

3) Experimental Analysis

For virtual storage, cloud capability is much larger than the terminal capability, so, time performance index of the system is more important than the storage capacity index. Fig.9 describes the relationships between capacity and response time when terminal equipment access cloud storage service. Three experimental data in the same test are showed. In every test, besides testing machine, 15 clients are simulated to test system concurrent effect. We can see, response time is mainly determined by the terminal connection bandwidth, system processing time can meet user requirement. On the other hand, virtualization technology will be the main computing tasks hosted by the cloud.

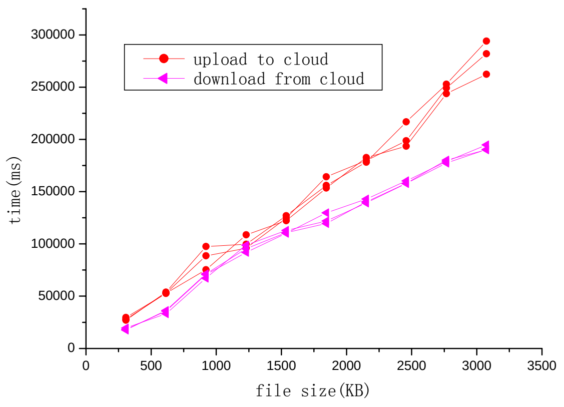


Figure 9. Relationship between cloud storage capacity and response time

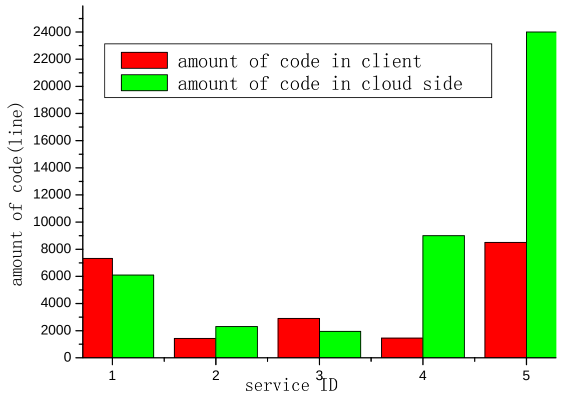


Figure 10. Compare amount of code between cloud and client

Fig.10 compares code quantity of 5 services released in Section 4.1 (by serial number, they are instant messaging[16], cloud storage, campus assistant, online words, mobile TV) in the clouds and the terminal to compare the amount of computations roughly. As the figure shows, numerous computing tasks focus on the cloud by virtualization, even respective code quantity of service terminal is relatively more; it is also be concentrated in user interface to be processed with. In addition, fig.10 shows that when service scale is small, for improving user experience, the calculation of client may closer to the cloud. However, more large-scale applications are more suitable for deployment in cloud computing virtualization platform.

C. Compare with Related Work

In virtualization architecture, typical system is designed by Chinese Academy of Sciences, named Virtual Management Architecture (VMA)[4]. The model aims to establish a unified resource management infrastructure for enterprises to realize the unified management of resources, resource systems and on-demand service of resource. The VMA focuses on management and use of underlying hardware facilities. This is completely different in form and nature with MCCAVM in this paper.

VMA is a resource management framework model, based on virtualization technology and equipped with the technique of independent scheduling, which unified the management and use of interface. VMA is made up of a number of resource management systems; each individual resource management system provides a kind of virtual resource services. VMA provides reasonable and uniform resource management infrastructure of the system by unifying these virtual resources management functions to a unified and consistent management platform.

MCCAVM is a Cloud computing, virtualization model based on Metadata-driven. According to the formal definition of the model, it includes metadata entities, metadata management engine, the bus architecture, input-output system and customer termination. From the perspective of model development and control, besides end users, the model role is divided into two major categories of developers and system operators. Among them, the development is divided into capability developers and service developers. Developers use the interface language to develop capability or service, the system operator uses the system language to complete the model management control task, Table 2 shows the comparison between the two roles.

TABLE II.
COMPARE MCCAVM WITH VMA

	MCCAVM	VMA
Model role	Developers, system operators	Resource users
realization mode	Metadata-Driven	SOA
Virtual level	Hardware resources , software services	Hardware source
objective	Dynamic scalability for capability and service, platform versatility and capability to control	Unified management of basic resources , on-demand use
Hierarchy	Five parts: Metadata entities ,metadata management engine, bus architecture, input, output systems and customer-side interface	Consists of one or more virtual resources management services (VMS) and a system of registration and inquiry services (SRCS)

From the table above we can conclude that VMA focuses on the basic integration of resources and rational management. It is equivalent to an integration of resources and scheduling platform, which can achieve the efficient use and reasonable allocation of resources. MCCAVM achieves a reasonable distribution of resources, and forms a broader perspective to understand the connotation and extension of the capability and service. Through fig.2, we can find that the capability includes not only just basic resources in IaaS layer of cloud computing, but all the basic hardware and software in MCCAVS. At the same time, all the production based on the capability or reproduced through the combination of capability are all service, which reflects the idea EaaS(Everything as a Service). And the model realizes the dynamic expansion of service and capability and hot-swappable, which further enhances the versatility and scalability of the model.

V. Conclusion

This paper proposes a metadata-driven cloud computing virtualization model MCCAVM, based on the discussion and analysis of real needs which exists in cloud computing virtualization technology currently. The corresponding system MCCAVS is implemented also. We formally define and verify the model based on data experiments. Compared with previous work, this model has good reference value which mainly reflected in: 1) Architecture which proposed by the model based on the Turing machine model and the Von Neumann computer is clear and simple. Simultaneously, it implements original intention of design efficiently. 2) The concept of cloud computing platform capability and service with a broader perspective is proposed. The full life cycle management of capability and service is implemented. The model regards capability and service as "peripheral", making the model has a good dynamic scalability. 3) The form of a metadata-driven not only make the model has management-control ability, a simple mechanism and versatility, but also transform the traditional strong function coupling relationship to loosely data coupled relationship of the various components in the model, which plays an important role in the decoupling.

Acknowledgment

This research is supported by import National Science and Technology Specific Project under grants of 2011ZX03002-004-03, Science & Technology Research Program of the Chongqing Municipal Education Committee under grants of KJ110529, Special Foundation of Cloud Computing of Chongqing University of Posts and Telecommunications A2010-13, Educational Reform Projects of Chongqing University of Posts and Telecommunications XJG1216.

References

[1] Gartner.com. Gartner Say's Cloud Computing Will Be As Influential As E-business. http://www.gartner.com/it/page.jsp?id=707508. Aug 2010.
[2] Eric K, Galen G. What cloud computing really means. http://www.infoworld.com/d/cloud-computing/what-cloud-computing-really-means-031. InfoWorld. June 2008.
[3] WikiMedia. Cloud Computing. http://en.wikipedia.org/wiki/Cloud_computing. last modified, June 2011.
[4] WANG Min, LI Jing, FAN Zhong-Lei, XU Lu. A Service Model for Virtual Resource Management and Its Implementation. Chinese Journal of Computers, 2005, 28(5): 856-863.
[5] Kessler M, Reifert A, Lamp D, Voith T. A Service-Oriented Infrastructure for Providing Virtualized Networks. Bell Labs Technical Journal, 2008, 13(3): 111-127.
[6] Bhattacharya K, King DJ. Interview with Douglas J. King on "The Impact of Virtualization and Cloud Computing on IT Service Management". Business & Information Systems Engineering, 2011, 3(1): 49-51.
[7] TIAN Guan-Hua, MENG Dan, ZHAN Jian-Feng. Reliable Resource Provision Policy for Cloud Computing. Chinese Journal of Computers, 2010, 33(10): 1859-1872.
[8] Flouris MD, Lachaize R, Chasapis K, Bilas A. Extensible block-level storage virtualization in cluster-based systems. Journal of Parallel and Distributed Computing, 2010, 70(8): 800-824.
[9] HUAI Jin-Peng, LI Qin, HU Chun-Ming. Research and design on hypervisor based virtual computing environment. Journal of Software, 2007, 18(8): 2016-2026.
[10] Baroncelli F, Martini B, Castoldi P. Network virtualization for cloud computing. Annals of Telecommunications-annals Des Telecommunications, 2010, 65(11-12): 713-721.
[11] Xiong J, Hu YM, Li GJ, Tang RF, Fan ZH. Metadata Distribution and Consistency Techniques for Large-Scale Cluster File Systems. IEEE Transactions on Parallel and Distributed Systems, 2011, 22(5): 803-816.
[12] Govedarica M, Boskovic D, Petrovacki D, Ninkov T, Ristic A. Metadata Catalogues in Spatial Information Systems. Geodetski List, 2010, 64(4): 313-334.
[13] Turing A M. On computable numbers, with an application to the Entscheidungsproblem. Proceedings of the London Mathematical Society, 1936, 42(2): 230-265.
[14] Neumann J. First Draft of a Report on the EDVAC. reprinted in full in Stern, N. From ENIAC to UNIVAC: An Appraisal of the Eckert-Mauchly Computers Bedford, Mass.: Digital Press , 1981: 181-246.
[15] ZHANG Tian-Ning, YUN Xiao-Chun, ZHANG Yong-Zheng, MEN Chao-Guang, SUN Jian-Liang. A Model of Network Device Coordinative Run. Journal of Software, 2011, 34(2): 216-228.
[16] Yunpeng Xiao, Yanbin Liu, Shasha Yang, Guangxia Xu. Design and implement of OMS IM system based on cloud computing. Journal of Chongqing University of Posts and Telecommunications(Natural Science Edition), 2010, (4): 468-472.

Yunpeng Xiao, born in 1979, Ph.D. candidate. His research interests include cloud computing, data mining and complex network.

Guangxia Xu, born in 1974, Ph.D., Associate professor. Her research interests include cloud computing and data mining.

Yanbing Liu, born in 1971, Ph.D., professor, Ph.D. supervisor. His research interests include network management and control, strategy and security.

Bai Wang, born in 1962, Ph.D., professor, Ph.D. supervisor. Her research interests include distributed computing and data mining.