2011 12th International Conference on Parallel and Distributed Computing, Applications and Technologies

A Framework for Scheduling Virtual Machines to Support Real-time Services for U-Life Care

Nguyen Trung Hieu, Jin Wang, Sungyoung Lee, and Young-Koo Lee
Department of Computer Engineering
Kyung Hee University, South Korea
{nthieu, wangjin, sylee}@oslab.khu.ac.kr, yklee@khu.ac.kr

***Abstract*—This paper presents an approach for scheduling U-Life care applications in the cloud computing environment based on virtual resources to support real-time services and to improve user Quality of Service (QoS) requirements. We design and develop an architecture called ULC^3 (Ubiquitous Life Care Cloud Computing) that uses virtual resources provided by cloud computing to schedule U-Life care applications. The ULC^3 is based on the concepts of cloud computing and wireless sensor networks. The architecture is very important and necessary to support create virtual clusters dynamically, deploys the required number of virtual machines (VMs) in potential computing resources to meet the application requirements, and to configure with the required software execution environment. Thus, the system can improve computation time, guarantee the QoS, and support real-time services. Finally, the results from the execute applications are provided to the end-users as a service.**

Keywords-Cloud Computing; Cloud Scheduling; Virtual Machine; Scheduling Algorithm; Distributed Resource Allocation

I. INTRODUCTION

Cloud computing can provide resource as services via virtualization technology which provides software environment in the form of virtual machine (VM) [1][2]. In cloud computing, applications with operating system (OS), specific hardware, software, and libraries requirements can be executed in a larger amount of resources by instantiating VMs from a repository so that requirements can be supported. For that reason, to employ VMs as a computing resource can deliver various advantages such as QoS guarantee, performance isolation, easy resource management, and can deploy effective computing environment. With huge resource provided by cloud computing, many U-Life care applications can be completed to meet the deadline and the QoS requirements. Integrating the U-Life care applications in the cloud computing can not only improve the QoS but also be accessible from anywhere at any time, optimal using the best resources for processing complex applications and storage a larger number of data, load balancing, load sharing, helps cutting down costs and minimizing the computation time [3][4].

Scheduling applications in the cloud computing environment based on the virtual resources is one of the core and challenging issues. The main idea behind scheduling applications focuses on how cloud computing can provide virtual resources that is encapsulated in a VM to meet user's application requirements. The user's application requirements consist of OS, computation resources (CPU, memory, storage, and network), and software execution environment (applications, libraries, simulation tools, and services) for application execute that meet real-time and guarantee QoS requirements. For scheduling applications, we find the best of distributed resources from cloud computing environment and dynamic creation of VMs to meet user's application requirements, and then map from application to a VM. In [5], Xuan Lin and et al use the real-time scheduling algorithm to distribute the applications to the best resources in the cluster computing environments.

The existing job scheduling technology such as Sun Grid Engine [6], Cluster Resources Torque [7], and as well as Condor [8] has limited functionality such as only focusing on-demand provisioning of suitable computing resource for job execution, not support real-time scheduling scenario, and not support dynamic deploy and create VMs.

Therefore, how to schedule applications in cloud computing environment to support real-time services and to improve QoS becomes an urgent problem. Many research works have proposed application scheduling methods, but there are some issues that should be addressed for a cloud computing.

First, [9][10][11] propose some approaches for scheduling application to virtual resources provided by cloud computing. They only focus on the useful of resources. In particular, they cannot achieve the real-time tasks such as real-time collect data, real-time schedule application to VMs with OS, computation resources, software execution environment, and real-time delivery services to end users. Additionally, CPU, computation resources, and software execution environment requirement must also be considered within the scheduling algorithm. A schedule VMs approach for the whole cloud computing is needed.

Second, many applications have varying resources demands, for example, an application may request OS, more computation resources, necessary software execution environment, and this resources are encapsulated in a VM that application can run on. Besides, the application may also require real-time scenario and dynamic resource allocation.

To address the above issues, we propose the architecture called ULC^3 in this paper based on the real-time patients' data collected by different sensors and cameras as well as the virtual resources provided by cloud computing. The architecture uses the cloud resources for scheduling the U-Life care applications

978-0-7695-4564-6/11 $26.00 © 2011 IEEE
DOI 10.1109/PDCAT.2011.5

and the applications can be completed to meet the deadline and the QoS requirements. The major contributions are summarized as follows:

- Real-time monitoring and collecting patient's data from sensors and cameras,
- choose the best of resources provided by cloud computing for dynamic create virtual clusters and deploy VMs with OS, computation resources, and software execution environment,
- schedule U-Life care applications to VMs for real-time processing the patient's information,
- real-time delivery U-Life care services to end users via Software as a Service model.

The rest of the paper is arranged as follows: In next section, contains a review of the recent studies. Section III we present overview architecture and scenarios that are used as a running example throughout the paper. Section IV describes the architecture of ULC^3 and its various components. The performance analysis is introduced in Section V. The last Section of this paper presents our conclusions.

II. RELATED WORK

Scheduling distributed VMs in a cloud computing environment [9][12], Thamarai Selvi Somasundaram and et al, and Lizhe Wang and et al present some models for scheduling parallel tasks and VMs based on cloud environment. There problems have similar with our problem, such as scheduling dealing with physical resources, VM resources, and software required. However, they does not consider a real-time requirements such as real-time monitoring and collecting data; choose the best of the resources provided by cloud computing for dynamically create and deploy VMs with OS, computation resources, and software execution environment; processing with real-time data as input, real-time delivery services to end users, and guarantee QoS requirements. In addition, they deployed VMs with software required that pre-installed.

Recently, in [13][14], the authors propose an approach integrate Wireless sensor network with cloud computing. They propose a solution to automate processes for patients' vital data collection by using sensor attached to existing medical equipments. The information becomes available in the cloud. From where it can be processed by expert systems, storage a large of patients' data, security health records, and/or distributed to medical staff for analysis. However, their work don't relate to the challenge problem such as how cloud computing can provide virtual resources for processing and analyzing patients' data, how cloud computing can guarantee QoS requirements and meet the deadline, and how schedule application to the best virtual resources that provide by cloud computing.

Herein, in our work, we consider the real-time and QoS tasks, and focus on monitoring and discovery resources status of the cloud computing environment. We propose Resource Allocation Algorithm to choose the best resources and to determine which physical resources will be selected to create the VMs. Then, we map from u-Life care application to a VM for execution to get the best response time. The cloud collaborator with live migration technology will be used when Cloud Provider can not meet user's application requirements.

III. OVERVIEW ARCHITECTURE AND SCENARIOS

U-Life care applications such as human activity recognize, sensor activity recognize, image processing, sensor signal processing, health monitoring, ... for detecting and analyzing patient's information are complex applications that many computation resources are needed for processing a large amount of data, storage a huge data, in order to minimum response time, reduce costs, and guarantee QoS requirements. Furthermore, each of these applications has different composition, configuration, and deployment requirements.

The overview architecture and the main components of this paper are shown as Figure 1. Communication, Sensing and Security Core is responsible for monitoring and collecting real-time raw patients' data from sensors and cameras. Sensed data is uploaded to cloud computing. In the cloud, this sensed data (sensor, video or audio-data) can be used to as input data of U-Life care applications that are running in VMs.

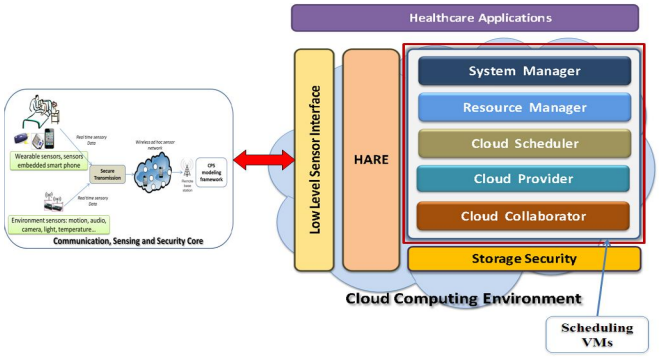

Figure 1. The overview architecture

To run the U-Life care applications in the cloud computing environment, we design and develop ULC^3 architecture. The architecture is very necessary and important because it discover of suitable resources for application execution, choose the best of resources for creating and deploying VMs with OS, computation resources, and software execution environment, schedule applications to the VMs for detecting, analyzing the information of patients, and then real-time delivery healthcare services to end users. Additional, the collaboration between cloud providers or other clouds aim to meet application requirements is presented. We solve the problem about improving U-Life care services by schedule all of U-Life care applications to run in the expert systems to improve computation time, reduce the transmission time, meet the deadline and QoS requirements.

In this work, we focus on ULC^3 that supports two scenarios:

The first scenario, patients' data is real-time collected from sensors and cameras, then being uploaded to the cloud computing environment. Inside the cloud computing, Cloud Scheduler chooses the best of resources and sends the provisioning instruction to Cloud Provider and then Cloud Provider dynamically creates and deploys VMs (from the

Image Repository) with U-Life care application requirements. After finishing successful creates and deploys VMs, Cloud Scheduler maps applications to VMs, so that applications can be executed with real-time data as input and the results will be delivered to end users via Software as a Service,

The second scenario, medical staffs would like to run offline medical applications with patients' data are collected from medical devices. They need many computing resources for processing and the minimum response time is required. They send request to the cloud computing with computing resource requirements. The large amount of resources can be provisioned and made available for applications execution that is encapsulated in the VMs.

IV. ULC^3 – U-LIFE CARE CLOUD COMPUTING

As shown in Figure 2, the ULC^3 framework is designed. It's implemented as a scheduler that responsible for mapping and managing applications run in VMs and schedule U-Life care applications to available cloud computing resources. We describe various components of ULC^3 to support real-time services and guarantee QoS requirements for U-Life care in the following subsections.

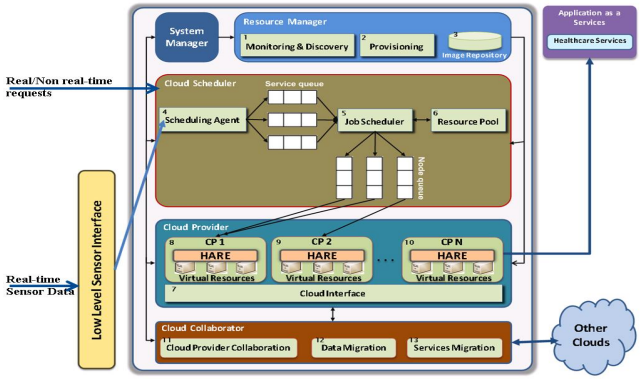

Figure 2. Scheduling virtual machines model

A. System Manager

Manage all of components in ULC^3. It provides functionality for managing the cloud environment.

B. Resource Manager

The Monitoring and Discovery service is responsible for real-time monitoring up-to-date and accurate information of the cloud computing environment.

Provisioning component provides a provisioning policy for allocating processing VMs to physical machine.

To reduce the time to download image OS and software necessary from the Internet, we used the Image Repository to storage image files, software, and libraries necessary for deploying VMs.

C. Cloud Scheduler

The Cloud Scheduler is a master component which controls various activities of the ULC3 such as receive application request, extract and get resource requirements, select of suitable resource from Resource Pool for creating virtual clusters and VMs, and schedule applications to run on VMs.

All real-time jobs which come to the system first enter a Scheduling Agent. Depending on the application requirements, Scheduling Agent determines the jobs will be added in a services queue or not.

Based on the list resources from Resource Pool, the Job Scheduler will apply the scheduling algorithm and send provisioning instruction to Cloud Provider for creating VMs, then mapping applications to VMs. One a ready real-time task is already scheduled it will be mapped to a certain available VMs on which it will be executed. Each work VM also has some software that task needs and a node queue that is responsible for the local task scheduling.

Resource Pool is responsible storage information on the current cloud environment such as the number and type of VMs already running in the cloud, the remaining resources available for creating new VMs, and how many resources available in the cloud computing environment.

D. Cloud Provider

The Cloud Provider (CP) is composed by a set of hosts (physical computing nodes in a Cloud), which are responsible for guaranteeing the QoS for running the VMs by supply all their computing needs such as process, store, and analyze huge data. It handles the VMs creation and deploy of specific environment that the applications needed. So, if the CP is unable to provide QoS for running VMs, the dynamic collaboration between Cloud Providers (CPs) is needed.

There are 3 most popular Cloud Interface which are OpenNebula, Nimbus, and Eucalyptus. All of them are providing function for manager, schedulers, create, and deploy VMs.

E. Cloud Collaborator

Collaboration between CP and CP, or our cloud and other Clouds with migration technology offers significant performance and financial benefit such as: improving the ability of delivering SaaS to end users and meeting QoS requirements, and enhancing the resources provisioning from federation [15].

F. HARE

Human Activity Recognize Engine is an example of U-Life care application that running in VMs. HARE can detect the human activity such as eating, sleeping, walking, reading … of patients.

V. PERFORMANCE ANALYSIS

A. Problem Definition

The problem of scheduling VMs on the cloud system can be defined as follows:

- U-Life care application: App = $\{App_i \mid i = 1 \ldots n.\}$

App is the set of n U-Life care application which will be schedule to VMs in cloud computing. Each application App_i has 7 parameters:

- App_i.AppID is the application identification
- App_i.ReqCPU is the CPU capabilities required,
- App_i.ReqMEM is the Memory available required,
- App_i.ReqDisk is the Disk space available required,
- App_i.ReqSoftware is the Software and libraries available required, this can be a set of software and libraries,
- App_i.ReqDTime is the Deadline required. It's time for application requirement that need for completing task.
- App_j.AppStatus = {Free, Allo, Sche, Wait, Exec, Comp} is the application status.

- Cloud Provider: $Cp = \{Cp_j \mid j = 1 \ldots m.\}$

  Cp is the set of m physical resources in the cloud system that is grouped in the CP. Each CP Cp_j has 4 properties:

  - Cp_j.CpID is the CP identification
  - Cp_j.CPUPerformance is the value of CPU performance,
  - Cp_j.MemorySize is the value of memory size
  - Cp_j.DiskSpace is the value of disk space.

- Virtual machines: $Vm = \{Vm_k \mid k = 1 \ldots v\}$

  Vm is the set of v VMs. Each VM Vm_k has 5 properties:

  - Vm_k.VmID is the VM identification
  - Vm_k.CPUCapability is the value of CPU capabilities,
  - Vm_k.MemorySize is the value of memory size,
  - Vm_k.DiskSpace is the value of disk space,
  - Vm_k.Software is the software and libraries installed on the VM, this can be a set of software and libraries.

The roots of the scheduling VMs are find the best value in a set of CPs that meet the application requirements, create and deploy VMs, and schedule applications to VMs. We define the resources allocation as a function **f1: Vm → Cp** which choose the best of resources in Cp(CPU, memory size, and disk resources) and the application scheduling as a function **f2: App → Vm**.

In this paper, we assume that our cloud computing environment have 3 CPs called Cloud Provider-1 (CP-1), Cloud Provider-2 (CP-2), and Cloud Provider-3 (CP-3). Each CP can provide virtual resources that are encapsulated in VMs. The resources provided by CPs depend on the application requirements such as OS, computation resources (CPU, memory, storage, and network) and software execution environment (applications, libraries, simulation tools, and services).

B. Scheduling Algorithm

The main idea of scheduling algorithm in this paper main focus on the two levels scheduling model: u-Life care applications model and Cloud Provider resources model. In the u-Life care applications model, we define the application requirements that provide configuration information for allocation of resources and creation of the VM. In another model, we propose Resource Allocation Algorithm to select the best currently available resources for running VMs in order to minimize the total execution time and to meet QoS requirements. The Resource Allocation Algorithm contains 3 steps: (1) All of applications should be sorted according to the deadline; (2) depending on the application (App) requirements; we implement the shared resource allocation policies by calculate the probability of capacity of each CPs (P_i) that will be assigned to each VM; (3) selection CP = max $\{P_i\}$ (choose CP based on the largest the value of probability of capacity). The probability of capacity P_i can be presented as follows.

$$P_i = \alpha * App.ReqCPU / Cp_i.CPUPerformance + \beta * App.ReqMem / Cp_i.MemorySize + \gamma * App.ReqDisk / Cp_i.DiskSpace \quad (1)$$

In which $\alpha + \beta + \gamma = 1$, and P_i is the probability of capacity when create VM to meet application requirements in Cloud Provider-i (Cp_i). The value of α, β, γ can be set according to real situation which is used to present the important of the various resources. For example, U-Life care applications require real-time and QoS, we set the priority of CPU, memory, and disk space according as follows: the CPU is very important as the first, the second is memory, and the third is disk space. Based on each probability of capacity, the largest value of P_i will be chosen to assign VM to Cp_i. In the same time, many applications are already in the service queue; therefore, after assigned VM to Cp_i, we update the available resources of CP and re-compute the probability of capacity. Here, we only consider computed probability of capacity which meet conditional as follows (2) to avoid creation of a VM that demands more processing power than is available within the CP.

$$App.ReqCPU \leq Cp_i.CPUPerformance$$
$$App.ReqMem \leq Cp_i.MemorySize$$
$$App.ReqDisk \leq Cp_i.DiskSpace$$
$$App.ReqSoftware \subseteq ImageRepository.ListSoftware \quad (2)$$

We shown an example for resource allocation based on the probability of capacity of VM hosted in the CPs. We choose $\alpha = 0.5$, $\beta = 0.3$, and $\gamma = 0.2$. We assume that inside our cloud computing environment, each CP can provide resources are shown in Table I, and the U-Life care application requirements are shown in Table II. The deadline in Table II is the time required for application completed processing.

TABLE I. AVAILABLE RESOURCES ON CLOUD COMPUTING

CPs	Cp ID	CPU(GHz)	Memory(GB)	Disk(GB)
Cp_1(1, 4, 3, 20)	1	4	3	20
Cp_2(2, 2, 2, 10)	2	2	2	10
Cp_3(3, 5, 8, 50)	3	5	8	50

TABLE II. U-LIFE CARE APPLICATION REQUIREMENTS

Application requirements	App ID	CPU (GHz)	Memory (GB)	Disk (GB)	Deadline (Second)
App_1(1, 1, 1, 5, 120)	1	1	1	5	120
App_2(2, 2, 2, 5, 200)	2	2	2	5	200
App_3(3, 3, 4, 10, 400)	3	3	4	10	400
App_4(4, 2, 1, 7, 500)	4	2	1	7	500
App_5(5, 1, 2, 20, 600)	5	1	2	20	600
App_6(6, 1, 2, 10, 800)	6	1	2	10	800

In the first step, we compute the probability of capacity P_i as following (1) with meet conditional as following (2). The results are shown in Table III. The probability P_2 = 0.5 is largest, therefore we choose CP-2 to create VM-1 with specific requirements (1, 1, 5). After creating VM-1 successfully, the available resources of CP-2 will be updated from (2, 2, 10) to (1, 1, 5).

TABLE III. STEP 1- PROBABILITY OF CAPACITY

Probabilities	Cp_1(1, 4, 3, 20)	Cp_2(2, 2, 2, 10)	Cp_3(3, 5, 8, 50)
$P_i = App_1/Cp_i$	0.275	**0.5**	0.1575
$P_i = App_2/Cp_i$	0.5	0.9	0.295
$P_i = App_3/Cp_i$			0.49
$P_i = App_4/Cp_i$	0.42	0.79	0.2655
$P_i = App_5/Cp_i$	0.525		0.255
$P_i = App_6/Cp_i$	0.425	0.75	0.215

In the second step, we re-compute the probability of capacity P_i where i = 2 to 6. The results are shown in Table IV. The probability P_1 = 0.5 is largest, therefore we choose CP-1 to create VM-2 with specific requirements (2, 2, 5). After creating VM-2 successfully, the available resources of CP-1 will be updated from (4, 3, 20) to (2, 1, 15). The scheduling process will be stop when all of applications are scheduled. The results to assign U-Life care applications to CPs that are encapsulated in VMs as shown in Figure 3:

TABLE IV. STEP 2 – NEW PROBABILITY OF CAPACITY

Probabilities	Cp_1(1, 4, 3, 20)	Cp_2(2, 1, 1, 5)	Cp_3(3, 5, 8, 50)
$P_i = App_2/Cp_i$	**0.5**		0.295
$P_i = App_3/Cp_i$			0.49
$P_i = App_4/Cp_i$	0.42		0.2655
$P_i = App_5/Cp_i$	0.525		0.255
$P_i = App_6/Cp_i$	0.425		0.215

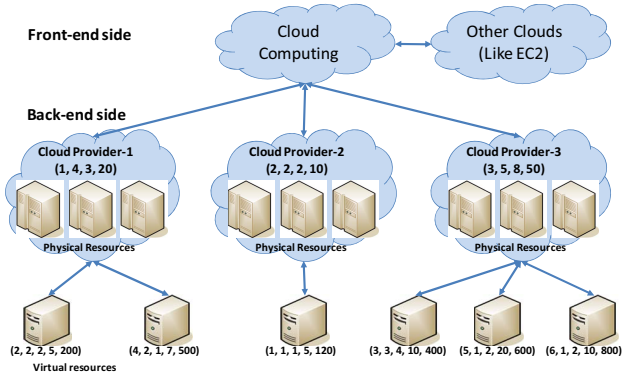

Figure 3. The final results of resource allocation

In the scheduling process, difference applications require difference resources, which require the corresponding VM can change dynamically. Based on the application requirements, we classify into 3 cases:

Case 1: Application requirements are met by Cloud Provider. In this case, the first scheduler creates the application description of a VM including OS required, computation resources required, software execution environment required, and deadline required. Then the second scheduler finds the best resources from the CP for creating VM. Via the two levels scheduling, the application can obtain the required resources. Algorithm-1 briefs the scheduling process adopted for this case.

Case 2: Application requirements are not met by Cloud Provider. In this case, the Cloud Provider collaborator with migration technology will be used to guarantee QoS requirements. In this process, two migration strategies can be taken: (1) the first one is to migrate the VM to another Cloud Provider with the resources available can meet the application requirements, (2) the second one is to allocate the additional required resources in current Cloud Provider for current VM, reconfigure the VM, therefore this VM will be schedule multiple applications run on.

Case 3: Application requirements are not met by a Cloud. In this case, the application requirements which our cloud computing can not provide. Therefore, the collaboration with other Clouds will be used which can provide more resources and software for application requirements.

```
Algorithm-1: Application requirements are met by Cloud Provider
Input: Application requirements and real-time patients' data
Variable: Cp, Vm, App, ListSoftware[] LS
Output: Patients' information
isResource = false; isSoftware = false; isCProvider = 0;
for resource in Cloud Providers do
    if App.ReqCPU ≤ Cp[i].CPUPerformance &&
       App.ReqMEM ≤ Cp[i].MemorySize &&
       App.ReqDisk ≤ Cp[i].DiskSpace then
          P[i] = Probability(App, Cp[i]);
          isResource = true;
    end if
end for
pTemp = P[0];
for Probability in array do
    if pTemp < P[i] then
         pTemp = P[i];
         isCloudProvider = i;
    end if
end for
if isResource == true then
    for software in ListSoftware[] LS do
       if App.ReqSoftware == LS[j].Software then
            isSoftware = true;
        end if
    end for
end if
if isResource == true && isSoftware == true then
        CreateVM(Vm, isCloudProvider);
        BootVM();
end if
```

C. *Scheduling Analysis*

We deploy a cloud environment for experiments using the Linux Ubuntu OS and open source cloud Infrastructure as a Service OpenNebula [16]. The OpenNebula is an open source

toolkit that supports dynamic allocation of VMs in a resource pool, supports live migration of VMs from one hypervisor host to another, and also supports scheduling algorithms through VM placement algorithms. We choose the OpenNebula because it provides policies for VM placement, allowing for custom placement policies to be implemented. The scheduler is based on a Requirement/Rank matchmaker, which allocates computer resources for VMs such as packing (placing as many VMs onto a single node as possible), striping (spreading VMs out across as many nodes as possible), and load-aware (placing VMs on the least heavily-loaded nodes).

To evaluate the VM scheduling, we use one physical machine as the front-end side in which we install the OS Linux Ubuntu 10.04, OpenNebula to manage and schedule VM. The CPU is Intel(R) Core 2 dual @2.67GHz, Memory is 3GB, and Disk capacity is 320GB. Meanwhile, we chose 3 physical machines as back-end side in which we install the OS Linux Ubuntu 10.04, KVM VM. The CPU is Intel(R) Core(TM) i7 CPU 870 @3GHz, Memory is 8GB, and Disk capacity is 640GB. The whole network was connected by LAN.

D. VM Image Analysis

The experiments mainly focus the create VMs with OS, specific hardware, necessary software, and U-Life care application run on. We deploy a VM image with Ubuntu Linux OS from ISO file, u-Life care application is Human Video Activity Recognize with data capture from camera, and some libraries necessary are installed. The OS ISO files, U-Life care applications, and libraries necessary are downloaded from Internet and stored in Image Repository. We use the Ubuntu's vmbuilder tool that allows we build VMs for multiple virtualization techniques.

The Figure 4 is show the result from our implement that are deployed successfully cloud computing environment and VMs in the Cloud Provider with OS, specific hardware, software, and libraries requirements.

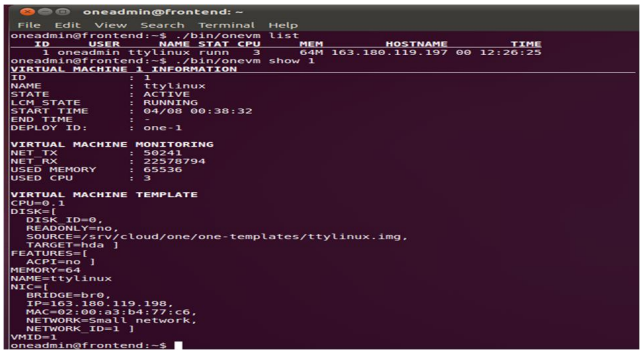

Figure 4. Deploy VM with specific requirements

VI. CONCLUSIONS

Scheduling VMs in the cloud computing environment is a new field and there are many challenges. The proposed architecture aims to overcome the resources limitation when execute U-Life care applications. We focus on the schedule virtual machines to support real-time services for U-Life care, to improve computation time by using power virtual resources provided by cloud computing, and to enhance QoS requirements. Furthermore, we show how cloud computing can provide virtual resources with OS, specific hardware, software and libraries necessary that is encapsulated in a VM to meet the dynamic requirements of users and increases the resource utilization. Dynamic virtual resources collaborate among Cloud Providers and among cloud computing is presented.

ACKNOWLEDGMENT

This research was supported by MKE (Ministry of Knowledge Economy), Korea, under the IT/SW Create research program supervised by the NIPA (National IT Industry Promotion Agency) (NIPA-2010-(C1810-1005-0001)).

REFERENCES

[1] Jinzy Zhu, Cloud Computing Technologies and Application, Hanbook of Cloud Computing, pp. 21-45, 2010.

[2] Shshil Bhardwaj, Leena Jain, and Sandeep Jain, “Cloud computing: A study of infrastructure as a service (IAAS)”, International Journal of Engineering and Information Technology, Vol.2, No.1, 2010.

[3] Jinhua Hu, Jianhua Gu, Guofei Sun, and Tianhai Zhao, “A scheduling strategy on load balancing of virtual machine resources in cloud computing environment”, 3rd International Symposium on Parallel Architectures, Algorithms and Programming, 2010.

[4] Sivadon Chaisiri, Bu-Sung Lee, and Dusit Niyato, “Optimization of resource provisioning cost in cloud computing”, IEEE Transactions on Services Computing, Vol. PP Issues 99, 2011.

[5] Xuan Lin, Anwar Mamat, Ying Lu, Jitender Deogun, Steve Goddard, “Real-time scheduling of divisible loads in cluster computing environments”, Journal of Parallel and Distributed Computing, Vol. 70, Issue 3, pp. 296-308, March 2010.

[6] Sun Grid Engine, “Sun Grid Engine Opensource”, http://www.gridengine.org.

[7] Cluster Resource Torque, “Cluster Resource Torque Opensource”, http://www.clusterresources.com.

[8] Condor, “Condor Opensource”, http://www.cs.wisc.edu/condor.

[9] Thamarai Selvi Somasundaram, Balachandar R. Amarnath, R. Kumar, and et al, “CARE Resource Broker: A framework for scheduling and supporting virtual resource management”, Journal Future Generation Computer Systems, Vol.26, Issue 3, March 2010.

[10] Valentin Kravtsov, Pavel Bar, David Carmeli, and et al, “A scheduling framework for large-scale, parallel, and topology-aware applications”, Journal of Parallel and Distributed Computing, Vol. 70, Issue 9, pp. 983-992, September 2010.

[11] Bo Li, Jianxin Li, Jinpeng Huai, and et al, “EnaCloud: An energy-saving application live placement approach for cloud computing environments”, IEEE International Conference on Cloud Computing, 2009.

[12] Lizhe Wang, Gregor von Laszewski, Marcel Kunze, and Jie Tao, “Schedule distributed virtual machines in a service oriented environment”, 24th IEEE International Conference on Advanced Information Networking and Applications, 2010.

[13] Le Xuan Hung, Sungyoung Lee, Phan Tran Ho Truc, and et al, “Secured WSN-Integrated cloud computing for u-Life care”, 7th IEEE Consumer Communications and Networking Conference, USA, 2010.

[14] Carlos Oberdan Rolim, Fernando Luiz Koch, and at al, “A cloud computing solution for patient's data collection in health care instititions”, Second International Conference on eHealth, Telemedicine, and Social Medicine, 2010.

[15] Mohammad Mehedi Hassan, Biao Song, Eui-Nam Huh, “A market-oriented dynamic collaborative cloud services platform”, Journal of Annals of Telecommunications, Vol.65, No.11-12, pp. 669-688, 2010.

[16] OpenNebula, “OpenNebula Software”, http://opennebula.org/, 2011