An Open Cloud Model for Expanding Healthcare Infrastructure

Sherif E. Hussein
Computer and Systems Department, Mansoura University, Mansoura, Egypt.

Hesham Arafat
Vice Dean of Engineering and Head of Computer and Systems Department, Mansoura University, Mansoura, Egypt.

***Abstract*—with the rapid improvement of computation facilities, healthcare still suffers limited storage space and lacks full utilization of computer infrastructure. That not only adds to the cost burden but also limits the possibility for expansion and integration with other healthcare services. Cloud computing which is based on virtualization, elastic allocation of resources, and pay as you go for used services, opened the way for the possibility to offer fully integrated and distributed healthcare systems that can expand globally. However, cloud computing with its ability to virtualize resources doesn't come cheap or safe from the healthcare perspective. The main objective of this paper is to introduce a new strategy of healthcare infrastructure implementation using private cloud based on OpenStack with the ability to expand over public cloud with hybrid cloud architecture. This research proposes the migration of legacy software and medical data to a secured private cloud with the possibility to integrate with arbitrary public clouds for services that might be needed in the future. The tools used are mainly OpenStack, DeltaCloud, and OpenShift which are open source adopted by major cloud computing companies. Their optimized integration can give an increased performance with a considerable reduction in cost without sacrificing the security aspect. Simulation was then performed using CloudSim to measure the design performance.**

Keywords—Cloud Computing; OpenStack; Openshif; Cloudsim; e-health

I. INTRODUCTION

Although much research effort has been put into the design and development of novel e-Health services and applications, their interoperability and integration remain challenging issues. Health services deal with large amount of private data which needs to be both fully protected and readily available to clinicians. However, health care providers often lack the capability to commit the financial resources for either the research or the infrastructure required. In addition, the public is often skeptical about whether IT systems can be trusted with private clinical information. Past failures have fuelled a culture to restrict any innovation by both members of the public and the politicians responsible for health services [1].

Cloud computing has recently appeared as a new computing paradigm, which promises virtually unlimited resources. Customers rent resources based on the pay-as-you-go model and thus are charged only for what they use. Opposite to other service models, in-house Picture Archiving and Communication System (PACS) and application service provider PACS [2], cloud computing offers relatively lower cost, higher reliability and scalability as shown in table 1.

TABLE I. STRENGTHS AND WEAKNESS OF IN-HOUSE PACS, ASP PACS, AND CLOUD PACS.

Computing Facilities	Service Model	Strengths	Weakness
Dedicated (Local)	**In-House PACS**	**- Smaller network expenses - Control over data - Clear ownership of data - Fast data transmission**	**- Need for in-site IT expertise - Funded out from the investment budget - Need for new technology and upgrading**
Shared (Hosted)	**Application Service Provider PACS**	**- predictable costs - Use of offsite IT expertise -Scalability - Possibility to share data between structures.**	**- Not economical with small examination numbers - Greater network expenses**
Public Cloud	**- IaaS ressources -PaaS development platform - SaaS PACS utilities**	**- Billed per study and/or megabyte - Economical with both small and large examination number - High reliability and Scalability**	**- Slow data transmission -security risk**

The cloud system can be divided mainly into three main layers. The infrastructure as a service (IaaS) is the lowest level which delivers computing infrastructure as a service to end users. IaaS is typically provided as a set of APIs that offer access to infrastructure resources. The APIs allow creating virtual machine instances, or to store or retrieve data from a storage or database. The main benefit of virtualized infrastructure, which is offered as a service, is scalability. The commercial model is to pay for the infrastructure that is actually used. This means the designer doesn't have expensive servers idling, and if there is a spike in the visitor numbers, the designer doesn't have to worry if the hardware will cope. He simply scales the infrastructure up and down as needed. Since the servers are virtual, it's much easier to create a new one than it was to add a new box to a server farm. IaaS provides users with a way to monitor, manage, and lease resources by deploying virtual machine (VM) instances on those resources. Amazon EC2, Eucalyptus, Nimbus, OpenStack and Open

Nebula are examples of cloud infrastructure implementations [3].

In Platform as a Service (PaaS), we move a step further. We no longer have to deal with every element of the infrastructure; instead, we can regard the system as one solid platform. For example, in the case of IaaS, if the designer has a website that suddenly requires more capacity because the amount of visitors increased, he would typically fire up more virtual machine instances. With PaaS, this is no longer the case; the platform will scale up and down as necessary and takes care of the implementation details.

The platform is typically represented as a single box. Since the platform usually acts as if it were a single box, it's much easier to work with, and generally there is no need to change much in the application to be able to run on a PaaS environment. PaaS doesn't only offer cpu, memory or file storage; but also offers other parts of the infrastructure, such as databases, either in the form of a scaling traditional RDBMS system, or one of the 'NoSQL' databases that are currently gaining momentum due to its ability to distribute large amount of data over the cloud infrastructure [4]. The third available service model goes another step further in the realm of abstraction. We no longer care about infrastructure as with IaaS, nor do we care about the platform, as with PaaS. Where in the past, software was installed on the desktop, now with software as a service (SaaS); the user just creates an account and is ready to use the applications, in the comfort of the web browser. As with SaaS, we only need to worry about the application we're dealing with. Good examples of SaaS are cloud PACS utilities which can offer services for imaging centers, reading physicians, primary care clinics, and hospital management [5].

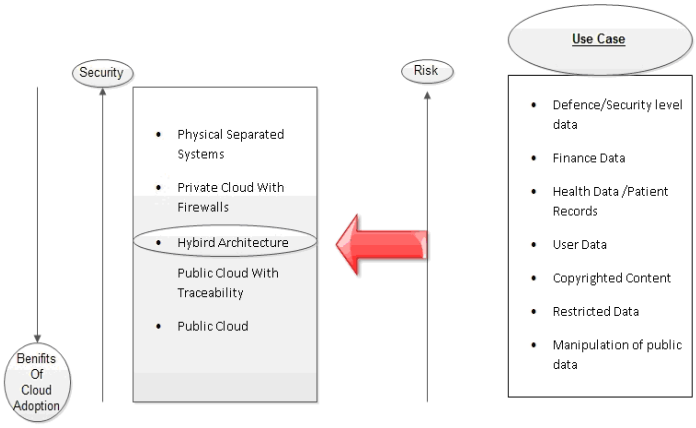

Fig. 1. The relation between security restrictions against the benefits of cloud adoption.

As shown in Figure 1, there is a spectrum of use cases where one end consists of use cases that manipulates public data, hence have very low risk associated with them, while the other end consists of use cases that manipulates credit cards, Social Security Numbers, or ultra sensitive data like nuclear tests.

For the low end of the spectrum, Cloud Computing is an obvious choice, while for the high end; Cloud Computing might never be used. However, instead of public and private cloud, Hybrid architectures provide another choice, a middle ground.

For example, hybrid architecture could move computations to the cloud while keeping sensitive data in a secure database that resides in the private network [6].

In this paper, we provide a closer look into the implementation of healthcare infrastructure using a private cloud based on OpenStack. That was accompanied by setting a PaaS on OpenShift for connecting the private cloud to other health care services, mobile and web applications for clinicians and patients, and resources provided by different cloud providers. That hybrid design was assessed using CloudSim to measure its performance.

The rest of this paper is organized as follow; section 2, introduces a brief note about private cloud implementation, section 3, explains the potentials of expanding the private cloud infrastructure by incorporating other public cloud providers, section 4, discusses a medical application that took advantage of the implemented hybrid cloud, section 5, provides a simulation analysis to assess the performance benefits behind the proposed implementation, and finally we concluded the paper by a discussion section.

II. Private Cloud Implementation

A private cloud implementation aims to avoid many of the objections including control over hospital and patients' data, worries about security, and issues connected to regulatory compliance. Because a private cloud setup is implemented safely within the corporate firewall, it remains under the control of the IT department. However, the hospital implementing the private cloud is responsible for running and managing IT resources instead of passing that responsibility on to a third-party cloud provider. Hospitals initiate private cloud projects to enable their IT infrastructure to become more capable of quickly adapting to continually evolving healthcare needs and requirements.

Launching a private cloud project involves analyzing the need for a private cloud, formulating a plan for how to create a private cloud, developing cloud policies for access and security, deploying and testing the private cloud infrastructure, and training employees and partners on the cloud computing project. To create a private cloud project strategy, a hospital identifies which of its healthcare practices can be made more efficient than before, as well as which repetitive manual tasks can be automated via the successful launch of a cloud computing project. By creating a private cloud strategy, the resulting cloud will be able to deliver automatic, scalable server virtualization, providing the benefits of automated provision of resources and the optimal use of hardware within the IT infrastructure [7]. It is important for the private cloud implementation process to analyze and ensure the proper processes and policies are in place to successfully build a secure private cloud. Research and acquire the private cloud infrastructure and cloud-enabling software that will be used, such as OpenStack, CloudStack, and Eucalyptus. Ensure the hypervisor that will manage the virtual machines and virtualized storage are available or can be purchased and installed.

A. The cloud management software

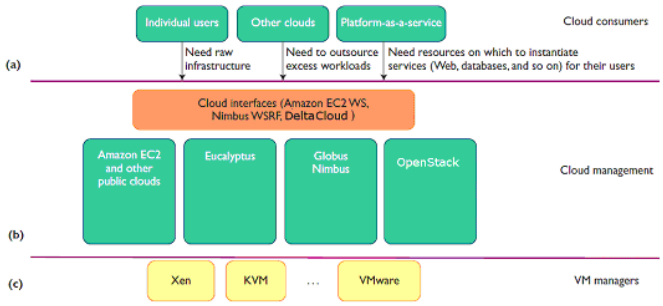

Fig. 2. The cloud ecosystem for building private clouds. (a) Cloud consumers need flexible infrastructure on demand. (b) Cloud management provides remote and secure interfaces for creating, controlling, and monitoring virtualized resources on an infrastructure-as-a-service cloud. (c) VM managers provide simple primitives (start, stop, suspend) to manage VMs on a single host.

Cloud management is the software and technologies designed for operating and monitoring applications, data, and services residing in the cloud as shown in figure 2. Cloud management tools help ensure hospital's cloud computing-based resources are working optimally and properly interacting with users and other services. Cloud management strategies typically involve numerous tasks including performance monitoring (response times, latency, uptime, etc.), security and compliance auditing and management, and initiating and overseeing disaster recovery and contingency plans. With cloud computing growing more complex and a wide variety of private, hybrid, and public cloud-based systems and infrastructure already in use, a hospital's collection of cloud management tools needs to be just as flexible and scalable as its cloud computing strategy. Choosing the appropriate cloud platform, however, can be difficult. They all have pros and cons.

We have decided to compare the capabilities of CloudStack, Eucalyptus, and OpenStack as the most notable open source systems available. Both Eucalyptus and CloudStack's application programming interface (API) provides compatibility with Amazon Web Services' Elastic Compute Cloud (EC2), the world's most popular public cloud. While OpenStack, supports public clouds built by its major vendors. OpenStack is backed by Dell, IBM, RackSpace the second leading IaaS provider after Amazon, NASA, HP the supplier of ARM cloud servers, Canonical the supplier of ubuntu which is tightly integrated with OpenStack in every release and the main operating system for ARM cloud servers [8]. The simple implementation and support of OpenStack for ARM cloud servers favored its choice for our private cloud implementation as it increases the possibility for a lesser cost and an ever growing performance [9, 10].

OpenStack refers to a collection of open-source software packages designed for building public and private clouds. OpenStack is implemented as a set of Python services that communicate with each other via message queue and database.

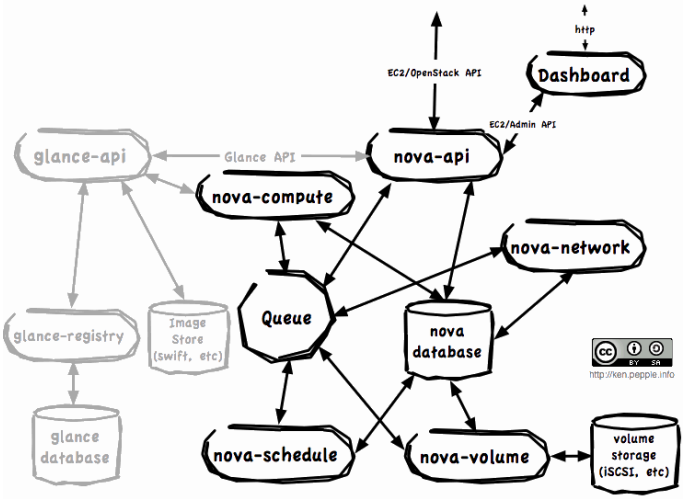

Fig. 3. OpenStack Architecture.

Figure 3 shows a conceptual overview of the OpenStack architecture, with the OpenStack Compute components (which provides an IaaS) bolded and OpenStack Glance components (which stores the virtual machine images) shown in a lighter color. The nova-api service is responsible for fielding resource requests from users. Currently, OpenStack implements two APIs: the Amazon Elastic Compute Cloud (EC2) API, as well as its own OpenStack API [11]. The nova-schedule service is responsible for scheduling compute resource requests on the available compute nodes. The nova-compute service is responsible for starting and stopping virtual machine (VM) instances on a compute node. The nova-network service is responsible for managing IP addresses and virtual LANs for the VM instances. The nova-volume service is responsible for managing network drives that can be mounted to running VM instances. The queue is a message queue implemented on top of RabbitMQ [12] which is used to implement remote procedure calls as a communication mechanism among the services. The database is a traditional relational database such as MySQL used to store persistent data shared across the components. While, the dashboard implements a web-based user interface.

B. The hypervisor

A hypervisor, also called a virtual machine manager, is a program that allows multiple operating systems to share a single hardware host. Each operating system appears to have the host's processor, memory, and other resources all to itself. However, the hypervisor is actually controlling the host processor and resources, allocating what are needed to each operating system in turn and making sure that the guest operating systems (virtual machines) cannot disrupt each other. Most OpenStack development is done with the KVM and XEN hypervisors. KVM however is free and easier to deploy than free XEN, well supported by every major distribution and is adequate for most cloud deployment scenarios (for example, EC2-like cloud with ephemeral storage) while storage support is improving for KVM. It is also simpler and more stable to use with OpenStack and is fully packed by ubuntu 12 Linux that is why KVM was chosen over XEN for our private cloud [13].

C. Legacy migration

Legacy physical server could be migrated into OpenStack cloud using a persistent volume so that the migrated VM is backed by persistent storage. The function provided by the legacy server was the same afterwards and had the manageability benefits from running on an infrastructure platform.

Two things were created: a volume that would contain the data from legacy server and a 'working' VM as a way to work with that volume throughout the migration process. Horizon dashboard has been used to create volumes. The volume size had to be large enough to store all the file systems from legacy system, with appropriate growth, and included any swap-space that might be needed. The 'working' instance was then launched to set up the new volume. This instance had port 22 open under access and security, and ssh key pair was placed onto the system. Then, under 'volumes' the volume was attached to the working instance. The rsync was used to synchronize files between the legacy system and the new VM. The rsync tool was perfect for this for a few reasons: If interrupted, it can resume where it left off. Further, it can be throttled so that data can trickle over to the new volume without impacting the performance of the legacy server. For minimal downtime two separate rsyncs were done. The first one would sync over the vast majority of the data from the legacy server to the volume. The second one was done while both servers were down in order to sync the final changes from the legacy server right before the replacement VM was booted.

The next step was to make the volume bootable by adding a bootloader at the beginning of the volume. OpenStack provides firewalling, so we needed to set up a special security-group for VM that allowed all expected traffic to reach the server. This was done under "Access & Security" in horizon. Before committing to the change-over we had to verify that newly-created bootable volume did indeed work as expected. Once VM was setup correctly, the volume was unmount from the working VM, and legacy server was powered down. New VM system was finally launched.

D. Private cloud security measures

Dome9 secures OpenStack cloud servers and makes them virtually invisible to hackers. That automation closes firewall ports like RDP and SSH, and enables on-demand secure access with just one click. With Dome9, OpenStack Security Groups, centralizing policy management were automated within Dome9 Central. It has as well an SaaS management console for the entire cloud infrastructure. In addition, inter- and intra-group security rules gave ultimate flexibility and granularity. With Dome9 Cloud Connect, setting up Dome9 with OpenStack took less than a minute. Using Dome9 Account, we could add OpenStack supported region, and enter credentials. It was then connected to the private via API to manage all of Security Groups.

III. HYBRID CLOUD IMPLEMENTATION

Public clouds involve the use of third party servers where the user is typically charged on the usage basis. It helps cut the user's capital expenditure significantly, while providing the user with greater flexibility and scalability. However, the advantages of the public cloud come at the cost of poorer performance and increased risks to data and applications. Private clouds attempt to solve the problem by providing cloud installation on-site with better performance and security, coming at increased capital expenditure and reduced flexibility. Therefore, hybrid clouds attempt to bridge the gap by providing the best of both worlds. Enterprises that use the hybrid cloud typically have a private cloud that handles the performance-sensitive core applications, while using the public cloud for scaling and non-core applications. For instance, the mail server and collaboration related components can be kept on the public cloud, while keeping the patient's database and large files in the private cloud. Another reason is that some applications are highly suited for public cloud, while some other legacy applications might not. Private clouds could be less robust than a public cloud managed by a reputed service provider. If the designer have a natural or manmade disaster attacking his hospital site, his private cloud infrastructure might become crippled. The designer can use the public cloud as a fail-over in that case. Thus, the designer might want to have a hybrid approach [14].

Although OpenStack integrates well with almost all IaaS providers, that doesn't stop our implementation from taking full advantages from those IaaS provider that don't support OpenStack. The key was to use a PaaS that could communicate with many different IaaS through a cloud interface. The cloud interface is the holygrail of cloud computing as it bridges the gap between different IaaS suppliers. On the higher level there is a unified API that could be called from a PaaS while on the lower level there are drivers specific to each IaaS and implemented with the vendor without compromising their codes. That cloud interface and the closely integrated PaaS are now reality with deltacloud and openshift thanks to the ever growing competition between cloud services providers to control the cloud computing market [15].

PaaS was initially conceived as a hosted solution for web applications. However the conceptual design forces the Cloud-compliant applications to accept several restrictions: (1) use the APIs exposed by PaaS owners; (2) use the specific programming paradigm that is adequate for the type of applications allowed by the PaaS; and (3) use the programming languages supported by the PaaS owner. These constraints are still valid for most current PaaS offers (e.g., Google AppEngine, Azure, Heroku, Duostack, XAP, Cast, CloudBees, and Stackato). A first step towards the developer freedom in building new PaaS independent applications was done by DotCloud which lets developers build their own software stack for a certain application. This solution is unfortunately not free and is currently based only on EC2 [16].

On the other hand, Deltacloud is an API developed by Red Hat and the Apache Software Foundation that abstracts differences between clouds. So, Deltacloud provides one unified REST-based API that can be used to manage services on any cloud. While each IaaS cloud is controlled through an adapter called "driver" and provides its own API. Drivers exist for the following cloud platforms: Amazon EC2, Fujitsu Global Cloud Platform, GoGrid, OpenNebula, Rackspace, OpenStack, RHEV-M, RimuHosting, Terremark and VMware vCloud. Next to the 'classic' front-end, it also offers CIMI and EC2

front-ends. Deltacloud is used in applications such as Aeolus to prevent the need to implement cloud-specific logic [17].

Deltacloud has some dependencies that need to be installed before its installation as Deltacloud server relies on a number of external rubygems and other libraries. However, once all the dependencies have been installed the Deltacloud server installation is done in one step [18].

OpenShift takes care of all the infrastructure, middleware, and management and allows the developer to focus on what they do best: designing and coding applications. It takes a No-Lock-In approach to PaaS by providing built-in support for Node.js, Ruby, Python, PHP, Perl, and Java. In addition, OpenShift is extensible with a customizable cartridge functionality that allows enterprising developers to add any other language they wish such as Clojure and Cobol. In addition to this flexible, no-lock-in, language approach, OpenShift supports many of the popular frameworks that make development easier including frameworks ranging from Spring, to Rails, to Play. OpenShift is designed to allow Developers to work the way they want to work by giving them the languages, frameworks and tools they need for fast and easy application development [19].

IV. The Medical Image Web Application

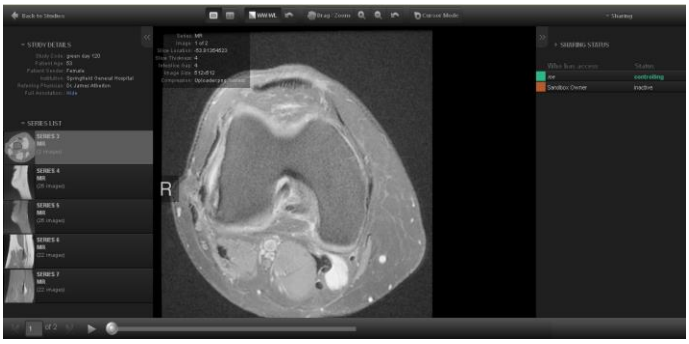

Fig. 4. Web application interface.

Healthcare generates a tremendous amount of data each day (CT, MRI, US, PET, SPECT, Mammography, X-Ray … etc) and consumes quickly the hospitals storage space. For the reason of combining private and public cloud resources, a web application has been designed to store medical images on the private cloud using Mongodb which automatically migrate images older than one month to the public cloud. The subjects information were removed by stripping out the PHI (Protected Health Information) to conform to HIPAA standard and a sharding key was selected for our database that combines exam date and exam id. Compression, upload, delete, retrieval, and viewing were all integrated into the web application. Hibernate OGM; java and SWING were used to manage the backend on OpenShift with the ability to detect which device is trying to access the database. That allows smart phones as well as desktops to access the data for consultation purposes as shown in figure 4. We now detail the design of the Medical Image Web Application (MIWA) to guarantee the Atomicity, Consistency, Isolation, and Durability properties. Each of the properties is discussed individually [20].

A. Atomicity

The Atomicity property requires that either all operations of a transaction complete successfully, or none of them does. To ensure Atomicity, for each transaction issued, MIWA is using shards for actually storing data, mongos processes for routing requests to the correct data, and config servers, for keeping track of the cluster's state. As soon as an agreement to "COMMIT" is reached, the mongos processes can simultaneously return the result to the web application and complete the second phase.

B. Consistency

The consistency property requires that a transaction, which executes on a database that is internally consistent, will leave the database in an internally consistent state. Consistency is typically expressed as a set of declarative integrity constraints. We assume that the consistency rule is applied within the logic of transactions. Therefore, the consistency property is satisfied as long as all transactions are executed correctly.

C. Isolation

The Isolation property requires that the behavior of a transaction is not disturbed by the presence of other transactions that may be accessing the same data items concurrently. The MIWA decomposes a transaction into a number of sub-transactions, each accessing a single data item. Thus, the isolation property requires that if two transactions conflict on any number of data items, all their conflicting sub-transactions must be executed sequentially, even though the sub-transactions are executed in multiple mongos processes.

D. Durability

The Durability property requires that the effects of committed transactions cannot be undone and would survive server failures. In our case, it means that all the data updates of committed transactions must be successfully written back to the back-end cloud storage service. The main issue here is to support mongos processes failures without losing data. For performance reasons, the commit of a transaction does not directly update data in the cloud storage service but only updates the in-memory copy of data items in the shards. Instead, each mongos process issues periodic updates to the cloud storage service. During the time between a transaction commit and the next checkpoint, durability is ensured by the replication of data items across several shards. After checkpoint, we can rely on the high availability and eventual consistency properties of the cloud storage service for durability.

E. Security

Cryptographic modules supplied by HP Atalla were used in the medical image web application to encrypt healthcare data and reduce the risk of data encryption and reputation damage without sacrificing performance using high-performance hardware security modules. Those Data Security solutions meet the highest government and financial industry standards-including NIST, PCI-DSS and HIPAA/HITECH-protect sensitive data and prevent fraud. HP Enterprise Secure Key Manager (ESKM) and Atalla Network Security Processors (NSP) provided robust security, high performance and

transparency while ensuring comprehensive, end-to-end network security.

V. SIMULATION

Evaluation of alternative designs or solutions for Cloud computing on real test-beds is not easy due to several reasons. Firstly, public Clouds exhibit varying demands, supply patterns, system sizes, and resources (hardware, software, network) [19]. Due to such unstable nature of Cloud resources, it is difficult to repeat the experiments and compare different solutions. Secondly, there are several factors which are involved in determining performance of Cloud systems or applications such as user's Quality of Service (QoS) requirements, varying workload, and complex interaction of several network and computing elements. Thirdly, the real experiments on such large-scale distributed platforms are considerably time consuming and sometimes impossible due to multiple test runs in different conditions. Therefore, a more viable solution is to use simulation frameworks which will enable controlled experimentation, reproducible results and comparison of different solutions in similar environments. Despite the obvious advantages of simulation in prototyping applications and developing new scheduling algorithms for Cloud computing, there are a few simulators for modeling real Cloud environments. For evaluating a scheduling algorithm in a Cloud computing environment, a simulator should allow users to define two key elements: (i) an application model specifying the structure of the target applications in Clouds, typically in terms of computational tasks and data communication between tasks; (ii) a platform model of Cloud computing data centers specifying the nature of the available resources and the network by which they are interconnected. Clouds currently deploy wide variety of applications both from industrial enterprises and scientific community [21]. In terms of the platform, Cloud computing is quite different from traditional distributed computing platforms defined by service-oriented features such as resource elasticity, multiple-level of services and multi-tenancy of resources.

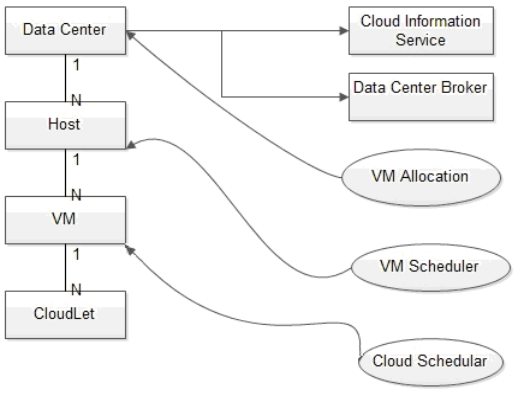

Fig. 5. The main parts and relations of CloudSim.

The experiments in this research were performed on the CloudSim cloud simulator which is a framework for modeling and simulating the cloud computing infrastructures and services [22]. The CloudSim simulator has many advantages: it can simulate many cloud entities, such as datacenter, host and broker. It can also offer a repeatable and controllable environment. And we do not need to take too much attention about the hardware details and can concentrate on the algorithm design. The simulated datacenter and its components can be built by coding and the simulator is very convenient in algorithm design [23]. The main parts which relate to the experiments in this research and the relationship between them are shown in Figure 5 while the functions of those components are explained in table 2.

TABLE II. CLOUDSIM COMPONENTS AND THEIR FUNCTIONS [24]

CloudSim Component	Function
Cloud Information Service	It is an entity that registers, indexes and discovers the resource.
Datacenter	It models the core hardware infrastructure, which is offered by Cloud providers.
Datacenter Broker	It models a broker, which is responsible for mediating negotiations between SaaS and Cloud providers.
Host	It models a physical server.
Vm	It models a virtual machine which is run on Cloud host to deal with the cloudlet.
Cloudlet	It models the Cloud-based application services.
VmAllocation	A provisioning policy which is run in datacenter level helps to allocate VMs to hosts.
VmScheduler	The policies required for allocating process cores to VMs. It is run on every Host in Datacenter.
CloudletScheduler	It determines how to share the processing power among Cloudlets on a virtual machine. It is run on VMs.

The application simulates an IaaS provider with an arbitrary number of datacenters. Each datacenter is entirely customizable. The user can easily set the amount of computational nodes (hosts) and their resource configuration, which includes processing capacity, amount of RAM, available bandwidth, power consumption and scheduling algorithms. The customers of the IaaS provider are also simulated and entirely customizable. The user can set the number of virtual machines each customer owns, a broker responsible for allocating these virtual machines and resource consumption algorithms. Each virtual machine has its own configuration that consists of its hypervisor, image size, scheduling algorithms for tasks (here known as cloudlets) and required processing capacity, RAM and bandwidth.

The simulation scenario models a network of a private and a public cloud (HP's cloud). The public and the private clouds were modeled to have two distinct data centers. A CloudCoordinator in the private data center received the user's applications and processed (queue, execute) them in a FCFS basis. To evaluate the effectiveness of a hybrid cloud in speeding up tasks execution, two test scenarios were simulated: in the first scenario, all the workload was processed locally within the private cloud. In the second scenario, the workload (tasks) could be migrated to public clouds in case private cloud resources (hosts, VMs) were busy or unavailable. In other words, second scenario simulated a CloudBurst by integrating the local private cloud with public cloud for handing peak in service demands. Before a task could be submitted to a public cloud (HP), the first requirement was to load and instantiate the VM images at the destination. The number of images

instantiated in the public cloud was varied from 10% to 100% of the number of hosts available in the private cloud. Task units were allocated to the VMs in the space-shared mode. Every time a task finished, the freed VM was allocated to the next waiting task. Once the waiting queue ran out of tasks or once all tasks had been processed, all the VMs in the public cloud were destroyed by the CloudCoordinator. The private cloud hosted 20 machines. Each machine had 2 GB of RAM, 10TB of storage and one CPU run 1000 MIPS. The virtual machines created in the public cloud were based on an HP's small instance (2 GB of memory, 2 virtual cores, and 60 GB of instance storage). We considered in this evaluation that two virtual cores of a small instance has the same processing power as the local machine. The workload sent to the private cloud was composed of 2,000 tasks. Each task required between 20 and 22 minutes of processor time. The distributions for processing time were randomly generated based on the normal distribution. Each of the 2,000 tasks was submitted at the same time to the private cloud.

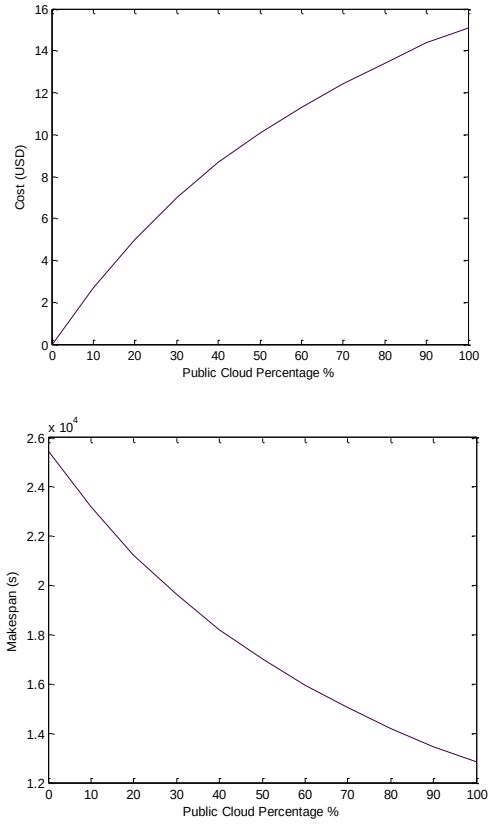

Fig. 6. The relation between cost in USD and the public cloud percentage (up). The relation between makespan in seconds and public cloud percentage (down).

Figure 6 shows the makespan of the tasks that were achieved for different combination of private and public cloud resources. The pricing policy was designed based on the HP's small instances (U$ 0.042 per instance per hour) business model. It means that the cost per instance is charged hourly. Thus, if an instance runs during 1 hour and 1 second, the amount for 2 hours (U$ 0.084) will be charged [25].

This experiment showed that the adoption of a hybrid public/private Cloud computing environments could improve productivity of the healthcare organization. With this model, organizations can dynamically expand their system capacity by leasing resources from public clouds at a reasonable cost.

VI. Discussion and Conclusion

Cloud computing is quickly becoming a dominant model for end-users to access centrally managed computational resources. Through this work in extending OpenStack, we have demonstrated the feasibility of providing healthcare users with access to heterogeneous computing resources using a hybrid cloud computing model.

Open cloud computing is not only a low cost choice for implementation but also provides the designer with a vast number of choices. That low cost solution for resources allocation can solve both the limited storage space and the lack of full utilization of computer infrastructure that healthcare always suffered from. We have to be aware that users' requirements may be very different and so the optimal infrastructure will vary. The ability to select suitable resources from different cloud providers can increase performance and lower cost considerably. That was achieved using Deltacloud and OpenShift which offers a communication layer between web applications, mobile applications and the different cloud providers. Simulation was an important step to measure the feasibility of our design and showed better makespan when public cloud took a higher workload share.

References

[1] David Villegas, Norman Bobroff, Ivan Rodero, Javier Delgado, Yanbin Liu, Aditya Devarakonda, Liana Fong, S. Masoud Sadjadi, Manish Parashar, "Cloud federation in a layered service model," Journal of Computer and System Sciences, Volume 78, Issue 5, September 2012, Pages 1330-1344.

[2] Pierre Boiron, Valère Dussaux, "Healthcare Software as a Service: the greater Paris region program experienc The so-called "Région Sans Film" program," Enterprise Distributed Object Computing Conference Workshops (EDOCW), 2011 15th IEEE International, 29 Aug.- 2 Sep. 2011, pp. 247-251,Paris, France.

[3] P. Sempolinski and D. Thain, "A comparison and critique of Eucalyptus, OpenNebula and Nimbus," in Proceedings of the 2010 IEEE Second International Conference on Cloud Computing Technology and Science (CloudCom), 2010.

[4] Ivo Jansch and Vito Chin. Php/architect's Guide to PHP Development in the Cloud. Blue Parabola, Canada, 2011.

[5] Trieu C. Mohindra, Ajay Karve, Alexei a. Segal, Alla Chieu, "Dynamic Scaling of Web Applications in a Virtualized Cloud Computing Environment," in 2009 IEEE International Conference on e-Business Engineering, 2009, pp.: 281-286.

[6] Perera, S.; Kumarasiri, R.; Kamburugamuva, S.; Fernando, S.; Weerawarana, S.; Fremantle, P., "Cloud Services Gateway: A Tool for Exposing Private Services to the Public Cloud with Fine-grained Control," Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2012 IEEE 26th International, 2012 , pp.: 2237 - 2246.

[7] F. Quesnel and A. L`ebre, "Cooperative dynamic scheduling of virtual machines in distributed systems," in Proceedings of the 6th Workshop on Virtualization in High-Performance Cloud Computing (VHPC '11) Euro-Par 2011, Bordeaux, France, Aug. 2011.

[8] new extreme low-energy server technology. (n. d.). Retrieved from http://h17007.www1.hp.com/us/en/iss/110111.aspx

[9] Open source software for building private and public clouds. (n. d.). Retrieved from http:// www.openstack.org

[10] Corradi, A., Foschini, L. , Povedano-Molina, J. , Lopez-Soler, J.M. , "DDS-Enabled Cloud Management Support for Fast Task Offloading," Computers and Communications (ISCC), 2012 IEEE Symposium on, 1-4 July 2012, pp.: 67 - 74, 2012.

[11] Crago, S.; Dunn, K.; Eads, P.; Hochstein, L.; Dong-In Kang; Mikyung Kang; Modium, D.; Singh, K.; Jinwoo Suh; Walters, J.P., "Heterogeneous Cloud Computing," Cluster Computing (CLUSTER), 2011 IEEE International Conference on, pp.:378 - 385.

[12] RabbitMQ - Messaging that just works (n. d.). Retrieved from http:// www.rabbitmq.com

[13] Kernel Based Virtual Machine (n. d.). Retrieved from http://www.linux-kvm.org

[14] J. Tordsson, R. S. Montero, R. Moreno-Vozmediano, and I. M. Llorente, “Cloud brokering mechanisms for optimized placement of virtual machines across multiple providers,” Future Generation Comp. Syst., vol. 28, no. 2, pp. 358–367, 2012.

[15] J. Lucas-Simarro, R. Moreno-Vozmediano, R. Montero, and I. Llorente, “Dynamic placement of virtual machines for cost optimization in multicloud environments,” in Proceedings of the 2011 International Conference on High Performance Computing & Simulation (HPCS 2011), July 2011, pp. 1–7.

[16] dotCloud - One home for all your apps (n. d.). Retrieved from https://www.dotcloud.com

[17] The Aeolus Project (n. d.). Retrieved from http:// www.aeolusproject.org

[18] Deltacloud (n. d.). Retrieved from http://deltacloud.apache.org

[19] Develop and scale apps in the cloud (n. d.). Retrieved from https://openshift.redhat.com

[20] Genqiang Gu,Qingchun Li,Xiaolong Wen,Yun Gao,Xuejie Zhang, "An overview of newly open-source cloud storage platforms," 2012 IEEE International Conference on Granular Computing (GrC-2012), August 2012, pp. 142-147.

[21] Inçki, K., Ari, I., Sozer, H., "A Survey of Software Testing in the Cloud", Software Security and Reliability Companion (SERE-C), 2012 IEEE Sixth International Conference on, pp.: 18 - 23.

[22] Wei Liu, Feiyan Shi,Wei Du and Hongfeng Li "A Cost-Aware Resource Selection for Data intensive Applications in Cloud-oriented Data Centers" in the International Journal of Information Technology and Computer Science 2011.

[23] The Cloud Computing and Distributed Systems (CLOUDS) Laboratory, University of Melbourne http://www.cloudbus.org/cloudsim

[24] Yuxiang Shi; Xiaohong Jiang and Kejiang Ye, "An Energy-Efficient Scheme for Cloud Resource Provisioning Based on CloudSim," Cluster Computing (CLUSTER), 2011 IEEE International Conference on, 26-30 Sept. 2011, pp.: 595 - 599.

[25] Wickremasinghe, B.; Calheiros, R.N.; Buyya, R., "CloudAnalyst: A CloudSim-Based Visual Modeller for Analysing Cloud Computing Environments and Applications," Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on Communication, Networking & Broadcasting ; Computing & Processing (Hardware/Software), 2010 , pp.: 446 – 452.

[26] HP Cloud Pricing (27 Jan. 2013). Retrieved from https://www.hpcloud.com/pricing#Compute

[27] J. Diaz, G. v. Laszewski, F. Wang, and G. Fox, "Abstract Image Management and Universal Image Registration for Cloud and HPC Infrastructures, " IEEE Cloud. 2012.

[28] Nordin, M.I., Amin, A.H.M., Shah, S.N.M., "Agent based Resource Broker for medical informatics application in clouds", Computer & Information Science (ICCIS), 2012 International Conference on, pp.: 802 – 807, Volume: 2, 12-14 June 2012.

[29] C. Quinton, R. Rouvoy, and L. Duchien, "Leveraging Feature Models to Configure Virtual Appliances," in 2nd International Workshop on Cloud Computing Platforms (CloudCP'12), (Bern, Switzerland), Apr. 2012.

[30] M. Rahman , R. Thulasiram , P. Graham, "Differential time-shared virtual machine multiplexing for handling QoS variation in clouds," Proceedings of the 1st ACM multimedia international workshop on Cloud-based multimedia applications and services for e-health, Nov. 2012, Nara, Japan.

[31] B. Nicolae, G. Antoniu, L. Boug´e, D. Moise, and A. Carpen-Amarie, “BlobSeer: Next-generation data management for large scale infrastructures,” J. Parallel Distrib. Comput., vol. 71, pp. 169–184, February 2011.

[32] Källander K, Tibenderana JK, Akpogheneta OJ, Strachan DL, Hill Z, ten Asbroek AHA, Conteh L, Kirkwood BR, Meek SR, "Mobile Health (mHealth) Approaches and Lessons for Increased Performance and Retention of Community Health Workers in Low- and Middle-Income Countries: A Review," J Med Internet Research, 2013;15(1).

[33] Mell, P.; Grance, T. (2010), "The NIST Definition of Cloud Computing," Communications of the ACM, Vol. 53 (2010) No. 6, pp. 50-50.

[34] von Laszewski, G., Diaz, J., Fugang Wang, Fox, G.C., "Comparison of Multiple Cloud Frameworks," Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on Communication, Networking & Broadcasting ; Components, Circuits, Devices & Systems ; Computing & Processing Hardware/Software, 2012 , pp.: 734 - 741.

[35] Lori M. Kaufman. Data security in the world of cloud computing. IEEE Security and Privacy, 7:61–64, 2009.

[36] Youseff, L.; Butrico, M.; Da Silva, D. (2008), "Toward a Unified Ontology of Cloud Computing," the Grid Computing Environments Workshop, 2008. GCE '08, Austin, Texas, USA, pp. 1-10.