Towards Efficient Automatic Scaling and Adaptive cost-optimized eHealth Services in Cloud

Elie Rachkidi, El Hadi Cherkaoui, Mustapha Ait-Idir, Nazim Agoulmine, Nada Chendeb Taher, Marcelo F. Santos, Stenio Fernandes

Towards Efficient Automatic Scaling and Adaptive cost-optimized eHealth Services in Cloud

Elie Rachkidi, El Hadi Cherkaoui,
Mustapha Ait-idir and Nazim Agoulmine
COSMO, IBISC Laboratory
University of Evry Val d'Essonne
Evry, France

Nada Chendeb Taher
Lebanese University, Faculty of Engineering
and Azm Center for Researches
Tripoli, Lebanon

Marcelo Santos and Stenio Fernandes
Informatics Center (CIn)
Federal University of Pernambuco
Recife, Brazil

***Abstract*—Cloud Computing is an emerging commercial model which allows organizations to eliminate the need to maintain costly hardware, software and network infrastructures. It also permits to avoid the high operational cost for operating and maintaining these infrastructures. Similarly, in the eHealth area, emerging eHealth applications used in conjunction with wearable medical sensor devices and personal devices are being adopted by more and more people with the aim to improve their lifestyle and health. eHealth organizations, willing to provide remote eHealth management, are integrating Wireless Body Area Networks (WBANs) technology and Cloud Computing technology. This integration allows eHealth organizations to deploy their eHealth services on demand and instantly to monitor patients health status. We propose in this paper, a solution for such organizations to efficiently deploy their eHealth services and adapt provisioned physical resources dynamically to satisfy the quality of health of potentially millions of subscribers.**

***Index Terms*—Cloud Computing, eHealth, SLA, Self-adaptive, Auto-scaling**

I. Introduction

Through the use of virtualization and resource time-sharing, Cloud Computing technology offers the possibility to organizations to use IT solutions on-demand. The technology is aimed to be flexible and scalable enough to allow clients to request computing, storage or networking capacities without investing in new infrastructures. In the area of eHealth, wearable medical sensor devices have improved significantly this last decade. They are impacting significantly nowadays peoples daily lives, offering them the possibility to monitor their own health status. These advances in technology are considerably extending the capacity of individuals to take control over their own personal health information. Traditionally, personal medical monitoring systems have been used only to collect data while the data processing and analyzing were performed off-line. This, has been for many years a refrain to the development of eHealth since these devices were impractical for continual monitoring and early detection of medical disorders. Systems with multiple sensors for physical rehabilitation often feature inconvenient wires between the sensors and the monitoring system. These wires eventually limits the patient's activity and level of comfort and thus negatively influence the measured results [1]. However, during the last few years there has been a significant advance in wearables and connected health monitoring devices constituting the so called Wireless Body Area Networks (WBANs). Sensors actually ranges from simple pulse monitors to sophisticated and expensive implantable sensors where health monitoring is performed remotely.

One of the most promising approaches to improve remote health monitoring is indeed the integration of WBAN technology [2] with Cloud Computing technology [3], [4]. The WBAN role will be to sense various physiological parameters (heart rate, blood pressure, oxygen saturation, activity) and/or environmental data (location, temperature, and humidity) and transmit them to the eHealth application via a gateway. A PDA (Personal Digital Assistance) or a smart-phone can play the role of a gateway for the WBAN [5]. It gathers the health data sensed by the WBAN sensors and transmits it to remote health provider(s) server(s) for diagnosis using any available long range communication network (3G, WLAN, GPRS or LTE). These data are stored in the Cloud and queried for more processing [6]. The eHealth applications running in the Cloud permit to analyze the data, diagnose the patient health state, and detect any anomaly.

As the cost of healthcare services increases, it is mandatory that healthcare organizations consider adopting a new implementation models where eHealth applications are hosted in the Cloud. With this model, eHealth Service Providers (HSP) will be able to deploy eHealth services rapidly and on demand.

Hence, they may compose new services from already available service components to speed up the new healthcare service deployment. With this approach, Cloud Providers deploy

eHealth services in their infrastructure and invoke them upon patients' demands.

Service composition and smart placement problems are mainly addressed in Cloud Computing where a requested service is dynamically composed of basic services that are deployed in the network [7]. This approach helps the service provider to make the best choice when deploying customer's services. The main challenges for the Cloud Provider (CP) is to provide dynamically and accurately resources to eHealth services as the demand changes (number of patients, localization, mobility, etc.). Furthermore, optimizing the use of assigned physical resources requires tasks scheduling algorithms [8] in order to distribute optimally the end-users' load. Several authors have proposed solutions to these problems. In [9], authors propose an auto-scaling mechanism which dynamically allocates resources in hybrid clouds for parallel and sequential tasks. They distribute these tasks on different available Virtual Machines (VMs). Authors have considered the execution deadline of the tasks as a decision metric. If the current execution time of one or more tasks has exceeded the estimated time, more resources are allocated. Whereas, if the execution time is respected, unused physical resources are released. However, this approach did not explicitly consider the VM's instantiation and setup time. In [10], Roy N. et al. tried to solve this issue and stated that in order to auto-scale physical resources efficiently in the Cloud, a predictive model is needed. In fact, physical resources allocation needs time (i.e. the same as the state transfer from an old overloaded VM to a newly instantiated one). In this context, they used a second order Auto-Regressive Moving Average (ARMA) filter in order to predict the behavior of the workload for the next time interval. This look-ahead approach enables an early auto scaling mechanism, which allows new VMs to boot and be in a ready state before workload increases. Another predicting based auto-scaling approach is proposed by Yang J. et al. in [11]. Authors used a linear regression model in order to forecast next interval workload. Furthermore, in their approach, authors have considered two ways to scale physical resources, horizontal scaling (add or delete a VM) and vertical scaling (increase or lower physical resources for a given VM). In a second case, authors have adapted physical resources based on both the current workload value and the predicted workload value.

Finally, in [12], authors focused on real time services which must respect a given deadline. Authors considered a healthcare application where transmission rate might vary depending on the status of the patient and where the generated data should be treated within a specified deadline. A moving average filter method is used to predict future system performance in order to scale up physical resources to meet the required workload for future tasks.

However, all these approaches did not take into account the localization on mobility of the patients which we consider as important and highly impacts the performances of the cloud.

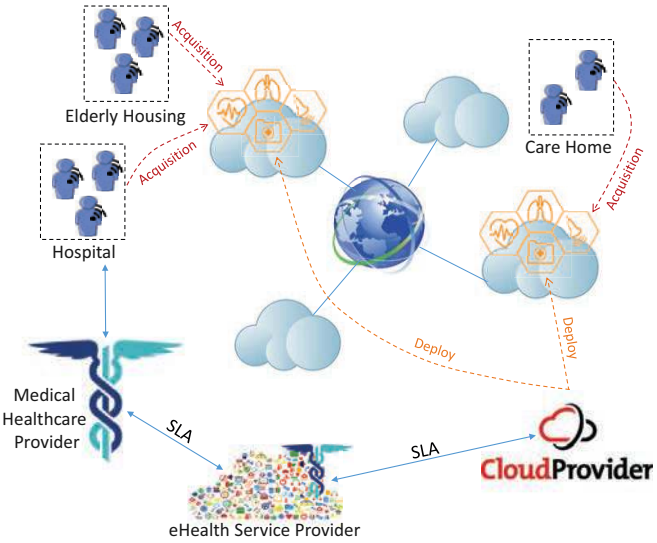

Figure 1. Medical Healthcare Provider Global Architecture

II. Scenario and Problem Statement

A. Scenario

Patients or the medical healthcare provider subscribe to the HSP which provides its services on the Cloud. Similarly, the HSP establishes a contract with the Cloud Provider to define which services to deploy and the QoS terms to be respected. The Service Level Agreement (SLA) should clearly specify the Quality of Service (QoS) objectives with the expected target performances from a virtual service usage. Each party has to implement the SLA using its own tools and processes. In particular, in this work, we are interested on how the automatic provisioning in the Cloud Infrastructure is performed and how to monitor the QoS of the deployed eHealth services while ensuring that SLA's terms are fulfilled. These mechanisms should be able to detect any deviation in the services behavior and automatically trigger particular techniques to maintain the patients eHealth services QoS, either by scaling up the resources in case of increasing workload or by scaling down the resources when the workload decreases. This approach allows efficient use of physical resources and optimal pricing with the "PAYG" (Pay-As-You-Go) [13] for a given SLA.

In Figure 1, we consider a scenario where several patients rely on a medical healthcare provider to monitor their health state in real time. The medical healthcare provider establishes a contract with the HSP in order to define the service(s) to deliver for end-users and the QoS these end-users must experience. Given the system set of specifications, the HSP defines the service(s) to deploy and the SLA terms which ensure the required QoS and sends them to a Cloud Provider. The latter deploys required services and monitors them continuously. The Cloud Provider, upon any violation of SLA terms, must take appropriate actions such as scaling physical resources to ensure expected QoS.

B. Problem Statement

Building a cloud-based platform for personal health sensor data management is costly and may depend on many criteria such as: geographical location, medical facilities and the application services. Many existing CPs (identified as Infrastructure as a Service (IaaS) providers) such as Amazon or Rackspace offer to their customers the possibility to deploy eHealth services in different sites located in different world regions. These sites are different in hardware capacity and support different cloud services. Although the same provider manages these sites, they are independent systems in terms of resources management, scheduling and provisioning. Based on this fact, we consider in this work a similar environment where an HSP could request to its CP a list of medical services to deploy. These services can be modeled as a composition of a service graph request formed by a set of basic services to instantiate (i.e. virtual machines, storage, etc.) and the edges between these nodes (i.e. required network links between the nodes). The deployment of the service request graph will depend on the terms specified in the SLA. These terms can be either localization, QoS or pricing objectives.

In order to satisfy the SLA with the HSP, the CP should solve the three following problems:

1) Find an initial instantiation of eHealth service based on the described demand.
2) Based on the demand changes (number of requests/s) from patient WBANs, or mobility of the patients, how to auto-scale physical resources to satisfy the QoS of eHealth services.
3) Since the instantiation of new VMs (i.e. eHealth service instance) is not instantaneous, how to derive from the mobility pattern a change in the resource demand.

The first problem can be seen as a Facility Location Problem (FLP), particularly under the metric distance. Given a network graph where each node is either a client or a location where a facility may be built, the FLP tries to find a set of locations to set up facilities in order to minimize the total cost. Traditionally, FLP deals with the problem of placing large facilities such as storage, supermarkets, or distribution centers which once installed, could not or should not be removed. Such facilities are much more costly compared to the cost that is needed to serve clients. Thus replacing one facility by another is only considered in extreme conditions. In contrary, service placing strategies in the cloud have to deal with the ever changing environment. While traditional FLP refers to large costly facilities, cloud services can be easily deployed and removed.

In this paper, we focus mainly on the second and third problems that are: given the location of the patients, what is the best provisioning plan that minimizes the cost of deploying service replicas while satisfying the eHealth service QoS level. To address the aforementioned issues, we aim to answer these following questions:

- How to choose a set of locations (provisioning plan) to set up the services that minimizes the total cost, including the installation cost and service delivery cost?
- In case of change in the demands, how to modify the provisioning plan to minimize the total cost while keeping the modification as small as possible?

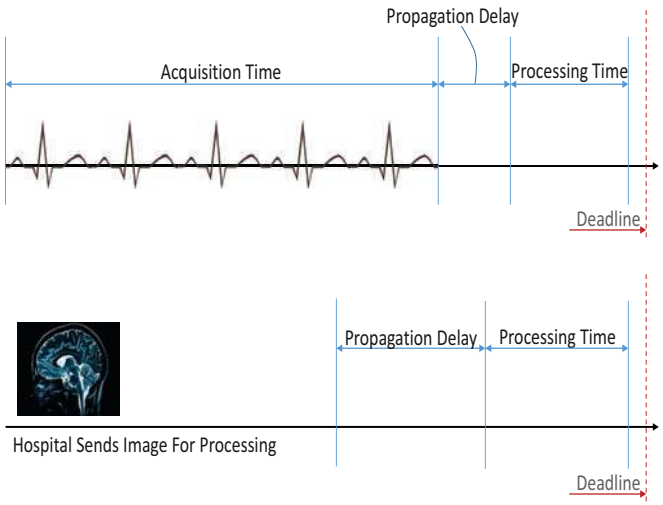

Figure 2. eHleath Service Requests Time Phases

III. Auto-Scaling Mechanism

The unpredictable evolution of medical services usage and patient's mobility could introduce peak loads at different location of the network, which could exceed the computing resources initially allocated to these services (i.e. VM, storage). This may negatively affect the QoS experienced by the end users (i.e. longer response time) and may violate the terms of the agreed SLA between the CP and the patients. In figure 2, we illustrate that each task, such as heart monitoring and analyzing, is affected by three parameters: (a) the acquisition time which is specific to the monitor belonging in the WBAN, (b) the propagation delay which corresponds to the delay between sending the information, and receiving it at the eHealth service side, and (c) the processing time which represents the time needed to finish treating the received information (compute, store, send notifications, etc.). The Cloud Provider needs to optimize the propagation delay and processing time in order to respect the QoS requested by the HSP in case of violations. Therefore all tasks must finish before their assigned deadline.

In order to respect the QoS, while optimizing the physical resources use, the Cloud Provider must auto scale the eHealth services. Services can be scaled horizontally or vertically. Vertical scaling consists of allocating more physical resources to a given running VM and is considered instantaneous, while horizontal scaling is instantiating a new VM to offload some tasks and needs time to be performed. The objective is to respect given deadline while minimizing the deployment cost on the HSP.

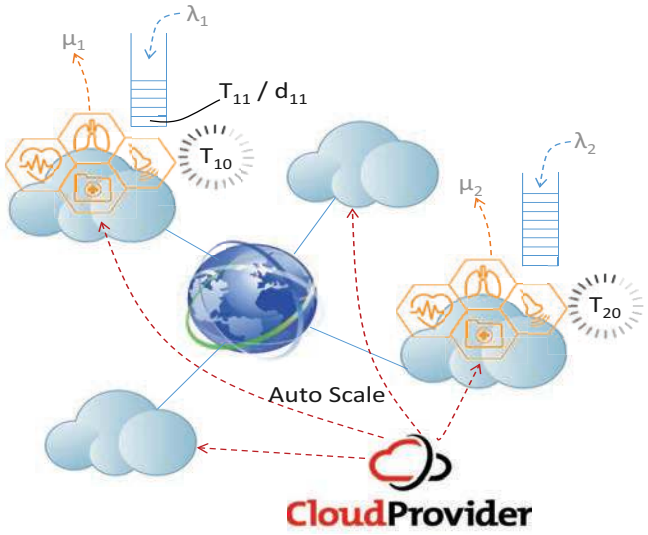

Figure 3. Auto-scaling Mechanism of the eHealth service by the Cloud Provider

A. Service replica placement optimization

Let us consider a connected network graph $G = (N, E)$, where each node $n_j \in N$ is a cloud datacenter. E is the set of edges interconnecting these datacenters. Let s be an eHealth service deployed by the Cloud Provider upon the HSP request and s_j the eHealth service s deployed in data center n_j. The contracted SLA between both parties will define the level of QoS required and how the Cloud Provider will perform auto scaling. We define a set T representing all tasks requested by the eHealth service s. A subset of tasks $T_j \subset T$ include the pending tasks of the eHealth service s_j, and where $T_{j,i} \in T_j$ is the ith task in the queue of s_j. $d_{j,i}$ is the deadline assigned to task $T_{j,i}$.

λ_i and μ_i correspond respectively to the measured request arrival rate and the measured service rate of the eHealth service at datacenter n_i as depicted in Figure 3. Upon the arrival of new requests we sort each queue in descending order with respect to deadline. After sorting queues, the following equation must be satisfied for each data center $n_j \in N$:

$$d_{j,|T_j|} > (1 + \mid T_j \mid) \times \frac{1}{\mu_j} \tag{1}$$

When condition (1) is satisfied, it means that all eHealth services are able to process pending tasks before their deadline. On the other hand, if the condition is not valid, scaling allocated resources is required. In this context, horizontal and vertical scaling can be performed. However, horizontal scaling introduce new VM instantiation and setup delay δ which might be long enough to violate the SLA, but horizontal scaling can decrease propagation delay by replicating the eHealth service closer to the client. In order to benefit from the speed of vertical scaling and location awareness of horizontal scaling we adopt both solutions. If the expected deadline violation occurs in a delay greater than the horizontal scaling completion or local physical resources do not allow vertical scaling, we proceed with horizontal scaling. Otherwise vertical scaling is used to decrease processing time and try meeting QoS terms.

$$(\mid T_i \mid -1) \times \frac{1}{\mu_i} > \delta + \epsilon \tag{2}$$

Where ϵ is the average time to transfer a task from one queue to another.

Condition (2), if valid, means that horizontal scaling can be performed. When horizontally scaling the eHealth service, the host for the replica, is the closest data center to the client who sent the task expected to be violated in terms of execution deadline. If the chosen data center already hosts an eHealth service s, we perform vertical scaling on this service instead of horizontal scaling. To optimize cost, we consider deploying in less costly data centers if several candidates were found for the service s replica. Replicating eHealth services is limited to the maximal cost the medical healthcare provider is willing to pay.

Since we measure the arrival rate λ_i for each service s_i, we can obtain the overall arrival rate λ of the system by adding arrival rates of all services (first equation in 3). The same goes for the service rate (second equation in 3). Knowing the behavior of the overall system, we can judge when to release scaled up physical resources. In fact, we calculate the Average Waiting Time (AWT) of the system considering we undid the last performed scale up (equation 4). If we already performed k scales up with $k > 0$, we calculate the AWT as following:

$$\lambda = \sum_i \lambda_i, \qquad \mu = \sum_i \mu_i \tag{3}$$

$$AWT = \frac{k}{(k-1) \times \mu - k \times \lambda} - \frac{k}{(k-1) \times \mu} \tag{4}$$

if the average waiting time calculated in equation 4 is lower than the highest deadline in the system, the system is able to deliver required QoS if some physical resources are released. We perform a scale down in this case and undo the last performed scale up. However, in the case of horizontal scaling down, we do not remove the last created VM, instead we remove the farthest VM from clients to decrease at most propagation delay. Therefore we allow the system to adapt to client movements and density. if $k = 0$, then $AWT = 0$, and no scaling down can be performed.

B. Service Placement Orchestrator

In order to calculate the optimal provisioning plan of a service graph request in such a multi-site cloud infrastructure, we use in this work a solution called Multi-site Orchestration System (MOSt) which relies on a placement algorithm called IGM (Iterative Graph Mapping) described in [14]. MOSt permits to find the best placement of the service request graph onto a networked multi-site cloud infrastructure. Each site is located in a specific geographical location and represented by a full IaaS cloud operated by a Site Manager (e.g., Openstack [15]) which has its own service portfolio and pricing. The system is responsible for the global provisioning only, meaning

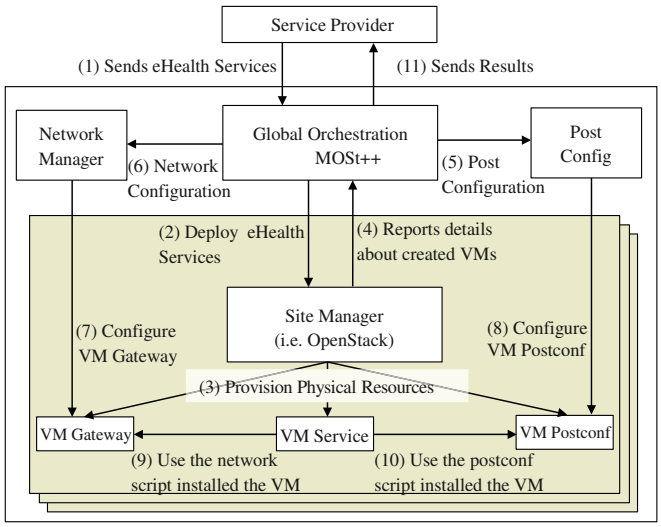

Figure 4. MOSt++ components and global architecture

that it will decide which service component will be initiated where (i.e. which datacenter site). Local provisioning depends on the local implementation. Therefore the core system of MOSt is responsible for building the provisioning plan and requesting the underlying sites to instantiate or replicate the resources based on the demand changes of the patients, on mobility patterns in order to fulfill the SLA terms.

The handling of a customer service request deployment is achieved in three phases:

1) Multi-site provisioning phase: MOSt calculates the optimal provisioning plan and engages the resources.
2) Post-configuration phase: MOSt launches the post-configuration of the deployed VMs based on the customer's request.
3) Networking provisioning phase: MOSt launches the configuration of the network connections between datacenters sites to fulfill nodes communication requirements of the service.

Figure 4 describes the complete diagram of a complex service deployment process in a multi-site datacenter.

Although MOSt calculates the best placement at the initial instantiation, it does not however guarantee the QoS delivered to patients during its all lifetime. In order for the CP to ensure that the SLAs terms are not violated, two solutions are possible:

1) scaling up/down the resources provided to the service based on the expected load or,
2) scaling up/down by duplicating the service components and efficiently load balancing between them as described in [16], [17] and [18].

However, both solutions introduce higher costs to the customer as well as weak resource utilization for the cloud provider.

In this context, we propose to leverage MOSt to MOSt++. A solution with new features such as dynamic auto-scaling management of complex services to solve both problems. The main features of MOSt++ are the capacity to monitor the eHealth service QoS and to scale up and down dynamically the computing resources allocated to these services in order to fulfill the patient's SLA terms.

IV. Implementation and Experimentation

The aim of the implementation is to highlight how MOSt++ will achieve the automatic scale up/down of services in a distributed cloud infrastructure.

A. OpenStack Infrastructure

For our experimentation, we have used OpenStack as a cloud management operating system for our data centers. The substrate nodes in the network graph are sites located in different geographical locations. Each site is a full IaaS cloud operated by a Site Manager (OpenStack in our implementation) which has its own service catalog and price. Node resources are Virtual Machines and Storage Volume that cloud customers (patients) may request, and their price depends on the node, service and the type of the resources (Virtual Machine/Volume). In our design, MOSt++ system plays the role of the global orchestrator that interfaces with the various underlying sites via OpenStack API.

B. Auto-scaling Evaluation

We have implemented the MOSt++ component and conducted a collection of evaluations, with and without the auto scaling mechanism. In order to evaluate the performance of auto scaling in this distributed cloud infrastructure, we have instantiated three sites, each site executes its own OpenStack platform. We have conducted an experiment to highlight the efficiency of the auto scaling mechanism and its impact on the users QoS. We deployed an eHealth service in one of the three sites. We have used Poisson distribution for the request model, λ parameter represents the requests arrival rate.

We considered that the medical healthcare provider wants the deadline to be respected for at least 99% of the requests (failure rate equal to 1%), however no more than two scaling up can be performed to keep the cost affordable. We send equal numbers of requests from two distinct locations (When $\lambda = 100$, it means we are sending 50 requests per second from location 1 and the rest from location 2). Each 5 minutes we change the arrival rate value and calculate the average failure rate during these 5 minutes, and indicate whether a scaling occurs. We can observe the evolution of the failure rate in Figure 5 for different λ. For low λ, the service is capable of treating all incoming requests. However, for $\lambda \geq 500$ requests per second, the failure rate starts to increase significantly without the auto scaling mechanism, while it's maintained when performing auto scaling. With MOSt++, an SLA violation is detected and a replica has been created to allow more requests to respect their given deadline and enhance the system's performance. However, for $\lambda = 1000$ requests per second, the cost limitation prevented more than two scales up which were insufficient to maintain the required QoS.

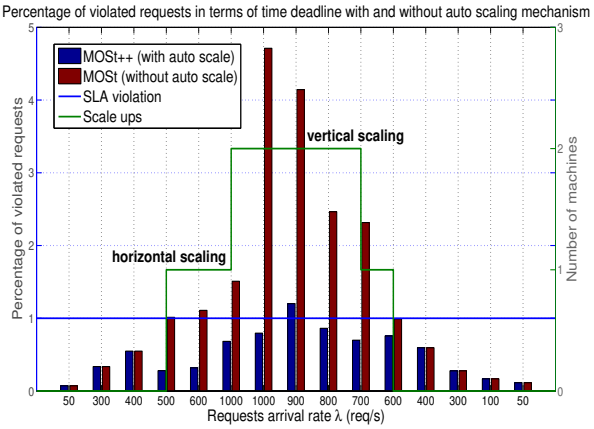

Figure 5. Percentage of violated requests in terms of time deadline with and without auto scaling mechanism

V. Conclusions and Future Works

In this work, we have presented a new contribution towards an efficient auto-scaling mechanism for an eHealth Service Provider in a Cloud infrastructure. With the development of cloud and eHealth technologies, eHealth service providers are requiring new mechanisms to efficiently instantiate their eHealth services in cloud infrastructure and ensure that allocated resources are compatible with the signed SLA terms and QoS terms with their customers. We have proposed in this paper a location aware auto-scaling mechanism (MOSt++) that leverages a previously developed service orchestration mechanism called MOSt. The new component called MOSt++ allows to calculate the best provisioning plan for complex eHealth services requested by an eHealth service provider. These services providers are also requesting efficient and automatic scale up and scale down mechanism. This auto-scaling mechanism relies on vertical and horizontal provisioning with application-delay constraints. MOSt++ has been implemented and its performance analyzed and highlighted its value. Further works are planned in this area to enhance the capabilities of MOSt++ to interpret any high level service requirement and translate it into infrastructure level mechanisms.

Acknowledgment

This research is partially funded by the AMSud SLA4Cloud project. Thanks to all the partners of the project who have helped with their discussions to improve the research work presented in this paper.

References

[1] J. E and all., "A wireless body area network of intelligent motion sensors for computer assisted physical rehabilitation." *J Neuroeng Rehabil.*, vol. 1, no. 6, 2005.

[2] C. Otto, A. Milenković, C. Sanders, and E. Jovanov, "System architecture of a wireless body area sensor network for ubiquitous health monitoring," *J. Mob. Multimed.*, vol. 1, no. 4, pp. 307–326, 2005.

[3] P. Mell and T. Grance, "The nist definition of cloud computing," *NIST Special Publication 800-145*, September 2011.

[4] E. C. Project. Extensible architecture and service infrastructure for cloud-aware software. [Online]. Available: http://www.easi-clouds.eu/

[5] M. Chen, J. Wan, S. Gonzalez, X. Liao, and V. C. M. Leung, "A survey of recent developments in home M2M networks," *IEEE Communications Surveys and Tutorials*, vol. 16, no. 1, pp. 98–114, 2014.

[6] O. Diallo, J. J. Rodrigues, M. Sene, and J. Niu, "Real-time query processing optimization for cloud-based wireless body area networks," *Information Sciences*, vol. 284, no. 0, pp. 84–94, Nov. 2014.

[7] F. Rosenberg, P. Leitner, A. Michlmayr, P. Celikovic, and S. Dustdar, "Towards composition as a service - a quality of service driven approach," in *Data Engineering, 2009. ICDE '09. IEEE 25th International Conference on*, March 2009, pp. 1733–1740.

[8] C.-w. Tsai, J. J. P. C. Rodrigues, and S. Member, "Metaheuristic Scheduling for Cloud : A Survey," *Systems Journal, IEEE*, vol. 8, no. 1, pp. 279–291, 2014.

[9] Y. Ahn, J. Choi, S. Jeong, and Y. Kim, "Auto-scaling method in hybrid cloud for scientific applications," in *The 16th Asia-Pacific Network Operations and Management Symposium*, 2014, pp. 1–4. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6996527

[10] N. Roy, A. Dubey, and A. Gokhale, "Efficient autoscaling in the cloud using predictive models for workload forecasting," in *Proceedings - 2011 IEEE 4th International Conference on Cloud Computing, CLOUD 2011*, 2011, pp. 500–507.

[11] J. Yang, C. Liu, Y. Shang, Z. Mao, and J. Chen, "Workload Predicting-Based Automatic Scaling in Service Clouds," *2013 IEEE Sixth International Conference on Cloud Computing*, pp. 810–815, 2013. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6740226

[12] Y. W. Ahn, A. M. K. Cheng, J. Baek, M. Jo, and H. H. Chen, "An auto-scaling mechanism for virtual resources to support mobile, pervasive, real-time healthcare applications in cloud computing," *IEEE Network*, vol. 27, no. 5, pp. 62–68, 2013.

[13] C. Janssen. (2014) Techopedia: What does pay as you go (payg) mean? [Online]. Available: http://www.techopedia.com/definition/26951/pay-as-you-go-payg

[14] K.-T. Tran, N. Agoulmine, and Y. Iraqi, "Cost-effective complex service mapping in cloud infrastructures," in *Network Operations and Management Symposium (NOMS), 2012 IEEE*, April 2012, pp. 1–8.

[15] "OpenStack Cloud Software," *Available at : http://www.openstack.org*.

[16] J. James and D. B. Verna, "Efficient load balancing algorithm in vm cloud environment," *International Journal on Computer Science and Engineering (IJCSE)*, vol. 4, pp. 1658, 1663, 2012.

[17] P. S. Meenakshi Sharma, "Performance Evaluation of Adaptive Virtual Machine Load Balancing Algorithm," *International Journal of Advanced Computer Science and Applications(IJACSA)*, vol. 3, no. 2, 2012.

[18] M. D. Shah and H. B. Prajapati, "Reallocation and allocation of virtual machines in cloud computing," *CoRR*, vol. abs/1304.3978, 2013.