Design and Analytical Model of a Platform-as-a-Service Cloud for Healthcare

Garrett Hayes¹, Hamzeh Khazaei², Khalil El-Khatib¹, Carolyn McGregor¹, J. Mikael Eklund¹
¹University of Ontario Institute of Technology, Oshawa, ON, Canada
²IBM Canada Research and Development Center, Markham, ON, Canada
Garrett.Hayes@uoit.ca, hamzehk@ca.ibm.com, {Khalil.El-Khatib, Carolyn.McGregor, Mikael.Eklund}@uoit.ca

Abstract

Recent progression in health informatics data analysis has been impeded due to lack of hospital resources and computation power. To remedy this, some researchers have proposed a cloud-based web service patient monitoring system capable of providing offsite collection, analysis, and dissemination of remote patient physiological data. Unfortunately, some of these cloud services are not effective without utilizing next-generation hardware management techniques. In order to make cloud based patient monitoring a reality, this paper shows how leveraging an underlying platform-as-a-service (PaaS) cloud model can provide integration with web service patient monitoring systems while providing high availability, scalability, and security. We also present an analytical model of the proposed platform and obtain performance measures such as delay in servicing as well as reject probability.

Keywords: Cloud computing, Health informatics, Platform as a service, Remote patient monitoring, Performance model.

1 Introduction

As technology continues to develop in our increasingly connected world, previous non-network enabled medical devices have begun to join hospital networks around the world. Thanks to recent medical device improvements, hospitals are now able to remotely collect patient data from bedside monitoring devices. The collection and analysis of this data has provided many opportunities for clinicians and researchers to discover new condition onset behaviours that are evident in a collection of physiological data streams before the current methods of detection for a range of conditions in critical care. Data collected from patients worldwide gives us the ability to extrapolate observations based on populous and historical data.

In order to support researchers using data collected from bedside monitoring devices, a number of researchers have proposed a cloud-based and offsite web service framework capable of collecting and analyzing patient data worldwide [2][7-12]. Unfortunately, underlying traditional technical infrastructure is incapable of providing a robust underlying platform that is highly available, expandable, and secure.

In this paper we will provide an overview of cloud computing, and present a specific cloud framework for use in critical care which demonstrates how leveraging a platform-as-a-service can provide a robust underlying architecture for such a deployment. We present an analytical model of the proposed platform and obtain performance measures such as delay in servicing as well as reject probability. We perform a numerical validation of that model using Maple 17.

2 Cloud Computing Overview

The concept of cloud computing generally refers to the delivery of one or more infrastructure components -- computational, storage, or otherwise -- as a scalable and reliable location-independent service. These services, often located offsite, are exposed to a consumer via preset client interfaces [13]. Such interfaces may include web browsers, standardized APIs, and other types of Internet-ready protocols. The location of a client’s provisioned cloud resources and data, unbeknownst to them, may span multiple servers, datacenters, or even countries.

Encompassing many service models, cloud computing allows the quick provisioning of resources regardless of underlying hardware or software components. Thanks to this client-centric model, enterprises are able to deploy or request additional infrastructure resources on demand, further streamlining their already complex IT operations [1][6].

The underlying amalgamation of various servers, storage area networks, and network hardware is conformed into a single standardized service interface referred to as a cloud. The resulting cloud provides the ability to elastically provision, span, and optimize resource and storage over multiple pieces or hardware while ensuring high availability and robust security.

Although the cloud concept seems esoteric by nature, the cloud model is being used to support many services on the web. Some of these services include popular products like Gmail, Facebook, Google Drive, Dropbox, Amazon AWS, Microsoft Office Web Apps, and many more.

While the majority of well-known cloud service deployments are Internet based (i.e., public clouds),...
in some instances enterprises may leverage the cloud architecture to consolidate resources and improve overall datacenter efficiency. In addition, when local resources are not cost efficient or sufficient enough for large projects, hybrid cloud models may be used to provision additional computation power on the fly. The four existing cloud models are as follows [13]:
(1) Private Cloud: operated by a private company.
(2) Community Cloud: operated by more than one company, often working together under a common agreement or area of interest and sharing a single cloud computing instance.
(3) Public Cloud: hosted by an external cloud provider whom offers cloud provisioning services to all types of customers.
(4) Hybrid Cloud: when two or more clouds are merged together, often through a standardized API interface, to provide cross-datacenter provisioning services on the fly.

Regardless of the cloud model used in a cloud deployment, various types of service models can be presented to administrators. Each service model aims to provide a specific set of resources pertinent to the requirements of the cloud consumer. These cloud service models are as follows: (1) Infrastructure-as-a-Service, (2) Platform-as-a-Service, (3) Software-as-a-Service, and (4) Data-as-a-Service.

3 Cloud Computing Advantages in Health Informatics

Cloud computing affords many advantages over traditional data center and hardware deployments. Benefits vary from increasing service availability to even reducing data center power consumption. The various benefits of cloud computing, particularly in health informatics, are as follows:
(1) Accessibility
(2) Availability
(3) Resource Consolidation and Multitenancy
(4) Reliability
(5) Scalability
(6) Security

4 Cloud Computing in Healthcare

The applications of cloud computing are not just limited to providing virtual machine (VM) instances for servers and other Internet facing services. Rather, the cloud model can be applied to any type of datacenter, regardless of its operational goals. One interesting application of cloud computing is in health informatics.

Current cutting-edge health informatics research projects aim to discover new condition onset behaviors that are evident in physiological data streams earlier than traditional detection of conditions in critical care data [7]. To do this, some hospitals may participate in pilot programs that aim to collect real-time patient data from network enabled monitoring devices. This collected data is then analyzed to extract relevant temporal behaviors and usually stored for future data mining and analysis operations. Naturally, not all hospitals may have the capacity -- in terms of networking, computational resources, and information technology support staff -- to fully support such pilot projects. In this section we will look at a case study implementing a neonatal pilot research project and show how the use of off-site platform-as-a-service cloud computing can help maximize project efficiency both inside the participating hospitals and out.

4.1 Current Cloud Applications in Health Informatics

Although cloud computing is still relatively new in the health informatics field, recently there have been a few cloud technology deployments in North American healthcare facilities. In addition, the advent of cloud-based services has created a fundamental shift in the way practitioners meet with patients and manage their electronic health record data. Below we will discuss two significant cloud deployments in the ever-growing health informatics field.

One of the most interesting recent cloud deployments is the Minnesota Tele-Health system. The Tele-Health healthcare network in Minnesota, USA leverages cloud-based medical consultation communications in their hospitals to provide secure remote video conferencing between patients and doctors [4]. This system allows patients to meet with doctors on a flexible, on-demand schedule regardless of physical location. This cloud-based Tele-Health system is capable of prioritizing calls, integrating with human translators when needed, and providing routine medical information by amalgamating existing hospital services and employees. In addition, Tele-Health conforms to the HIPAA standard thanks to the implementation of robust cryptographic controls [4].

Focusing more on health practice management, CareCloud provides a cloud-based framework for managing, analyzing, and charting patient data [3]. Their secure HIPAA compliant cloud service provides a scalable electronic healthcare record solution for healthcare practitioners, allowing them to focus on their patients instead of managing health data [3]. This system allows practitioners to add, edit, and share patient data using a web-based cloud service that provides secure, redundant, and offsite health record storage [3]. Furthermore,
CareCloud provides a community connection service that allows practitioners to meet virtually with their patients while sharing selective E-Health Record (EHR) data [3].

As we can see, both Tele-Health and CareCloud have provided innovative cloud-based services capable of streamlining existing healthcare practices while still conforming to HIPAA standards. However, those research projects do not support the transmission of high frequency physiological data streams in real-time for the provision of advanced clinical decision support remotely to support the Service of Critical Care.

4.2 Artemis Framework

The Artemis framework provides multidimensional real-time analysis of high-speed physiological data and also support advanced clinical research [7-12]. McGregor et al.’s platform showed the plausibility of supporting online real-time monitoring and analysis of clinical data to detect onset physiological conditions within the cloud-based environment [7][11-12]. Collected data may be used to provide extensive insight into novel discoveries regarding physiological data for earlier onset detection of a range of developing physiological conditions.

The Artemis framework, seen in Figure 1, consists of five main components which, when acting together, are capable of providing in-depth online analysis of developing physiological conditions in patients at participating hospital [2][7][11].

The first component facilitates client data acquisition. A medical data hub, located within each hospital, is responsible for collecting real-time clinical data from network-enabled medical devices onsite. In addition, medical data hubs pair real time patient data with existing e-health data from the hospital’s Clinical Information System [2][7][11].

Once all relevant data has been collected -- in real time- the data is pushed to the online analysis portion of the Artemis framework. Patient data is streamed in real time to the Artemis software-as-a-service interface for analysis. Collected data is then analyzed with IBM’s InfoSphere Streams (hereafter Streams) software to apply appropriate data analysis algorithms [7]. Leveraging the Streams software, “enables the real-time deployment of clinical rules representing correlations between behaviours of interest in physiological streams for each condition that is being monitored” [7].

Knowledge extraction of stored data is then done via Service-based Multidimensional Temporal Data Mining [9]. This can be performed on both real time and historical patient data. Extracted knowledge is then ready to be presented to researchers for further analysis. Finally, data is stored persistently for the long term to assist future knowledge extraction processes [7].

4.3 Artemis Cloud

In order to further the effectiveness of Artemis, McGregor et al. proposed a cloud-based software-as-a-service model that would allow hospitals to interact with the Artemis framework by consuming various web services [7][11]. Furthermore, hospitals would have persistent access to long-term data and knowledge extraction stored in the Artemis cloud, also accessible using a data-as-a-service storage service [7]. This can be seen in Figure 2.

Using various proposed web services, hospitals would be capable of initializing a Streams environment with custom rule sets and knowledge extraction algorithms [7]. Patient data can then be streamed into the appropriate web interface, encoded in the proposed XML protocol format [12], and coupled with existing patient data in the HL7 V3 format. Ultimately this setup would allow clinicians to deploy their own knowledge extraction rules to effectively monitor patient changes in real time and perform extensive clinical research.

Although this framework seems to fit the needs of both
researchers and clinicians, McGregor states the “plausibility of using cloud environments for robust healthcare monitoring systems is unknown” [7].

5 Providing PaaS for Artemis Cloud Deployments

As we have seen in the above Artemis cloud framework, each participating hospital must be able to provision a private Streams instance capable of conducting knowledge extraction through customized rule sets. The real challenge of successfully deploying the Artemis cloud lies in the provisioning of underlying resources needed by each hospital. If each component of the cloud architecture in Figure 2 were run entirely on individual servers, eventually computational resources would be exceeded. In addition, the provisioning and addition of extra hardware resources would be difficult and could result in unneeded downtime. In this section we will define the requirements for an Artemis cloud deployment and show how the use of an underlying platform-as-a-service model can be effectively leveraged by McGregor et al.’s proposed web service framework.

5.1 Requirements

The underlying PaaS deployment leveraged by the Artemis cloud framework must meet various requirements to be effective. These range from high availability to stringent security policies. The most fundamental set of requirements is as follows:

(1) Hospitals must be able to securely interact with the Artemis web endpoint without worrying about data leakage or plaintext data transfers. This can be done using various virtual private network (VPN) technologies. VPN security settings should be chosen based on regional data privacy requirements.

(2) Each hospital must have access to its own instance of Streams VM via their Artemis web service interface.

(3) Each hospital’s patient data must be isolated as per regional regulatory standards and must be stored long-term for later knowledge extraction. Long-term data storage should have high redundancy and zero loss, even during hardware failure.

(4) The Artemis cloud must suffer no downtime. 99.999% availability or better should be provided at all times, regardless of hardware failure.

(5) Underlying physical resources can be added or removed at will with zero downtime. This ensures the expandability of computation and data storage resources in the future.

(6) The exploitation of a single hospital’s web service should under no circumstances provide access to, or affect, another hospital’s patient data.

(7) The Artemis web service should be capable of interfacing with underlying platform-as-a-service components using a standardized API. This facilitates the deployment of Streams and customized rule sets via the Artemis web services interface.

5.2 High Level PaaS Architecture

In order to meet the requirements outlined above, we first analyzed the deployment requirements for each hospital leveraging the Artemis cloud. After breaking down underlying hardware and software components, we concluded that each hospital’s web service deployment constituted only four underlying components:
Design and Analytical Model of a Platform-as-a-Service Cloud for Healthcare

5.3 Underlying PaaS Architecture

In order to provide the ability to deploy the logical PaaS model outlined in the previous section, we were tasked with selecting the underlying software and hardware capable of meeting the requirements outlined in section 5.1 of this paper. Of upmost importance was ensuring the selected underlying software was capable of utilizing a plethora of hardware in multi-vendor environments while providing a standardized API interface capable of interacting securely with virtualized resources.

After much research, we decided to leverage VMware’s vSphere hypervisor to meet our underlying software requirements. vSphere, a low level hypervisor software, allows us to pool any number of hardware resources into a logical cluster capable of providing high availability and disaster recovery [5]. This virtualized datacenter is easily managed and monitored via VMware’s vCenter product and is able to provision Artemis cloud resources while providing a robust and mature API interface for interacting with said components [5].

The vSphere product, when coupled with vCenter can provide [5]:

1. High availability in the datacenter, capable of avoiding downtime even during hardware failure.
2. The ability to add or remove hardware resources at any time with zero downtime.
3. Deployment of virtualized Artemis assets extremely fast based on pre-defined service templates.
4. Access to Artemis resources using robust and well tested API interfaces.
5. Automatic provisioning of hardware resources to ensure a single hospital does not utilize all computation resources.
6. Host isolation capable of reducing the attack surface of the Artemis cloud and ensuring hospitals are 100% isolated from all other internal components.
7. Pooling of underlying hardware and storage resources to prevent unexpected data loss, even during hardware failure.
8. Snapshotting of instances in time to facilitate system roll-back, recovery, or knowledge extraction.
9. A simple and secure administrative web interface capable of safely manipulating and monitoring underlying hardware resources in real time.

As we can see, the vSphere and vCenter product exceed our requirements for an underlying PaaS platform that can be leveraged by the Artemis cloud framework.

6 Performance Model of the Platform

In this section we present an analytical model of the
proposed platform and obtain the important performance measures such as delay in servicing as well as reject probability on incoming hospital requests for Streams VMs and potentially DB2 and Data Miner VMs. We assume a large number of hospitals so that we can consider the arrivals as a Poisson process [15]. We also assume the service time for each patient, i.e., the life time of Streams VMs, is exponentially distributed. A hospital submit a request to the platform whenever it requires a new instance of Streams. The hospital could be a new hospital or the one that already has a number of running Streams VMs on the platform. In this study we assume that each Streams VM can accommodate 5 to 10 patients. For new hospitals, the DB2 and Data Minter VMs instantiation will be done in parallel with the instantiation of Streams VM. In the numerical validation part we determine the mean value of service time and mean arrival rate of hospital requests based on the data from a study on Intensive Care in Canada outlined in [21]. According to [21], the length of stay in special care units was 4.7 days for adults and 8.1 days for neonatal across Canada in 2003 ~ 2004.

6.1 Analytical Model

In the proposed platform, when a request is processed, a prebuilt or customized disk image is used to create one or more VM instances. In this paper, we assume that prebuilt images, i.e., default Streams images, fulfill all user requests and only one VM instances is granted to the each hospital per each request. We assume that physical machines, PMs, are categorized into three server pools: hot (i.e., with running Streams VMs) and warm (i.e., turned on but without running VM) [14]. Such categorization reduces operational costs and offers the capability for disaster recovery [18-19]. Each PM has up to two virtual machine monitor (VMM) units that configure and deploy VMs on a PM. Instantiation of a VM from an image and provisioning it on hot PMs has the minimum delay compared to warm PMs. Warm PMs require more time to be ready for provisioning. Requests are queued in a global finite queue and then processed on the first-in first-out basis (FIFO).

The instance allocation sub-model (IASM) processes the request at the head of global queue as follows: First, it tries to provision the request on a PM machine in the hot pool. If the process is not successful then IASM tries the warm pool. Ultimately, IASM either assigns a PM in one of the pools to a request or the request gets rejected. As a result, hospital requests may get rejected either due to lack of space in global input queue or insufficient resource at the cloud center. When a running task finishes, the capacity used by that VM is released and becomes available for servicing the next request.

6.2 Instance Allocation Sub-Model (IASM)

The PM allocation process is described in the Figure 4. IASM is a 2D CTMC in which we take care of number of requests in the queue as well as the current pool on which provisioning is taking place (Figure 5). Since the number of potential hospitals is high, and each hospital submits requests one at a time with low probability, the requests arrival can be modeled as a Poisson process with rate $\lambda$. Requests are lined up in the global finite queue to be processed in a FIFO basis. Each state of Markov chain is labeled as $(i,j)$, where $i$ indicates the number of requests in queue and $j$ denotes the pool on which the leading request is under provisioning. State $(0,0)$ indicates that system is empty that means there is no request under provisioning or in the queue. Index $j$ can be $h$ or $w$ that indicates the current request is undergoing provisioning on the hot or warm pool, respectively. The global queue size is $L_q$ and one more request can be at the deployment unit (i.e., VMM) for provisioning so that the total system capacity is $L_q + 1$.

Let $P_h$ and $P_w$ be the success probabilities of finding a PM that can accept the current request in hot and warm pool, respectively. We assume that $1/\alpha_h$ and $1/\alpha_w$ are the

![Figure 4 The Steps of Hospital Requests’ Servicing in the Platform](image-url)
mean look-up delays for finding an appropriate PM in hot and warm pool, respectively. Upon arrival of first request, system moves to state (0,h) which means the request will be provisioned immediately in the hot pool. Afterward, depending on the upcoming event, three possible transitions can occur:

1. Another request has arrived and system transits to state (1,h) with rate $\lambda$.
2. A PM in hot pool accepts the request so that system moves back to state (0,0) with rate $P_h \alpha_h$.
3. None of PMs in hot pool has enough capacity to accept the request, so the system will examine the warm pool (i.e., transit to state (0,w)) with rate $(1 - P_h) \alpha_h$.

On state (0,w), IASM tries to provision the request on warm pool; if one of the PMs in warm pool can accommodate the request, the system will get back to (0,0). If none of the PMs in warm pool can provision the request as well, the system moves back from (0,w) to (0,0) with rate $(1 - P_w) \alpha_w$ which means the hospital request will get rejected due to insufficient resources in the platform. Finally, the request under provisioning decision leaves the deployment unit, once it receives a decision from IASM and the next request at the head of global queue will go under provisioning. In this sub-model, arrival rate of requests and look-up delays ($1/\alpha_h$ and $1/\alpha_w$) are exogenous parameters and success probabilities ($P_h$ and $P_w$) are calculated from the VM provisioning sub-model. The VM provisioning sub-model will be discussed in the next section.

Two types of blocking may happen to a given request:

1. Blocking due to a full global queue occurs with the probability of
   
   $$BP_q = \pi_{(1,1)} + \pi_{(1,0)}$$

2. Blocking due to insufficient resources (PMs) at pools [14], with the probability of
   
   $$BP_p = \sum_{i=0}^{k} \frac{\alpha_i (1 - P_i)}{\alpha_w + \lambda} \pi_{i,w}$$

The probability of reject then, $P_{rep}$ is then $P_{rep} = BP_q + BP_p$. To calculate the mean waiting time in queue, we first establish the probability generating function for the number of requests in the queue [15],

$$Q(z) = \pi_{(0,0)} + \sum_{i=0}^{\infty} \pi_{(i,h)} + \pi_{(i,w)})z^i$$

The mean number of requests in queue ($\bar{q}$) is the first derivative of $Q(z)$ at $z = 1$ [14],

$$\bar{q} = Q'(1)$$

Applying Little’s law [16], the mean waiting time in queue ($\bar{w}$) is given by

$$\bar{w} = \frac{\lambda}{1 - P_{ni}}$$

Look-up time among pools can be considered as a Coxian distribution with two steps. So it can be calculated as [15],

$$\bar{lut} = \frac{1/\alpha_h + (1 - P_h)(1/\alpha_w)}{1 - BP_q}$$

### 6.3 VM Provisioning Sub-Model (VMPSM)

Virtual machine provisioning sub-model captures the instantiation, deployment, and provisioning of VMs on a PM. VMPSM also incorporate the actual servicing of each request (VM) on a PM. Figure 6 shows the VMPSM for a PM in hot pool. A PM in warm pool can be modeled with the same VMPSM, though, with different arrival and instantiation rate. Consequently, each pool (hot and warm) can be modeled as a set of VMPSM with the same request arrivals and instantiation rate. Each state in Figure 6 is labeled by $(i,j,k)$ in which $i$ indicates the number of requests in PM’s queue, $j$ denotes the number of task that is under provisioning, and $k$ is the number of VM that are already deployed on the PM. Let $\phi_i$ be the rate at which a VM can be deployed on a PM at hot pool and $\mu$ be the service rate of each VM. So, the total service rate for each PM is the product of number of running VMs by $\mu$. State (0,0,0) indicates that the PM is empty and there is no task either in the queue or in the instantiation unit.

The arrival rate to each PM in the hot pool is given by

$$\lambda_h = \frac{\lambda(1 - BP_q)}{N_h}$$

in which $N_h$ is the number of PMs in the hot pool. The state transition in VMPSM can occur due to requests arrivals, task instantiation, or service completion. From state (0,0,0), system can move to state (0,1,0) with rate $\lambda$ lambda. From
(0,1,0), system can transit to (0,0,1) with rate \( \varphi \) (i.e., instantiation rate). From state (0,0,1) system can get back to (0,0,0) with rate \( \mu \). Using steady-state probabilities, we can obtain the probability that at least one PM in hot pool can accept the request for provisioning. First, we need to compute the probability that a hot PM cannot admit a request for provisioning (\( P_h^{na} \)).

Therefore, probability of success provisioning (\( P_h \)) in the hot pool can be obtained as

\[
P_h = 1 - (P_h^{na})^\gamma
\]

Note that \( P_h \) is used as an input parameter in the IASM (see Figure 4). The provisioning model for warm PM is the same with the one for hot PM, though, there are some differences in parameters:

1. The arrival rate to each PM in warm pool is

\[
\lambda_w = \frac{\lambda(1-B_P)(1-P_h)}{N_w}
\]

where \( N_w \) is the number of PMs in the warm pool.

2. Every PM in warm pool requires extra time to be ready (i.e., hot) for first instantiation. This time is assumed to be exponentially distributed with mean value of \( \gamma_w \). Instantiation rate in warm pool is the same as in hot pool (\( \varphi_h \)). Like VMPSM for a hot PM (see Figure 6), the model for a warm PM is solved and the steady-states probabilities are obtained. The success probability for provisioning a request in warm pool is

\[
P_w = (1 - P_w^{na})^\gamma_w
\]

From VMPSM, we can also obtain the mean waiting time at PM queue (\( \bar{w}_{PM} \)) and mean provisioning time (\( \bar{p} \)) by using the same approach as the one that led to the computation of \( \bar{w} \). As a result, the total delay before starting of actual service time, is given by

\[
\bar{d} = \bar{w} + \text{lat} + \bar{w}_{PM} + \bar{p}
\]

There is an interdependence among IASM and VMPSM. This cyclic dependence is resolved via fixed-point iterative method [20] using a modified version of successive substitution approach [14].

7 Numerical Validation

The resulting sub-models have been implemented and solved using Maple 17 from Maplesoft, Inc. [17]. The successive substitution method is continued until the difference between the previous and current value of blocking probability in the global queue (i.e., \( BP_q \)) is less than \( 10^{-6} \). Usually the integrated model converges to the solution in less than 10 iterations. Under different configurations and parameter settings, we obtain two important performance metrics, namely, rejection probability of requests and total response delay. The Streams VM will be shutdown when all patients who were assigned to that Streams instance leave the hospital. Since we assumed 5 to 10 patients per instance, the mean life time of Streams instance can be estimated from 30 and 90 days.

We assume a large number of hospitals, i.e., 200 hospitals or more, so that the arrival rate could range from 250 to 600 requests per day and global queue size is set to 50 requests. We assume 6 to 16 PMs in each pool and each PM can run up to 6 VMs simultaneously. The look-up time for finding a proper PM is assumed to be independent of the number PMs in the pool and the type of pool (i.e., hot, and warm). Look-up rate is set to 12 searches per minute. Mean preparing delay for a warm PM to become a hot
PM (i.e., be ready for first VM instantiation) are assumed to be 1 to 3 minutes. Mean time to deploy a VM on a hot PM is assumed to be between 1 and 10 minutes. First VM instantiation on a warm PM are to be 5 to 20 minutes. After first VM instantiation on a warm PM, it has already become a hot PM so that the next VM can be deployed just like a hot PM.

We show the effects of changing arrival rate, request service time and number of PMs in each pool on the interested performance indicators. In the first experiment, the results of which is shown in Figure 7, it can be seen that by increasing the mean service time the rejection probability increases almost linearly. Also, we noticed that curves (for pool capacity 6 ~ 16 PMs/pool) in Figure 7 are almost equidistant. Figure 8 shows that a longer service time will result in a longer total delay on requests. In addition, it can be observed that by increasing the capacity of system (i.e., having more PMs in the pools) the maximum gains occurs at the longest service time (i.e., 180 days).

Under different arrival rates, we also determine the two performance metrics while the average service time is set to 140 days (Figures 9 and 10).

One turning point can be noticed in rejection probability curves (Figure 9) at which rejection probability increases suddenly. Such points (e.g., Figure 9, 400 requests/day for 8 PMs in each pool) may suggest the appropriate time to upgrade or inject more resources (i.e., PMs). However, in case of total delay, two turning points (three regimes of operation) can be identified (Figure 10): the first regime (i.e., stable regime) is the region in which the platform provider would like to operate. An admission control may use provided information here and helps the platform to operate in such regime by not accepting extra requests. Transient regime is in between two turning points which the total delay increases sharply. Platform owner should avoid entering such region since the violation of Service Level Agreement (SLA) is about to happen. However, since the transient regime is not too narrow, it is possible for cloud center to return to stable zone after a short time (i.e., the admission control has enough time to adjust the new admissions). In saturation regime, the requests are experiencing the highest possible amount of delay and this zone is not in the interest of both platform providers and hospitals.
8 Conclusions and Future Works

This paper has presented a specific cloud framework for use in critical care. We present an analytical model of the proposed platform and obtain performance measures such as delay in servicing as well as reject probability. We perform a numerical validation of that model using Maple 17.

Thanks to recent advances in cloud technologies, like VMware’s vSphere and vCenter products, it is possible to provide a robust and security underlying PaaS framework that can be leveraged by the Artemis cloud proposed by McGregor. Such a PaaS architecture can ensure high availability and seemingly infinite resource expansion capable of facilitating real time analysis of remote patient data. Furthermore such a robust architecture, when coupled with McGregor’s Artemis cloud design, can be used to provide effective and secure web services that can be leveraged by hospitals worldwide.

Currently the Artemis cloud architecture, and the proposed underlying PaaS model, is being deployed at the University of Ontario Institute of Technology. Future work will focus on tightly integrating the proposed PaaS framework with McGregor’s Artemis web service interface to ensure the effective provisioning and management of virtualized hospital resources.

By coupling Artemis with our proposed PaaS model, we can showcase the plausibility and effectiveness of leveraging cloud-based resources to provide real-time knowledge extraction from network-based patient monitoring devices.

References


Design and Analytical Model of a Platform-as-a-Service Cloud for Healthcare


Biographies

Garrett Hayes is a lecturer in the Faculty of Business and Information Technology at the University of Ontario Institute of Technology, Canada. He received his MSc in Computer Science from the University of Ontario Institute of Technology. His focus areas include computer and network security, industrial control system security, and cloud technologies.

Hamzeh Khazaei is a Research Scientist with the IBM Canada Research and Development Centre. He received his PhD in Computer Science from University of Manitoba, Canada and MSc & BSc degrees from Amirkabir University of Technology, Tehran, Iran. His research interests include cloud computing architectures and cloud computing modelling.

Khalil El-Khatib is an Associate Professor in the Faculty of Business and Information Technology at the University of Ontario Institute of Technology, Canada. He received his PhD from the University of Ottawa. His research interests include: security and privacy for smart grid, smart environments.

J. Mikael Eklund is an Associate Professor in the Department of Electrical, Computer and Software Engineering at the University of Ontario Institute of Technology, Canada. He received his PhD in Electrical and Computer Engineering from Queen’s University, Kingston, Canada. His present research is on real-time analytics for healthcare applications.

Carolyn McGregor is the Canada Research Chair in Health Informatics and Professor in the Faculty of Business and Information Technology at the University of Ontario Institute of Technology, Canada. She has led pioneering research in Big Data, analytics, event stream processing, data stream data mining, process modeling and cloud computing.

Biographies