Automatic Resource Scaling for Medical Cyber-Physical Systems Running in Private Cloud Computing Architecture[*][†]

Yong woon Ahn and Albert Mo Kim Cheng

Department of Computer Science, University of Houston
4800 Calhoun Road, Houston, Texas, U.S.A.
{yahn,cheng}@cs.uh.edu

Abstract

Cloud computing and its related virtualization technologies have become one of dominant trends to deploy software, compute difficult problems, store different types of data, and stream real-time video and audio. Due to its benefits from cost-efficiency and scalability to maintain server solutions, many organizations are migrating their server applications running on physical servers to virtual servers in cloud computing infrastructures. Moreover, cloud computing has enabled mobile and battery-powered devices to operate without strong processing power and large storage capacity. However, it is not trivial to use this trendy technology for medical Cyber Physical Systems (CPSs) which require processing tasks' requests to send instructions to the local actuator within specified deadlines. Since a medical CPS device monitoring a patient's vital signs may not have a second chance to recover from an erroneous state, achieving cost-efficiency with higher resource utilization in cloud computing may not be the ultimate goal to configure the healthcare IT infrastructure with medical CPS devices. In this paper, we focus on private cloud infrastructures with the fair resource sharing mechanism in order to run medical CPS applications. First, we introduce our medical CPS device model used for designing our cloud infrastructure following the Integrated Clinical Environment (ICE) standard developed by the Medical Device Plug-and-Play (MDPnP) project. Second, we investigate limitations to deploy CPS applications using existing auto-scaling mechanisms. Finally, we propose our novel middleware with a virtual resource sharing mechanism inspired by autonomic computing, and present its performance evaluation results simulated in the OpenStack private cloud.

1998 ACM Subject Classification C.2.4 Distributed Systems, C.3 Special-Purpose and Application-Based Systems, J.3 Life and Medical Sciences

Keywords and phrases Auto-Scaling, Cloud Computing, Medical Cyber-Physical System Device, Virtualization, Autonomic-Computing

Digital Object Identifier 10.4230/OASIcs.MCPS.2014.58

1 Introduction

Cloud computing has become one of common technologies to provide unlimited computing experiences with small and battery powered mobile devices. Moreover, it provides other great advantages for deploying and maintaining server-side applications because of its flexibility to scale up and down computing and storage resources elastically. This flexibility is implemented by various hardware virtualization techniques which enable virtual machines (VMs) to be

* Supported in part by the National Science Foundation under Awards No. 0720856 and No. 1219082.
† This research is a part of the yPlatform project for medicine by Amerra Inc.

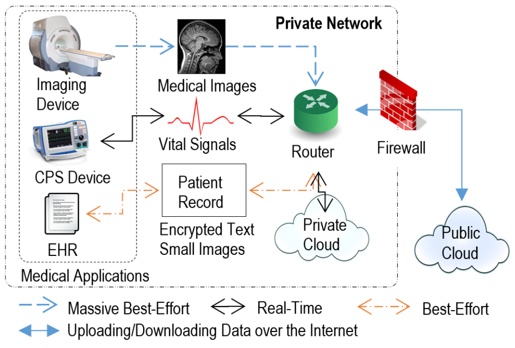

Figure 1 A hybrid cloud for medical devices and applications.

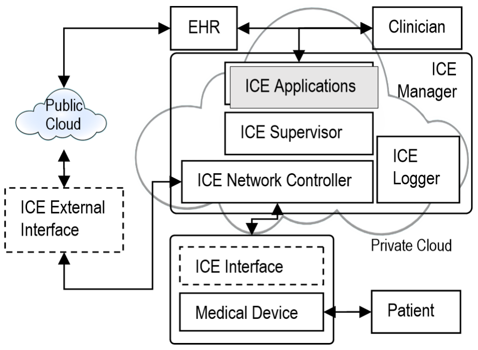

Figure 2 A hybrid cloud for medical CPS devices.

easily launched or terminated on demand to maintain the desirable Quality of Service (QoS) level for different types of common application. Like other IT areas, cloud computing is getting more attention from healthcare IT industries not only to reduce costs but also to improve patient care. By taking advantages from cloud computing, the healthcare cloud helps clinical environments remove their in-facility server rooms entirely by subscribing a public cloud Infrastructure as a Service (IaaS) or partially by deploying a private or hybrid IaaS [14]. Since medical records are generally very sensitive, and must be protected by highly secure physical facilities, multi-layered software or hardware network security systems, data encryption, and redundancy, operation of the in-facility server rooms can be more expensive and less secure to process and store medical data. More seriously, some medical imaging devices generate extremely massive data requiring huge data storages. To satisfy these requirements, a public IaaS could be a common solution because of its highly secure physical and virtual resources which also can be scaled up and down easily.

For non-real-time medical devices, public cloud computing solutions could be more suitable to provide healthcare IT systems by taking advantage of existing cloud computing solutions. However, these public and remote cloud solutions are too unpredictable to maintain the desirable QoS of medical CPSs because VM monitors used by their services basically do not know what type of medical application server is being loaded in a VM. Also, many uncertainties from transferring data over various networks would be serious issues in using public cloud solutions for medical CPSs. Although cloud computing service vendors provide Service Level Agreements (SLAs) covering cases where the user cannot access their subscribed computing and storage resources, they cannot guarantee that an application server responds to a source application within a specific task deadline. To overcome this limitation, a hybrid cloud architecture can be a solution. For non-real-time medical devices, we use VMs in a public cloud, and for medical CPS device, we use VMs in a private cloud locally located as shown in Figure 1. In this paper, we restrict a private cloud to operate as the in-facility server solution built with a stable network environment. No private cloud offered by public IaaS providers is considered as a private cloud in this paper. Each medical CPS device can transmit real-time tasks to this private cloud which can be specially configured to process real-time tasks with the highest priority. In this paper, we assume that each medical device can be interconnected via the ICE standard [4] which is one of the MDPnP projects [10] to support a cross-manufacturer medical device interoperability. Figure 2 shows data flows of CPS medical devices connected via ICE interfaces to deliver their real-time and life-critical tasks to ICE application servers running as a part of the ICE manager. The ICE supervisor is responsible for generating alarms to indicate that the required tasks cannot be processed with the current configurations of the ICE application servers. All ICE manager software and hardware components can be run in a private cloud, if these components can be emulated by virtualization technologies. If the medical device is a closed-loop CPS system, one or more ICE application servers are run as computing resources to process requests, and to respond action instructions to the source device. These ICE application servers also can provide user interfaces help clinicians check patient's states, and upload medical records to EHR systems as shown in Figure 2. Despite using a hybrid cloud solution, there are still possible issues to maintain the desirable QoS of medical CPS devices because most open sourced and commercialized private cloud solutions such as OpenStack [13], Eucalypus [12], and VMWare vCloud [16] have very similar auto-scaling mechanisms to adjust virtualized computing resources dynamically. Although these existing auto-scaling mechanisms are primary technologies to achieve the main goal to operate VMs on demand, these mechanisms are commonly performed by checking system performance metrics which human system administrators select. These system metrics can include the monitoring values of Virtual CPU (VCPU), memory, storage I/O, and network bandwidth. Although these metrics can represent a health status of each VM, they cannot represent whether all tasks are processed within their deadlines specified by medical CPS devices [9].

There are three major requirements to design and implement a private cloud for medical CPS devices using the ICE standard.

(a) A group of ICE application servers should always be ready to process all incoming real-time tasks from multiple CPS devices, and respond action instructions to the source CPS devices within specified deadlines.

(b) The private cloud must be designed to achieve a goal of higher physical and virtual resource utilization except any emergency case.

(c) Computing resources must be automatically adjusted by an implemented mechanism when a new CPS device is discovered, or one of connected devices increases bigger tasks.

These three requirements are essential, if we assume that the clinical environment adopts an ICE standard. Since computing powers of ICE application servers are generally configured before initializing the entire system, it requires that all ICE application servers always run to satisfy the worst-case scenarios even for not discovered medical CPS devices. In other word, there would be no advantage of the higher resource utilization by using cloud technologies. Also, even if the clinical environment adopts a private cloud, supporting the MDPnP standard would be another consideration. In this paper, we design middleware running in a private cloud infrastructure to provide a novel auto-scaling mechanism to preserve the cost-efficiency. Our middleware is performed independently in each VM with an ICE application server without modifying the ICE standard. Moreover, in order to implement an automatic service which can be self-optimized, we adopt the autonomic computing concepts to design our middleware [5].

The remainder of this paper is organized as follows. In Section 2, we introduce other researches working towards similar goals. Design and implementation of our solution are presented in Section 3 and 4. In Section 5, we evaluate our proposed middleware running in an OpenStack private cloud.

2 Related Work

For common real-time applications, S. Liu et al. proposed an on-line scheduling algorithm of real-time services for cloud computing in [7]. Their algorithm modifies the traditional utility accrual approach [3, 8] to have two different time utility functions (TUFs) of profits and penalties on executing tasks. One important assumption the authors made is that the timeliness with relative task deadlines would be a more realistic principle for most real-time applications than the absolute deadline guarantee for hard-real-time systems due to the nature of diverse network communication methods causing many uncertainties. Although their assumption about the timeliness is reasonable, this research does not consider cases of medical CPS devices which might be discovered at any time.

As we stated in the previous section, the most feasible solution to operate medical CPS devices in the cloud can be the auto-scaling mechanism to scale up the number of VMs to process real-time tasks without missing their deadlines, and to scale down to provide the cost-efficient and energy-saving physical data center alternative. In [9], M. Mao et al. proposed an auto-scaling mechanism considering task deadlines and budget constraints. Although their approach is based on deadline constraints to overcome downsides of the threshold-based stock auto-scaling mechanisms monitoring system metrics, a system administrator still has to adjust the configuration file manually whenever a new application needs to be connected. Also the authors did not consider cases with real-time medical applications which are possibly required to compress and transmit massive data to remote locations. To the best of our knowledge, our approach is the first attempt to adopt the autonomic computing concepts to operate the MDPnP environment.

3 System Design

In order to deploy medical CPS devices connected to a private cloud infrastructure, we first show common procedures to process CPS tasks. We assume that all CPS medical devices follow ICE standards to send and receive messages to and from an ICE supervisor, and we also assume that healthcare environments use a private cloud to configure the ICE manager.

3.1 Medical CPS Device

Figure 3 shows procedures to sample data from patient's body. In this paper, we focus on medical CPS devices which sample patient health condition data via various sensors, and compress them before transmitting to an ICE application server periodically. To interact between two different types of medical device, we use the device profile protocol introduced in [6]. This profile protocol is based on the ISO/IEEE 11073 Domain Information Model. In order to discover a new medical device, each device must send its profile protocol message to an ICE supervisor. This message includes device type, device health status, manufacture information, clock, device model, medical nomenclature, sampling period, event-trigger function, network interface, network protocol version, and so on. As Figure 3 shows, compressed data, Δ, would have a deadline, δ, which must be processed by the server before arriving the next period job to operate actuators. We call this data to the destined ICE application server real-time tasks. Also, if the medical CPS device did not receive this action command before sending the next period of sampled data, its device health status would become the *"fail"* state.

3.2 Proposed Middleware between an ICE Supervisor and ICE applications

In order to provide standard-compatible solutions for other existing ICE compliant systems, we do not change an ICE supervisor and any of its internal procedures. Our system must be

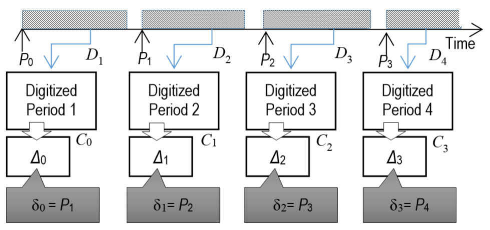

Figure 3 An example of real-time tasks collected by one sensor: P_i is the value of time starting i_{th} period, Δ_i is the number of time slots of the i_{th} compressed period, and D_i is the delay to digitize sampled or input original data for ith period.

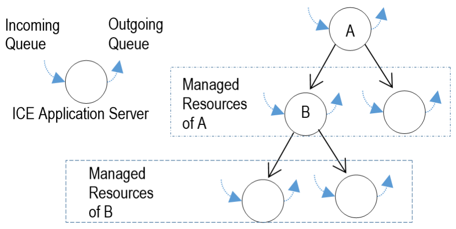

Figure 4 A parent VM with an ICE application server which has two VMs installing child ICE application servers which are dealt as managed resources in the autonomic computing concept.

run independently from the existing ICE manager. The ICE supervisor is responsible only to check whether the application servers and network interfaces can handle requests from newly discovered medical devices. If it detects any resource shortage or inappropriate profiles from the device, the supervisor indicates an alarm [4]. Our goal is to protect a medical device from being rejected due to shortage of computing resources when using a private cloud. To design our middleware to manage VMs running ICE application servers, we use autonomic computing concepts to implement the self-management with four essential attributes such as self-configuration, self-healing, self-optimization, and self-protection [5]. After initializing the ICE manager, our middleware must accumulate knowledge from the previous history of processing data and responding action instructions to decide whether the ICE supervisor needs more resources or not.

In our middleware design, each VM is a node in a tree data structure shown in Figure 4. Each node also can become an autonomic manager as shown in Figure 5. Our autonomic manager has managed resources which can be other ICE application servers running in different VMs. Our middleware is running in each VM and checking the health state of its ICE application server usages by counting how many tasks missed their deadlines for the predefined duration. There are four steps to accumulate knowledge and adjust managed resources in our autonomic manager. In the monitoring step, our autonomic manager collects managed resource states from its child VMs by checking the number of missed deadlines. In the analyzing step, the autonomic manager calculates the number of VMs for the next iteration, and sends a request to the next step to make a new plan such as task assignments for ICE application servers in child VMs, if it would improve system's overall utilization. In the planning step, the new plan is setup to launch or terminate one or more VMs, if it is requested by the previous step. Finally, in the executing step, the autonomic manager would execute this plan by calling private cloud management functions such as jClouds [2]. From these four steps, the managed resource can be adjusted by our autonomic manager running as middleware without human interventions. Also, since each middleware runs in different VM

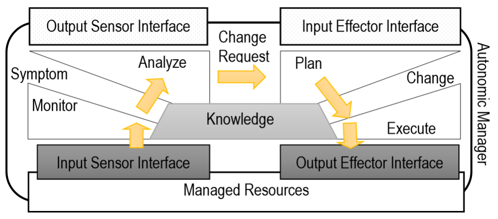

Figure 5 A generic architecture for autonomic manager.

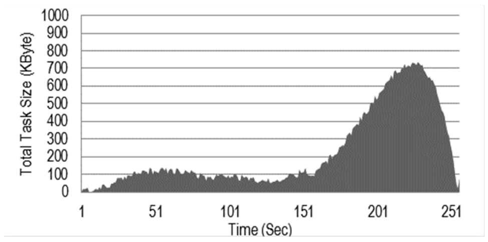

Figure 6 Real-time tasks used for the simulation.

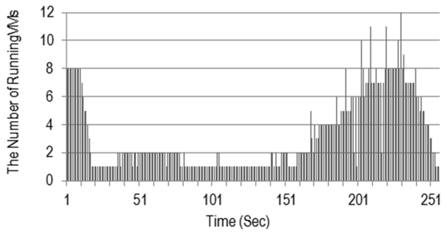

Figure 7 The number of VMs processing data.

independently, we can avoid any possible bottleneck when having one centralized coordinator to manage all VMs. To avoid any issue from having less knowledge at the beginning, the managed resources must be started with a sufficient number of VMs to process the worst case scenario, and gradually this number of VMs would decrease if no new medical device is discovered, or existing medical CPS devices stably transmit their real-time tasks periodically.

4 Implementation

We are implementing our system design on the OpenStack cloud which is one of the most well-known open source cloud infrastructure. Currently, this OpenStack environment only is used for medical CPS devices, and other medical applications use Amazon EC2 [1] public cloud. To implement our middleware approach, we wrote a medical CPS device simulator in Java. This simulator uses sampled ECG data from the MIT-BIH database [11] as its input, and sends compressed data to the ICE application servers periodically, and must receive action commands before sending other data to control its virtual actuator. In order to launch and terminate VMs, we use jClouds to control VMs.

5 Performance Evaluation

To evaluate our approach, we use the OpenStack Grizzly version running with Intel Xeon E3 Quad-Core CPU, 16 GB RAM, 500GB SAS HDD, two network interface cards, and Ubuntu server operating system. We use the Ubuntu 12.04 VM image [15] to run ICE application server simulators, and each VM is launched with 512Mbyte RAM, one VCPU, and no local storage. We assume that the ICE supervisor knows all current states of ICE application servers, and works well to coordinate tasks for every connected medical CPS device. We setup each ICE application server processes 100Kbyte ECG data per one second in the incoming task queue. Figure 6 shows our total workload receiving from medical CPS device simulators. At the beginning, we only have one medical device, but it increases its data size slightly between 40 and 80 seconds. This scenario can represent unknown network errors or

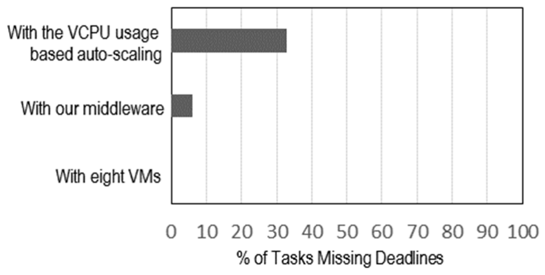

Figure 8 The percentage of subtasks missing deadlines.

possible cases intentionally increasing sensor's resolution. New medical devices are started to be discovered from 150 seconds, and at 210 seconds, all eight medical devices start sending tasks to request instructions. Tiny spikes shown on the slope mimic minor network errors such as congestions. Figure 7 shows the number of VMs running ICE application servers. When initializing our system, the ICE manager starts with eight VMs to prepare the worst case scenario receiving tasks from all eight devices. In our simulation, all eight VMs are initially structured as a balanced tree, and four autonomic managers in VMs have child VMs as managed resources. Until ten seconds, each autonomic manager checks its manager resources whether they process data or not. If they are not used to process tasks for this amount of time, the autonomic manager plans to terminate its managed resources. After ten seconds, it terminates its managed resources to achieve the higher server utilization. After 150 seconds, new medical devices are getting discovered and send tasks. Since our autonomic managers work independently from the ICE supervisor, the total number of VMs could be more than eight from our simulation for a moment. However, we believe that this issue can be fixed by enabling communications between our autonomic managers to balance the tree quickly. We compared our autonomic manager with the threshold-based auto-scaling mechanism which monitors VCPU usages. If it detects over 80% of VCPU usages, it launches a new VM in our simulation. Figure 8 shows the percentage of tasks missing their deadlines. As we can see, if we run eight VMs all the time without considering resource utilization and cost-efficiency, no deadline would be missed. However we lose the higher resource utilization of using cloud technologies. As using the VCPU usage based auto-scaling mechanism, ICE application servers missed deadlines of 240 tasks of total 735 subtasks as average values. But, after applying our mechanism, it only missed 45 subtasks even without a separated physical server only to run VMs. Our middleware improves 81.25% of the system reliability. We can see some tasks missed deadlines during the second catastrophic overflow after 150 seconds because of delays to launch new VMs. To overcome this drawback, we are adding workload prediction algorithms to our system to launch new VMs less frequently.

6 Conclusion and Future Work

Recently, cloud computing with virtualization technologies has become a big trend providing a new way to release software as a service and processing calculation-intensive tasks on remote VMs because it is very scalable, reliable, and cost-efficient with the on-demand computing model. Although most types of computing system and application can be migrated to cloud computing services, there are several serious issues when running medical CPS device applications. First, currently exiting cloud computing solutions with virtualization technologies do not have any special mechanism to support real-time and sensor-based applications. Second, to achieve the higher resource utilization when using the cloud, an auto-scaling mechanism is essential. However, existing performance metric based auto-scaling mechanism is not suitable to support medical CPS devices because of its inability to meet task deadlines. In order to support deadline-critical medical CPS devices following the MDPnP standard, we propose novel middleware running in a private cloud infrastructure with the ICE manager. This middleware with the autonomic manager uses self-management concepts from an autonomic computing architecture to develop our proposed auto-scaling mechanism for reserving virtual resources to meet timing constraints without human intervention. From our simulation, we have demonstrated that our approach can scale up when new devices are discovered. This research is still in progress as we develop more reliable workload prediction algorithms for already connected medical devices dynamically changing their task sizes. These prediction algorithms would enable our autonomic manager to launch new VMs even before the ICE supervisor requires more resources.

References

1 Amazon Elastic Compute Cloud. `http://aws.amazon.com/ec2/`.
2 Apache jCloud. `http://jclouds.apache.org/`.
3 R. K. Clark. *Scheduling dependent real-time activities.* PhD thesis, Carnegie Mellon University, 1990.
4 ASTM F2761-2009. Devices in the Integrated Clinical Environment.
5 Jeffrey O. Kephart and David M. Chess. The vision of autonomic computing. *IEEE Computer*, 36(1):41–50, 2003.
6 Tao Li and Jiannong Cao. Safety-ensured coordination of networked medical devices in mdpnp. Technical report, Hong Kong Polytechnic University, 2012.
7 S. Liu, G. Quan, and S. Ren. On-line scheduling of real-time services for cloud computing. In *Services (SERVICES'10), 2010 6th World Congress on*, pages 459–464, 2010. `http://dx.doi.org/10.1109/SERVICES.2010.109`.
8 C. D. Locke. *Best-effort decision making for real-time scheduling.* PhD thesis, Carnegie Mellon University, 1986.
9 M. Mao, J. Li, and M. Humphrey. Cloud auto-scaling with deadline and budget constraints. In *Proc. 11th IEEE/ACM Int'l Conf. Grid Computing (Grid'10)*, pages 41–48, 2010.
10 Medical Device Plug and Play Program. `http://www.mdpnp.org/`.
11 MIT-BIH Database Distribution. `http://ecg.mit.edu/`.
12 D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov. Eucalyptus opensource cloud-computing system. In *CCA'08: Cloud Computing and Its Applications, IEEE*, 2008.
13 OpenStack. `http://www.openstack.org/`.
14 B. Sotomayor, Ruben S. Montero, I.M. Llorente, and I. Foster. Virtual infrastructure management in private and hybrid clouds. *IEEE Internet Computing*, Vol. 13, 2009.
15 Ubuntu Cloud Image. `http://cloud-images.ubuntu.com/precise/current/`.
16 VMWare vCloud. `http://www.vmware.com/products/vcloud-hybrid-service`.