# Scheduling loT healthcare requests with high efficiency under fog computing environment

Li Cheng

Phone. (+61) 452 660 670 Email. lche206@student.monash.edu
FIT3036 Computer Science Project
Faculty of Information Technology, Monash University, Australia

## Abstract

Healthcare data come from implanted or wearable wireless sensors that continuously monitor a patient’s health status. These data are transmitted between sensor devices and remote datacenters at a large volume and often need rapid response. This study aims to achieve efficient data transmission between the smart gateways and the remote datacenters. With CloudSim simulator, I experiment on a number of different approaches for allocating virtual machines (vm) for huge amount of cloudlets. I developed a relatively efficient data transmission policy which allocates the relative small-size cloudlets to the relative small-size virtual machines and relative large-size cloudlets to the relative large-size virtual machines. The efficiency of this data transmission policy is demonstrated by comparing to the execution time of two other vm allocation policies. The results of this study are plotted on a bar chart in the Results section of this study.

## Keywords

Cloud Computing; fog computing; resource management; data transmission; performance evaluation.

## 1.0 Introduction

Cloud computing has a wide range of real-time applications such as social networking, content delivery, healthcare data request, real-time time data processing and transmission[13]. Optimizing the performance of scheduling and resource management policies are very important because 1) all these distributed and heterogeneous data creates ever-increasing streams of data, and 2) users are having higher and higher QoS requirements for these data streams[14]. In my case study of monitoring a group of patients distributed geographically, 1) cloud computing responds slowly to a huge stream of data, this is because the

datacenters are relatively scarce and often away from the hospitals and patients[12]. 2) though cloud computing has the compute and storage resources to perform advanced data analytics and decision-making[7], they are not economical due to the setup and maintenance cost. Fog computing, on the other hand, processes real-time data in a data hub on the smart device, close to where data is created[5]. In comparison to cloud computing's long-term and resource-intensive analytics, fog computing preprocesses the data and performs shorter-term analytics before sending them to the cloud server. It is an ideal middle layer between sensor networks and cloud servers.

The problem statement of this paper is to improve data transmission efficiency between smart gateways and remote datacenters. The objective of this study is to improve healthcare request processing time. In this study, I extend cloud computing with fog computing and simulate the fog computing environment using CloudSim, which is part of the experiment requirements. Experiments also include three resource scheduling policies and a comparison between the three. The constraint of this study is that the priority factors, referring to the last project specification that are known as distance to datacenter and urgency, satisfies QoS requirement but they do not improve the actual execution time, so it is largely ignored in this study.

The rest of the paper is organized as follows: In Section 2, the background of this study are discussed. Section 3 describes the project methods. Section 4 presents the experiment results. Section 5 discusses and analyses the experiment results. Section 6 discusses the future works of this study. Section 7 Conclusion.

# 2.0 Background

## 2.1 Related Works and Motivations

*Fog Computing Characteristics*

Al-Fuqaha *et al.* ( 2015) in their loT survey paper introduces the fog computing as a bridge between smart devices and large-scale cloud computing datacenters. Fog computing extends cloud computing to the edge of the network and offers better delay performance - proximity to the user, cost efficient - small fraction of cost compared to data center, scalability - flexibility to add micro fog centers rather than expensive data centers, and on-the-fly analysis - data preprocessing as opposed to huge amount of raw data sending to data centers[6]. Gia *et al.* ( 2016) discuss the enhancement of healthcare monitoring loT-based system in a diversified environment using fog computing. They described the QoS requirements of healthcare monitoring as 1) low latency, where care-givers or doctors can respond to patient's conditions in the smallest possible time; 2) scalability, where the fog can be expanded by simply setting up new smart gateways and distributed databases; 3) heterogeneity, where data can be collected from numerous devices in their different forms; and 4) real-time[3]. Bonomi *et al.* (2012) present another fog computing characteristic - geographical distribution, where fog demands widely distributed deployment in contrast to a centralized cloud datacenter. In their study the fog plays an active role in delivering high-quality streaming services to moving vehicles via widely distributed fog devices along tracks and highways. This is similar to healthcare monitoring in which a hospital receives patient's streaming healthcare data through proxies and access points positioned along household blocks[7].

*Fog Computing Applications in Healthcare Industry*

Cloud computing, IoT ( internet of things), fog computing are emerging technologies used to build smarter healthcare solutions. Recently, an IoT-based smart rehabilitation has been introduced to alleviate scarce healthcare resource problems. It is a healthcare system connecting all healthcare resources, such as, hospitals, rehabilitation centers, doctors, ambulances, healthcare devices, as a network to diagnose, monitor, and conduct remote surgeries over the internet. An automated resource allocator is used to schedule resources to meet individual patient's healthcare requirements( Yuehong, Zeng, Chen & Fan, 2016) [1]. In a second study, patients with long-term healthcare requirements can now be monitored at lower cost and in real time. In this study, Bulimia (eating disorder) patient can be safely monitored through the BAN ( body area network) to detect any abnormality in their body temperature, blood pressure, or even the odor of vomit in patients' homes. Patients suffered from Alzheimer's disease can be prevented from wandering or unwanted mobility behaviors using location service. Doctors or healthcare providers can simply place the sensors in patient's body or home to facilitate real-time medical assistance without the spatial and time limitations (Laplante & Laplante, 2016)[2].

*Challenges*

Mohammad *et al.* (2014) in their experiment, discovered that the synchronization delay for bulk-data ( multitude of files) takes almost 4 times as much as synchronizing a single file format (video/audio file, for example) [4]. Rahmani *et al.* (2015) in their paper discuss higher-level services offered by fog computing which greatly improve the energy efficiency, scalability, and reliability of the overall healthcare system. In their experiment, they constructed their own experimental tool UT-GATE, a Smart e-Health Gateway, to reduce the communication overhead in sensor nodes and perform security related tasks. The UT-GATE is a demonstrated smart gateway to reduce energy consumption, secure communication channel, and open for reconfiguration[5].

In Jiang *et al.* (2009), they discussed a scheduling algorithm schedules the data packets with the highest delay priority on individual network adaptors and transmit to the next hop, and queue management algorithm drops the packet with the lowest reliability priority when the queue at the backbone router is at maximum capacity. [8] Two things are achieved for individual packets being transmitted over network adaptors, 1) delay priority, 2) reliability priority. [8]. This study considers user's QoS requirements in the expense of dropping insignificant transmitting packet, which is not applicable in healthcare settings where each packet counts. Chebrolu *et al.* (2006) proposed a bandwidth aggregation scheme for real-time applications in heterogeneous wireless network, using earliest delivery path first (EDPF) scheduling to choose data packets from different applications to be transmitted over multiple wireless interfaces so that low latency requirement is achieved[9]. In Karetsos *et al.* (2005) a hierarchical radio resource management framework was designed for integrated WLAN and cellular network to alleviate the network congestion[10]. In Niyato *et al.* (2009) a stochastic programming (SP) problem was formulated and solved to give the number of reserved connections at a particular location which minimizes the expected cost of network access under the uncertainty of the number of patients in different locations in a service area. In the same paper, they present the solution to the optimization problem of scheduling bio signal data packets transmission at minimum latency and Constrained Markov Decision Process (CMDP) is formulated to address the delay requirement while maintaining minimum network access cost[11]. Based on the algorithms proposed from these studies, I am motivated to experiment three algorithms and find the most efficient implementation.

2.2 Project Information

Risk Analysis

| Risk | Likelihood | Impact | Mitigation |
|---|---|---|---|
| Commitment to the project | 2 | 5 | Make a project schedule (refer to Project Plan section) and stick to it. If a week’s schedule is incomplete on the first day of second week. Re-assign tasks as per personal timetable. |
| Project requirements incorrectly understood | 3 | 5 | Weekly workbook to be assessed by supervisor. Constant communication with supervisor, email, face-to-face. |
| CloudSim website maintenance | 1 | 2 | Refer to these two papers, or other helpful CloudSim resources.<br>1) CloudSim: a toolkit for modelling and simulation of cloud computing environments ;<br>2) Fog Computing in Healthcare - A Review and Discussion |
| Any hardware or software required for this project breaks down, lost | 1 | 5 | Move the software component of this project to Google Drive folder on the computer. If hardware breaks, |

| | | | |
|---|---|---|---|
| | | | repair or find replacement |

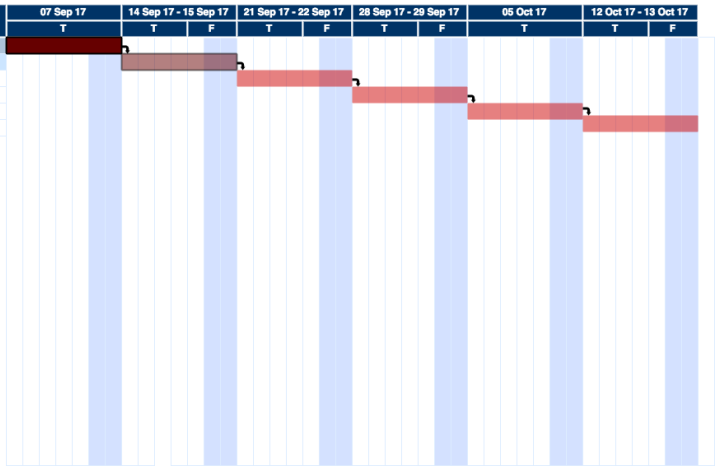
* Refer to *Appendix - Risk Register Legend* to understand the importance of this table

Resource Requirements

1) Hardware. The simulation is run on laptop-class machine MacBook Pro Mid 2015, processor 2.5 GHz Intel Core i7, memory 16 GB 1600 MHz DDR3.
2) Software. The simulation is run on operating system macOS Sierra version 10.12.6, IntelliJ IDEA CE 2017.2, and CloudSim 4.0 from CLOUDS laboratory of the University of Melbourne.

Project Tasks

| | Task Name | Duration | Start | ETA |
|---|---|---|---|---|
| 1 | Identify performance matrics | 1 days | 07.Sep.17 | 07.Sep.17 |
| 2 | Generate synthetic dataset | 2 days | 14.Sep.17 | 15.Sep.17 |
| 3 | Design test cases | 2 day | 21.Sep.17 | 22.Sep.17 |
| 4 | Implement cloudlet-scheduling algorithm | 2 days | 28.Sep.17 | 29.Sep.17 |
| 5 | Pass all test cases | 1 day | 05.Oct.17 | 05.Oct.17 |
| 6 | Evaluate algorithm effectiveness | 2 days | 12.Oct.17 | 13.Oct.17 |




# 3.0 Method

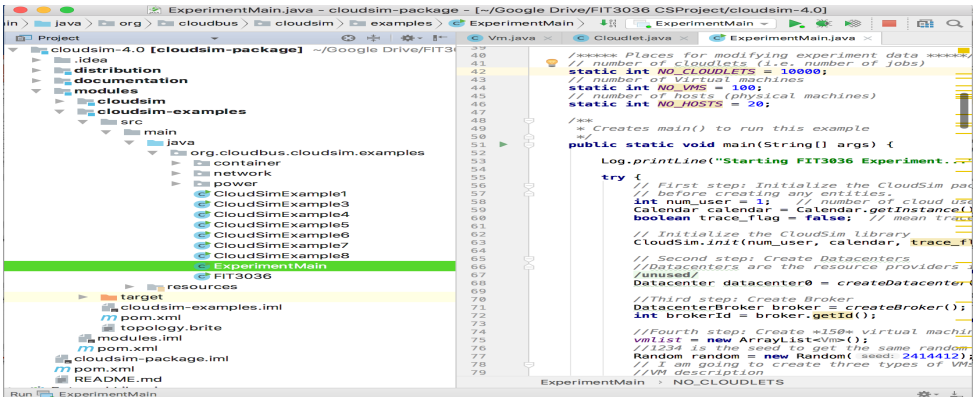
## 3.1 Project Methodology

Step 1

Download my project repository “cloudsim-4.0.zip”, extract the zip file and open the extracted file “cloudsim-4.0” in IntelliJ IDEA (my version is 2017.2.1)

Step 2

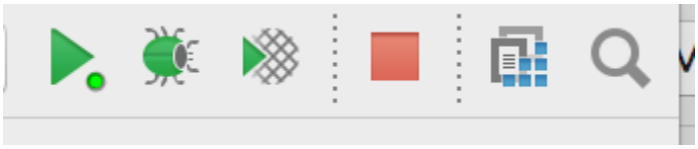
Within IntelliJ, locate and click on Modules>cloudsim-examples>src>main>java>mainExperimentMain file. You should be able to see the following picture now.



Step 3 (key)

You are now ready to run the experiment simulation with control+R or the green right-facing triangle (on mac computer). Before that, let’s get familiar with how to feed the experiment different variables so you can experiment with different size of input and have corresponding experiment results. You can simply do that by locating the comment “*/***** Places for modifying experiment data *****/*”, and modifying the “*NO_CLOUDLETS*” to be executed.

Note. Be aware, the execution time depends on NO_CLOUDLETS (aka. number of cloudlets), NO_VMS (aka. Number of virtual machines), NO_HOSTS (aka. Number of hosts), and the algorithm you choose to run the simulation (The “default” algorithm is random allocation which guarantees worst case performance. So for better execution time you need comment and uncomment some lines within Modules>cloudsim>src>main>java>DatacenterBroker file>submitCloudlets() function). For details of selecting different algorithm, refer to Step 4. A random allocation of 10,000 cloudlets takes “MacBookPro Mid 2015 2.5 GHz Intel Core i7 16GB 1600 MHz DDR3” more than one and a half hour to run, the execution time also depends on your computer technical specifications.



(The button to run the simulation, green right-facing triangle, top right of the IntelliJ Workbench)

Step 4 (key)

To experience another two algorithms and better execution time you need comment and uncomment some lines within Modules>cloudsim>src>main>java>DatacenterBroker file>submitCloudlets() function).

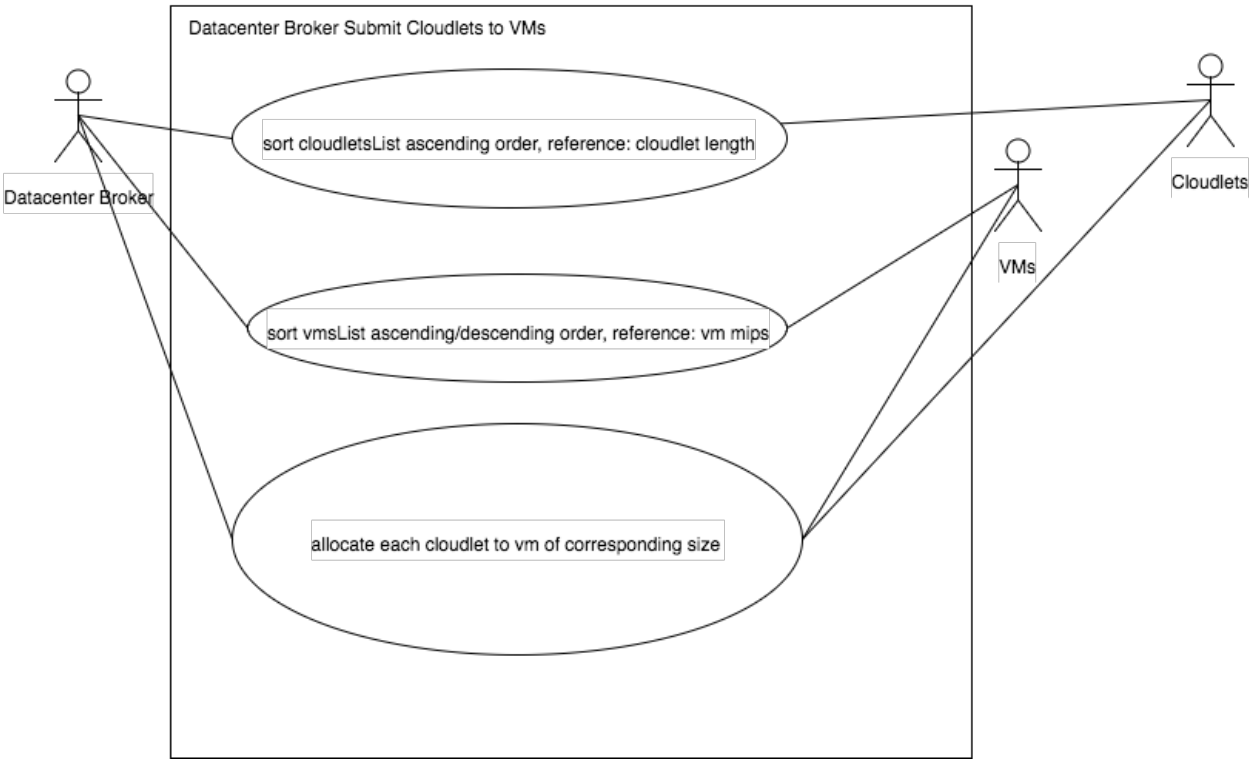
1) To run Approach 1. Random allocation of virtual machines. ( this is a “default” setting) Uncomment the line below the comment “// Approach 1...” and comment every line below the comment “// Approach2...”, “// Approach3...”, “// Approach 2 and Approach 3...”.
2) To run Approach 2. Smallest cloudlets to largest virtual machines and largest cloudlets to smallest virtual machines.

Comment the line below the comment “// Approach 1...” and “// Approach 3...”.
Uncomment every line below the comment “// Approach2...”, “// Approach 2 and Approach 3...”.

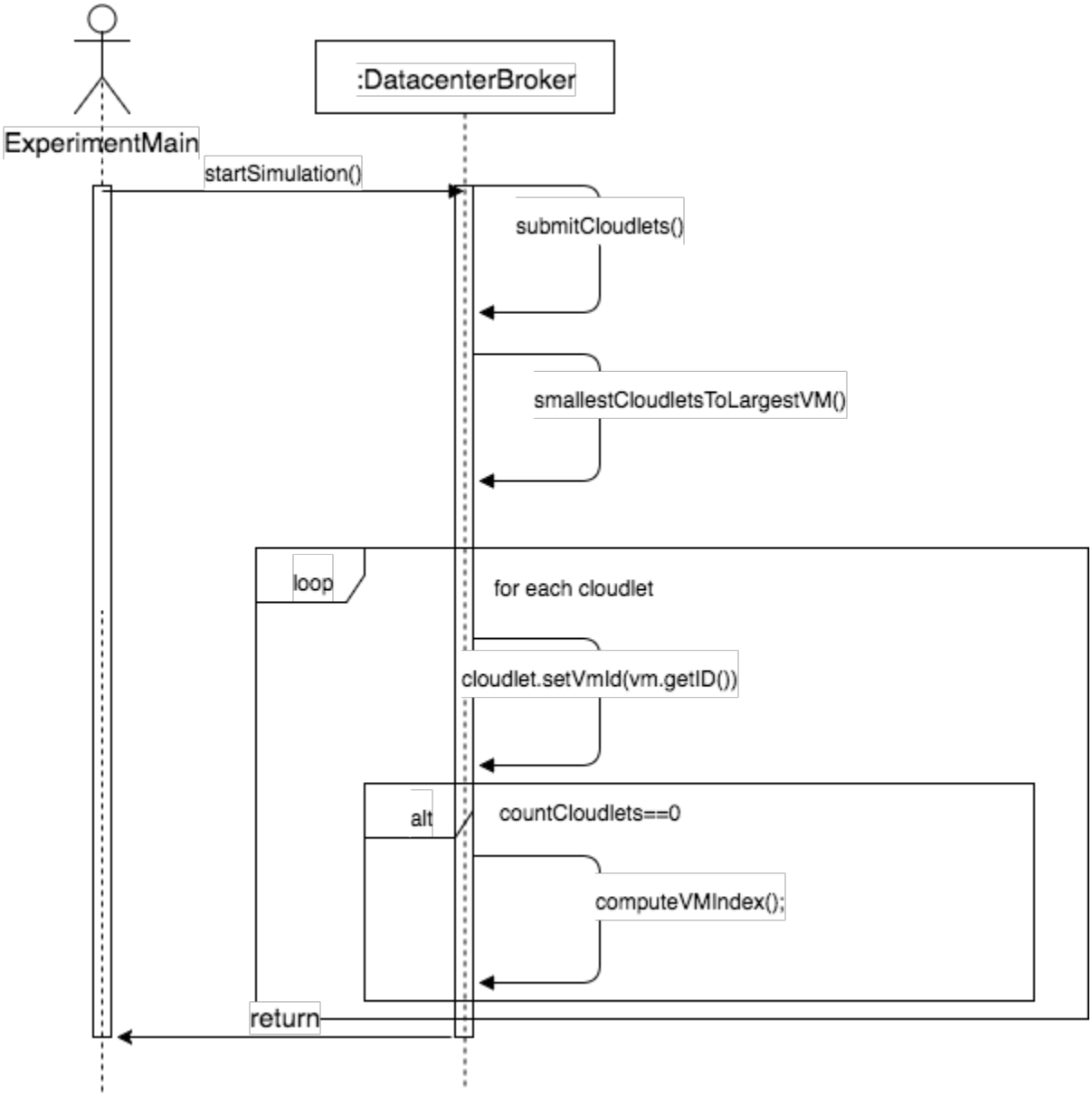
3) To run Approach 3. Smallest cloudlets to smallest virtual machines and largest cloudlets to largest virtual machines.
Comment the line below the comment “// Approach 1...” and “// Approach 2...”.
Uncomment every line below the comment “// Approach3...”, “// Approach 2 and Approach 3...”.

Step 5 (optional)
You may also wish to adjust the virtual machine, cloudlet, host, datacenter settings for more dynamic and realistic performance. They can be located in the aforementioned Modules>cloudsim-examples>src>main>java>mainExperimentMain file, though that is not mandatory for someone who only wishes to run the project and get some quick results.

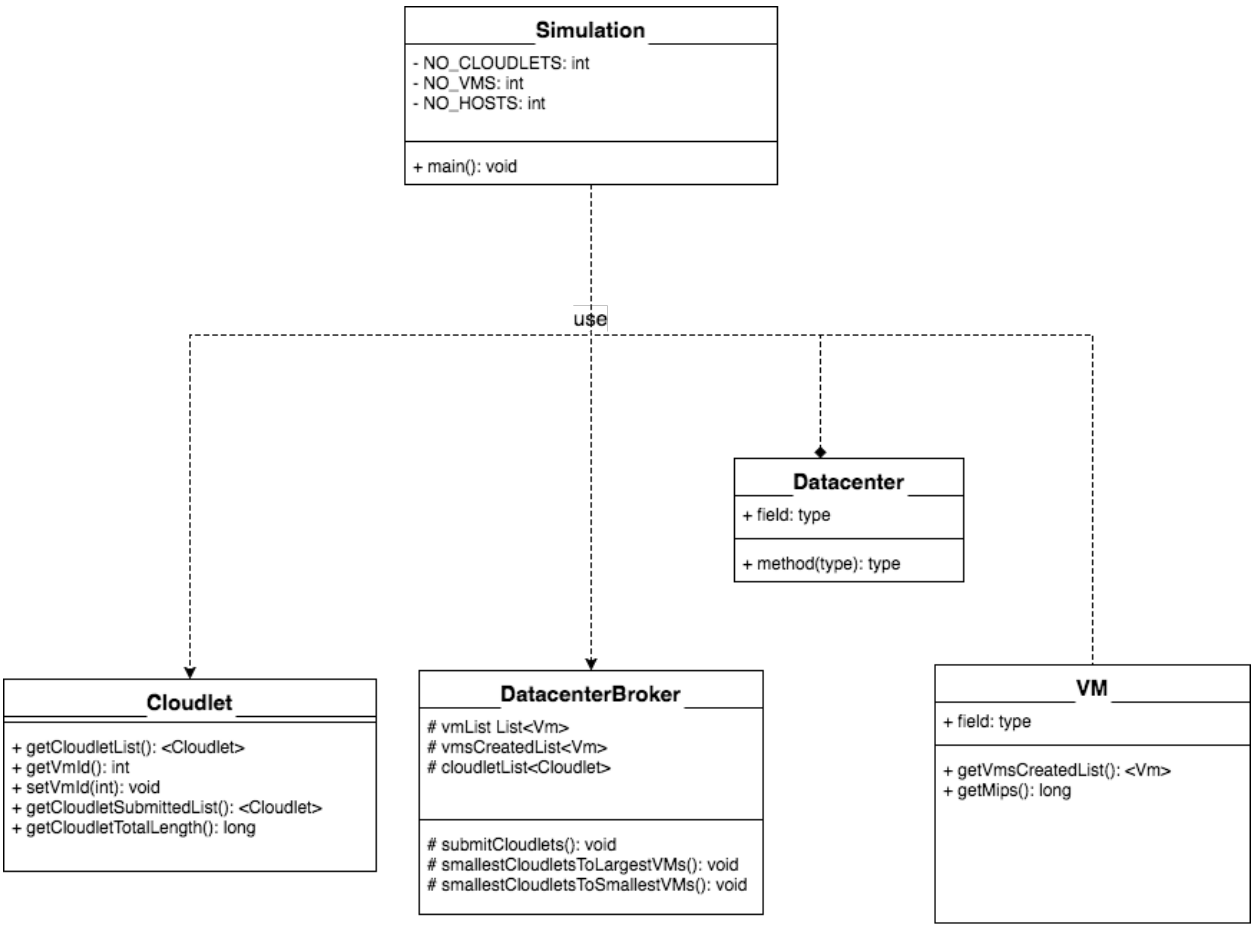
3.2 Internal Design
Use case diagram




Sequence diagram




3.3 Software architecture
Class diagram




3.4 Pseudocode

# Priority factors are largely left out even though it is first proposed in previous project specification. They satisfy the QoS requirements of the project, i.e.,urgent tasks are handle first so urgent patients are put first, but that does not make task of proper size run first. Which means they have no effect on the execution time of the simulation.
# I conclude, the priority factors does not satisfy the project objectives and therefore are left out.

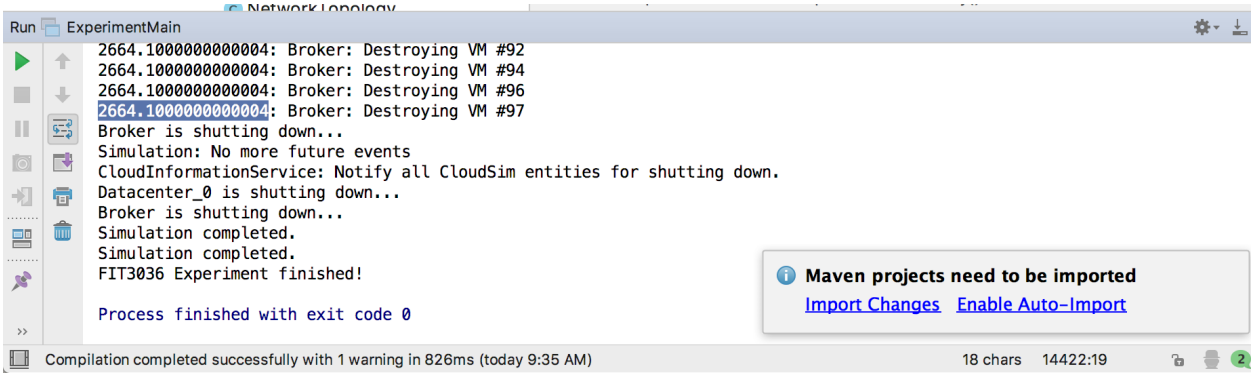
Datacenter broker submits cloudlets to vms():
    Sort cloudletList in ascending order, reference cloudlet length;
    Sort vmsList in ascending order, reference vm mips;
    Sort createdVmsList in ascending order, reference vm mips;

    For each cloudlet:
        Assign the smallest cloudlet to the largest/largest vm;

3.5 Statistics

The runtime of each simulation is captured. All runtimes are collected are plotted against different input task sizes as bar chart in excel.

To gather the run time, run the simulation and observe the CloudSim clock of destroying the last virtual machine. An example of finding the execution time ( relative time unit) is attached as below.

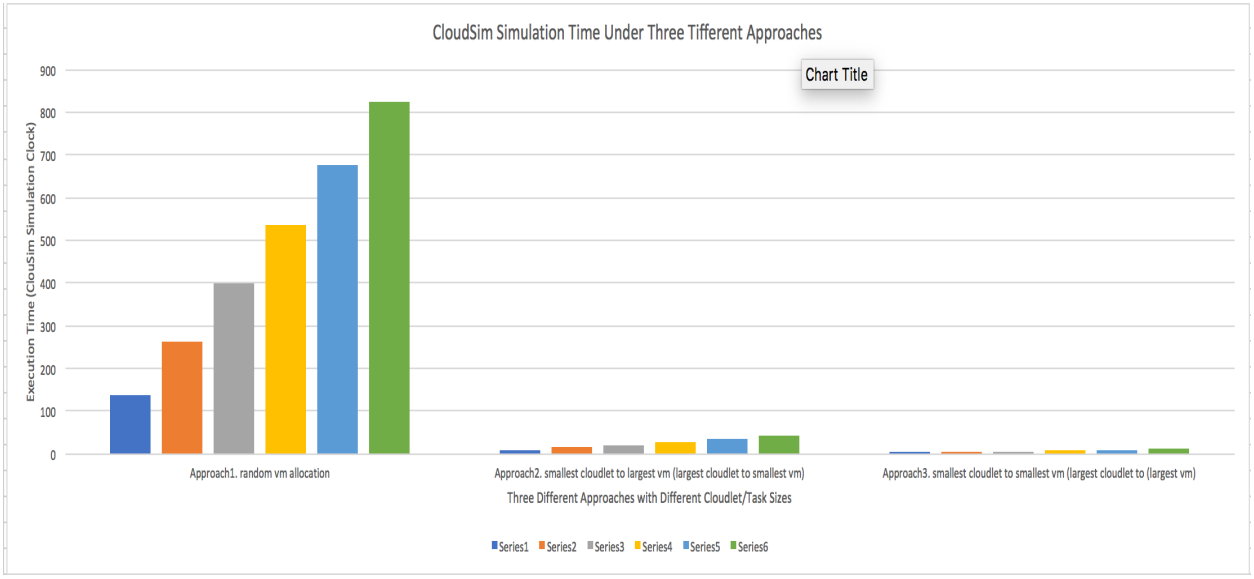



# 4.0 Results

Execution time table (Zoom in to view clearly)

| Data Table | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Fixed: NO_VMS=100, NO_HOSTS=20 | NO_CLOUDLETS | 500 | 1000 | 1500 | 2000 | 2500 | 3000 | 3500 | 5000 | 7500 | 10000 |
| CloudSim simulation clock (relative time) | Approach1. random vm allocation | 138.3605 | 262.8165 | 399.114 | 534.493 | 676.716 | 823.085 | 947.49 | 1331.126 | 2006.78 | 2664.1 |
| | Approach2. smallest cloudlet to largest vm (largest cloudlet to smallest vm) | 7.2965 | 14.492 | 21.689 | 28.884 | 35.284 | 42.475 | | | | |
| | Approach3. smallest cloudlet to smallest vm (largest cloudlet to (largest vm) | 1.899 | 3.698 | 5.497 | 7.296 | 8.896 | 10.694 | | | | |

Execution time plot




Experiment Steps

Refer to Section 3.1 Project Methodology and project source code.

Analysis of Results

Performance Improvement (Zoom in to view clearly)

| Peformance Improvement (%) Table | | | | | | | |
|---|---|---|---|---|---|---|---|
| Fixed: NO_VMS=100, NO_HOSTS=20 | NO_CLOUDLETS | 500 | 1000 | 1500 | 2000 | 2500 | 3000 |
| | Approach1. random vm allocation (base case) | n/a | n/a | n/a | n/a | n/a | n/a |
| | Approach2. smallest cloudlet to largest vm (largest cloudlet to smallest vm) | 94.73% | 94.49% | 94.57% | 94.60% | 94.79% | 94.84% |
| | Approach3. smallest cloudlet to smallest vm (largest cloudlet to (largest vm) | 98.63% | 98.59% | 98.62% | 98.63% | 98.69% | 98.70% |

The execution time performance improvement is significant, each of the other two algorithms results in more than 90% execution speedup. That is the result of allocating the cloudlet of proper size to the virtual machine of the proper size. Compared to the base case where random allocation between cloudlets and virtual machines are facilitated, the experiment results satisfy the project objectives - Scheduling loT health care requests with high efficiency under fog computing environment.

Note

The data points at number of cloudlets 3500 and above are left out, because these data points cannot be represented easily in the bar chart.

# 5.0 Analysis and Discussion

It was proposed in the project presentation (on week 12 Tuesday) that for best resource scheduling performance, the smallest task needs to be assigned to the largest vm ( the most capable vm). But the approach 3, assigning the smallest task to the smallest vm and the largest task to the largest vm, contradicts with my original belief. As stated by experiment results, approach 3 is the best performant resource scheduling algorithm with an average 4% improvement on top of approach 2, assigning the smallest task to the largest vm and the largest task to the smallest vm. Compared with random resource/vm allocation, my experiment demonstrates the important role a good resource scheduling algorithm plays in cloud and fog computing environment.

However, it is important to note that the algorithms conducted in this study are not the fastest way to assign resources. Firstly, during the execution each cloudlet is executed on its assigned vm from start to finish. If a virtual machine’s resources is not currently available, it will wait until its assigned virtual machine becomes available again. This waiting time can be minimized by migrating this “waiting-state” cloudlet to another available vm, most likely a vm whose host is not very far away from the previous vm. To implement this feature, one needs to be able to detect the cloudlets in its “waiting-state”, and assign a new vm whose resource is available.

Secondly from an individual perspective, the experiments carried out in this study is an individual third year computer science research project. Some of the analysis might be inaccurate. Potential readers should be critical in judging the accuracy of this study paper.

# 6.0 Future Work

As mentioned in last section, a more performant algorithm could be implemented by minimizing the waiting time of “waiting-state” cloudlet. Secondly, from an affordability point of view, this healthcare request service is to be provided to health practitioners who want to monitor patient’s critical status and

these patients. What these two parties want are fast response, as well as financially achievability. This viewpoint was presented during the project presentation such that the cost for processing, cost for ram, cost for bandwidth, cost for storage need to minimized with potential compromise on execution time. It means to sort the cloudlets, virtual machines one more time base on the cost criterions.

In conclusion, I can implement 1) a more performant resource scheduling algorithm, and 2) a cost-efficient resource scheduling algorithm. This concludes the Future Work section of the experiment report.

# 7.0 Conclusion

In this study, I proposed the problem statement of "how to schedule loT health care requests under fog computing environment with higher execution efficiency". I proposed three resource scheduling algorithms to address this problem. The experiment results features that when the relatively small cloudlets are assigned to the relatively small vm and when the relatively large cloudlets are assigned to the relatively large vm, the execution time is optimal. I conclude that future works extending this study can be carried out in finding 1) a more performant resource scheduling algorithm, and 2) a cost-efficient resource scheduling algorithm.

Bibliography

[1] Yuehong, Y. I. N., Zeng, Y., Chen, X., & Fan, Y. (2016). The internet of things in healthcare: an overview. *Journal of Industrial Information Integration*, *1*, 3-13.

[2] Laplante, P. A., & Laplante, N. (2016). The internet of things in healthcare: Potential applications and challenges. *IT Professional*, *18*(3), 2-4.

[3] Gia, T. N., Jiang, M., Rahmani, A. M., Westerlund, T., Liljeberg, P., & Tenhunen, H. (2015, October). Fog computing in healthcare internet of things: A case study on ecg feature extraction. In *Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing (CIT/IUCC/DASC/PICOM), 2015 IEEE International Conference on* (pp. 356-363). IEEE.

[4] Aazam, M., & Huh, E. N. (2014, August). Fog computing and smart gateway based communication for cloud of things. In *Future Internet of Things and Cloud (FiCloud), 2014 International Conference on* (pp. 464-470). IEEE.

[5] Rahmani, A. M., Thanigaivelan, N. K., Gia, T. N., Granados, J., Negash, B., Liljeberg, P., & Tenhunen, H. (2015, January). Smart e-health gateway: Bringing intelligence to internet-of-things based ubiquitous healthcare systems. In *Consumer Communications and Networking Conference (CCNC), 2015 12th Annual IEEE* (pp. 826-834). IEEE.

[6] Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., & Ayyash, M. (2015). Internet of things: A survey on enabling technologies, protocols, and applications. *IEEE Communications Surveys & Tutorials*, *17*(4), 2347-2376.

[7] Bonomi, F., Milito, R., Zhu, J., & Addepalli, S. (2012, August). Fog computing and its role in the internet of things. In *Proceedings of the first edition of the MCC workshop on Mobile cloud computing* (pp. 13-16). ACM.

[8] Jiang, S., Xue, Y., Giani, A., & Bajcsy, R. (2009, June). Providing QoS support for wireless remote healthcare system. In *Multimedia and Expo, 2009. ICME 2009. IEEE International Conference on* (pp. 1692-1695). IEEE.

[9] Chebrolu, K., & Rao, R. R. (2006). Bandwidth aggregation for real-time applications in heterogeneous wireless networks. *IEEE Transactions on Mobile Computing*, *5*(4), 388-403.

[10] Karetsos, G. T., Kyriazakos, S. A., Groustiotis, E., Di Giandomenico, F., & Mura, I. (2005). A hierarchical radio resource management framework for integrating WLANs in cellular networking environments. *IEEE Wireless Communications*, *12*(6), 11-17.

[11] Niyato, D., Hossain, E., & Camorlinga, S. (2009). Remote patient monitoring service using heterogeneous wireless access networks: architecture and optimization. *IEEE Journal on Selected Areas in Communications*, *27*(4).

[12] Kraemer, F. A., Braten, A. E., Tamkittikhun, N., & Palma, D. (2017). Fog Computing in Healthcare – A Review and Discussion. *IEEE Access*.

[13] Buyya, R., Ranjan, R., & Calheiros, R. N. (2009, June). Modeling and simulation of scalable Cloud computing environments and the CloudSim toolkit: Challenges and opportunities. In *High Performance Computing & Simulation, 2009. HPCS'09. International Conference on* (pp. 1-11). IEEE.

[14] Cardellini, V., Grassi, V., Lo Presti, F., & Nardelli, M. (2015, June). Distributed QoS-aware scheduling in Storm. In *Proceedings of the 9th ACM International Conference on Distributed Event-Based Systems* (pp. 344-347). ACM.

## Appendices

*Section 2.2. Risk Register Legend*

- Likelihood:
  - 5 - Very Likely to occur
    - 91-100% probability that risk will eventuate
  - 4 - Likely to occur
    - 61-90% probability that risk will eventuate
  - 3 - May occur about half of the time
    - 41-60% probability that risk will eventuate
  - 2 - Unlikely to occur
    - 11-40% probability that risk will eventuate
  - 1 - Very unlikely to occur
    - 0-10% probability that risk will eventuate
- Impact:
  - 5 - Extreme
    - May result in project failure
    - Budget overrun (in term of hours spent on assignment) could exceed 50%
    - Project delivered more than 50% late
  - 4 - High

        - Some required functionalities not delivered or not high quality.
        - Budget overrun 26-50%
        - Project late by 26-50%
    - 3 - Moderate
        - Required functionalities likely to still be delivered
        - Budget overrun 11-25%
        - Project late by 11-25%
    - 2 - Nominal
        - Budget overrun 5-10%
        - Project late by 5-10%
    - 1 - Minimal
        - Little to no impact on project