National College of Ireland

Implementation of Load Balancing Algorithm in Middleware System of Volunteer Cloud Computing

MSc Research Project
MSc in Cloud Computing

Gargee S Hiray
Student ID: x17154740

School of Computing
National College of Ireland

Supervisor: Mr.Vikas Sahni

National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Gargee S Hiray
Student ID:	x17154740
Programme:	MSc in Cloud Computing
Year:	2018
Module:	MSc Research Project
Supervisor:	Mr.Vikas Sahni
Submission Due Date:	28/01/2019
Project Title:	Implementation of Load Balancing Algorithm in Middleware System of Volunteer Cloud Computing
Word Count:	5200
Page Count:	18

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	
Date:	28th January 2019

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Implementation of Load Balancing Algorithm in Middleware System of Volunteer Cloud Computing

Gargee S Hiray
x17154740
MSc in Cloud Computing

28th January 2019

Abstract

Mobile phones are utilized widely by everybody as they are versatile and helpful. Their errand has changed basically from making telephone calls to performing high-end tasks. Volunteer computing is where volunteers donate their device processing capacity to a project and this process is executed through a middleware. BOINC is one of the middlewares which transmits tasks between cloud user and the device donor. There is an issue of low performance of the middleware. The fundamental reason for this is the volunteer node demands the BOINC server for assigning a task and this creates delay which degrades the overall performance of the middleware. By implementing a load balancer this issue could be reduced which will enhance the performance and provide better service to the customers.

1 Introduction

With the growing innovation in the field of technology cloud computing is a concept which everyone has come across might be because of business requirements or just for entertainment purpose. Cloud has boosted the computing power of the devices by enabling it to work more than its intended traditional use. It has also made the small business invest less and earn more with its feature of pay as you go, just pay for the required amount of time you use the service and make the maximum profit out of it Alonso-Monsalve et al. (2018). On an individual level cloud has made once life easy as anyone can access their data from the distinct location without the restriction of time. Taking about restriction cloud is now accessible and can be used with any device, computers are not only devices which can access cloud but high-tech devices such as mobile phone and tablets can also use cloud efficiently.

Mobile phones have become the need of an hour, the task which you use to do separately with various gadgets have now been combined in a form of single convenient device which is accessible, portable and user-friendly. Mobile use has increased tremendously over the last two decades and the reason for this increase is due to the functionality which it delivers. These new devices have enhanced the living standard for everyone with more flexibility and reliability. Mobile phones are always available with most of us which have not only made the communicate easy but has reduced the time and energy one needs to

put into performing certain tasks. While mobile manufacturers are launching new devices with increased configuration it is still not sufficient enough to process task which requires very high computation. Mobile cloud computing (MCC) is a concept which is a combination of cloud computing with a mobile phone as the device to operate it Alonso-Monsalve et al. (2018). As mobile devices are not capable to perform a certain high-end task due to the low configuration, through MCC mobile devices can connect to cloud in order to accomplish certain job by uploading the workload to cloud environment where the job is being performed and output is being transferred back to the mobile device. Despite, this being a good solution there are certain issues to it such as low bandwidth, insufficient storage, low battery life, and security.

Volunteer computing is a concept which is being implemented from quite some time, it is being used for donating computing power from your computer system in order to perform a task which requires systems with high configuration. It was basically innovated with the idea of supporting projects which are performing scientific innovation and lack the financial power to support high configuration system. SETI@Home [1] is the oldest project which was implemented using the concept of Volunteer computing. There have been many such projects which have been developed with this similar concept with different middleware. The drawback of using the cloud on mobile phones can be removed by combining the concept of volunteer cloud computing through mobile phones as a device instead of a computer system which will provide a better solution to this increasing demand of computation power in mobile phones. Mobile devices are not being used to their capacity, there is an instance when these devices are underused, such devices can be used to volunteer their computing capacity which will help to perform the task which requires more computing power and provide the output to the required client.

1.1 Relationship between Mobile Cloud Computing and Cloud Computing

Use of cell phones has expanded exponentially throughout the years, and they have progressed toward becoming a fundamental entity of our day to day life as we can convey them wherever we go. Their compactness is the motivation behind why they have turned out to be prominent and are being utilized widely by everybody. Cloud computing offers services which have also made a similar impact on technology making it more affordable for most of us. The three main services which cloud offers are Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS)Alonso-Monsalve et al. (2018) . Mobile cloud computing utilizes this element of Cloud to give an enhanced ordeal to its clients.

[1] `https://setiathome.berkeley.edu/`

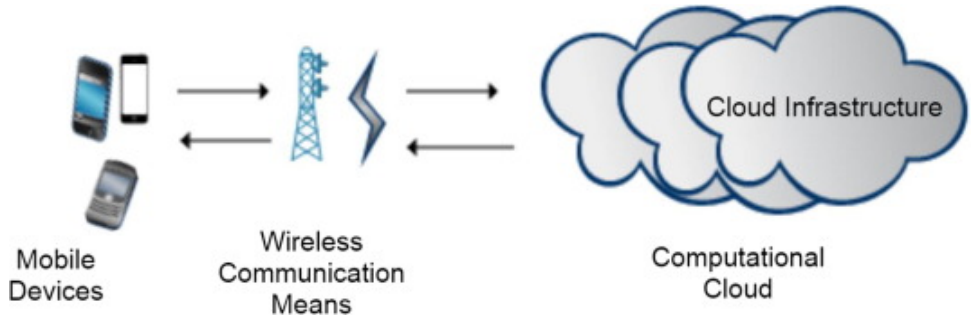

Figure 1: MCC and Cloud Computing

1.2 Middleware for Volunteer Cloud Computing

Berkeley Open Infrastructure for Network Computing (BOINC)[2] is a middleware which enables to perform Volunteer cloud computing. It operates in a way where BOINC client end needs to be installed on the volunteer device and BOINC server is the one which is responsible for the task distribution. Once the client end of the BOINC is installed on the volunteer device node has the benefit to pick how their cell phone would be utilized in the entire procedure of supporting the project. Volunteer computing is also referred to as distributed computing Alonso-Monsalve et al. (2018) where the workload is being distributed and output is being generated to satisfy the requirement of the client.

1.3 Problem Statement

BOINC distributes the task to the volunteered cell phones. While doing so it has a downfall where the process in which BOINC waits for the volunteer nodes to communicate to the server that it can assign a task for computation. In this scenario, there is wastage of time as well as resources as the communication between the BOINC server and BOINC client is single directional. This creates overhead on certain volunteer nodes and other nodes remain idle creating wastage of resources. Hence, to eradicate this drawback implementing load balancer will help in distributing the task and enhance the performance.

1.4 Research Question

Can the performance of volunteer computing be improved by using a load balancing algorithm in the middleware system?

1.5 Research motivation

Cloud computing is being used widely that means the number of users which use to connect to the cloud has increased rapidly. The increase in the number of users is also due to an increase in the high-end mobile devices which now support the cloud. This rapid growth in mobile users is now creating a network saturation making it difficult for mobile service providers Alonso-Monsalve et al. (2018). Volunteer computing comes into the picture to reduce this issue and create better and faster procession for the clients. While there is a time when the high-end gadgets i.e. mobile phones are not being used especially during the night this time can be utilized to perform certain tasks instead of

[2] `https://boinc.berkeley.edu/`

keeping them idle. Through load balancer, the task could be distributed amongst the node for better utilization.

1.6 Project structure

The following sections are as follows second section has the Literature review. In section three Research methodology has been explained. Section 4 contains the Design Specification followed by Section 5 which has Implementation next Section 6 has Evaluation and Section 7 Conclusion and last Section 8 contains Future work.

2 Literature Survey

2.1 Volunteer Computing and existing approach

The purpose of mobile devices has been evolved over a decade which was traditionally used only for communication. As proposed by Cushman et al. (2017) there are drawbacks in the innovation which are less storage, low durability of the battery and an increase in the consumption of data. To overcome this challenge the author has introduced a new method by which speed of the data would be increased and would also enhance the security. This project has been developed with the help of an open stack. The challenged faced during the implantation has been to segregate the mobile devices. Through this approach, the size of the data was being analysed and is assigned according to the available storage. Once this task is being performed load balancer needs to decide the most suitable node to fetch the data from. The data has been categorized in various types which include user created data, data generated from application and data which is developed due to the system. This incoming data needs to be predicated by the smart load balancer. Security of the cloud has been checked by creating multiple users. The load balancer has been implemented through simulation and is being performed on a small dataset which is a drawback as it will not be suitable if the data volume is high.

In this paper Alonso-Monsalve et al. (2018) the author has illustrated how mobile cloud computing is cost-effective, scalable and has demonstrated other benefits of it. It demonstrates how mobiles have become an important part of everyones lives and combining them with cloud will help the users. The BOINC middleware has been used for this project to support volunteer cloud computing. One of the challenges of Volunteer computing is heterogeneity which can be beneficial through the fog and mobile edge computing. Cloud system can be used by the business to implement this which will be cost-efficient and will be deployed easily. A heterogeneous mobile cloud computing model consolidates current mobile cloud architecture with the use of volunteer platform as resource providers. Client-side BOINC should be installed on the mobile of volunteer and cloud should install server-side software product of BOINC. Authors have additionally utilized an open source simulator for volunteer computing. Mobiles devices are being utilized when they are not being actually used or are being plugged in for charging which is during the night time when the phone is idle. This duration is favourable for performing tasks. The minimum charge in the battery should be 90 percent also it should be on sleep mode with the screen being off are the consideration to perform this project. As this is volunteer based computing the guarantee of the resources being available for computing all the time is very less. The drawback of this system is it does not have proper load

balancing which makes it difficult to allocate the task to the intended node.

Hierarchical brokering architecture (HIBA) has been proposed by Lin et al. (2015) for the mobile cloud environment. In this project Microsoft public cloud has been used where there are three modules cloud service interface node CSIN), CSI allocation table and CSI Manager on which load balancer has been implemented. Through load balancing algorithm traffic of high-frequency data has to be managed. Cloudlet tier which is nothing but public cloud needs to be connected by HIBA. In this client sends the request to the CSIN and RESTful is being used to obtain the result through interacting with them.

Ochi and Fukushi (2015) has proposed parallel volunteer computing job scheduling technique based on group. There are volatile nodes in case the resources leave the volunteer computing then the task allocated might not be completed. Communication amongst the resources is done by process execution in parallel computing. Which helps to execute multiple projects in parallel. While processing if certain working node leaves the task in between, it can cause the task to be incomplete. This paper has implemented job scheduling, where the jobs will finish their task without any disturbance. Credibility based voting where master performs the voting is being implemented. Credibility value is assigned which determines the correctness along with final outcome determines the highest credibility which is being derived through mathematical calculations providing accurate results. Group based scheduling is being implemented post finding the credibility. This proposal has not been implemented to retrieve results to prove its accuracy, however it is theoretically proposed.

Author Rochwerger et al. (2009) explains Mobile edge computing as mobile being at the network end of cloud computing. Computing capacity gets enhanced by using edge computing. Orthogonal frequency division multiple access (OFDMA) and Time division multiple access (TDMA) helps to offload the work allocation in multiuser mobile edge computing. Energy consumption while using mobile edge computing can be reduced through resource allocation. The ideal approach is being executed so that there is a need for clients relying upon the utilization of local energy. When the need is set once which are above and underneath are offered threshold in order to operate and for least offloading. Through this method allotment of the multiuser framework is improved the situation finite and infinite cloud computing. The offloading capacity has been offered need to actualize this method which gave the ideal solution.

2.2 Load balancing algorithms

Author Geetha and Robin (2017) explains load balancing as the technique which distributes the load by dividing or redistributing the existing load to available nodes whose existing load is less and for the nodes whose load is more the load gets reduced. This paper is a comparative study of load balancers. Scalability can be enhanced and bottleneck can be reduced through load balancing. There is majorly two category of load balancing: First is the static load balancing algorithm and the second is dynamic load balancing algorithm.

Chen et al. (2017) has proposed a load balancing scheme it is according to the priorities of the user, it is a modification to exiting min-min scheduling algorithm for load

balancing. Firstly, the minimum value of execution time for all the task has been found out post which maximum value has been selected. This algorithm was proposed for the static environment which was simulated on cloudsim.
Nakai et al. (2011) has proposed a load balancer for internet distributed services. This project has a load balancing technique for web servers which are distributed on large scale. In this project, the author has tried to reduce the response time by applying the limits on redirecting requests to other remote servers.

In this paper author Lu et al. (2011) introduced algorithm called as the Join-idle queue (JIQ), it reduces the communication overhead amongst the dispatches and processors who arrive at the job. This algorithm is based on distributed load balancing which is done by the distributed dispatcher. In the first step, all the distributed dispatchers make idle processors queue. Then they join these idle queues to assign the jobs coming to the servers to reduce the load of another overloaded node. It also claims to reduce the response time. This project has been implemented on the large system to determine how effectively the results are produced. Liu et al. (2011) has developed a multiprocessing load balancing algorithm. Multi-core system allows running multiple parallel load balancing task to enhance the performance. The main criteria while operating a load balancer task is to transfer requests to the server where the user has communicated before. This algorithm reduces the use of shared memory and improves the overall performance in a multicore environment.

Load balancing is a critical part of Cloud computing if there is no response from the cloud on time the users will consequently change to other technology and this will make the issue for cloud service providers. Cloud Computing Service providers (CCSPs) have defined a term know as Ranjan and Buyya (2010) Open Cloud Computing Federation (OCCF). The issue faced by service providers can be resolved by implementing a load balancer, it will enhance the workload by distributing the work evenly amongst the nodes. This will help to retain the trust of the clients as they will serve on time without facing any delays. In a situation where clients want an immediate response and they are stuck in where they are not able to reach to the server in such situation there will be frustration and would just leave the service. If users dont find the cloud to be reliable, they will not use the service which will impact the business for a cloud. To enhance the cloud service ant colony algorithm has been proposed by Zhang and Zhang (2010). Through this algorithm, the service providers will be able to speed up the response time and serve the customers in a better way.

Author Li et al. (2014) has proposed an improved scheduling algorithm which was based on a standard max-min algorithm for load balance in the elastic cloud. In this work, the author has maintained a task status table to store the estimated real-time load of VMs and approximate completion time of tasks. They have simulated this algorithm using cloudsim and has demonstrated the improvement in response time and resource utilization. In this paper Chopra and Singh (2013) author has proposed load balancing algorithm HEFT based workflow scheduling for cost optimization with a deadline in hybrid clouds. They have simulated their work on workflowsim. They have taken deadline completion as the main issue in load balancing for independent tasks and tried to reduce the overall cost.

Author Krishna (2013) has introduced an algorithm for the dynamic cloud-based algorithm which is a nature-inspired algorithm i.e. Honey bee behaviour. Author has aimed to maximize the throughput. The project has tried to distribute the load in such a way that the overall time required in the queue is reduced. They have shown the simulation on Cloudsim. Grid computing optimized load balancing algorithm has been proposed by Moradi et al. (2010). In this project, the algorithm is used to choose the task according to two criteria i.e. updated past status and min completion time. Author has tried to improve response time for the dynamic environment.

In Randles et al. (2010) paper has a comparison between different distributed load balancing algorithm for cloud computing. Author has proposed that there are 3 methods for large-scale load balancing in cloud-based systems- 1. Nature inspired 2. A random sampling of system domain, 3. A restructured system to optimize job assignments.Author Lee et al. (2012) has proposed a scheme profit-driven scheduling. Where the author has designed an algorithm to maximize profit with satisfactory service quality. They developed a policy based on prioritization for data service to maximize the profit of data service in a dynamic environment. This project has simulated this algorithm in CIC++. Algorithm which saves energy and which was based on vacation queuing theory for dynamic environment has been proposed by Cheng et al. (2015). Author has used vacation queuing model with exhaustive service to schedule the tasks. On the basis of busy period and busy time they have analysed the energy consumption of nodes. Project is being implemented through simulated algorithm using Matlab tool. Ergu et al. (2013) has proposed a framework for resource allocation based on tasks. Allocation of resources is done according to hierarchy, availability of resources and user preferences. Author has used some examples to prove their method validity.

Bhadani and Chaudhary (2010) has developed an algorithm for virtual machines based on central load balancing. This policy improves the overall performance of the system without considering fault tolerance. Project has been done through simulation using cloudsim. The concept of Grid has been implemented because of the cost of high configuration computers creating a network of computer resources. In this paper Etminani et al. (2009) author has developed a new scheme of load balancing considering two standard algorithms - min-min and max-min by utilizing their benefits and tried to reduce the completion time. They have called it min-min min-max selection algorithm. Author has evaluated the experiment with Gridsim in static environment.

3 Research Methodology

Cloud has provided a means of low budget execution with the addition of volunteer computing the task of executing a certain process will be much cost effective. Volunteer computing can be performed with the help of middleware which will be a junction between the node and the client. Taking about middleware BOINC is one of the middlewares which can be used to implement volunteer cloud computing. The task execution which takes place through the middleware needs to be well managed as the volunteered resource would not be available at all the time. In such a situation, it is important to distribute the task amongst the volunteer node with a load balancer which will keep track of the available nodes, the incoming tasks and the execution time. The load balancer will not

only increase the processing of the volunteer computing but will also provide efficient use of the available resources. To understand the concept of volunteer computing we need to understand the working of mobile cloud computing along with the middleware BOINC.

3.1 Architecture of Mobile Cloud Computing

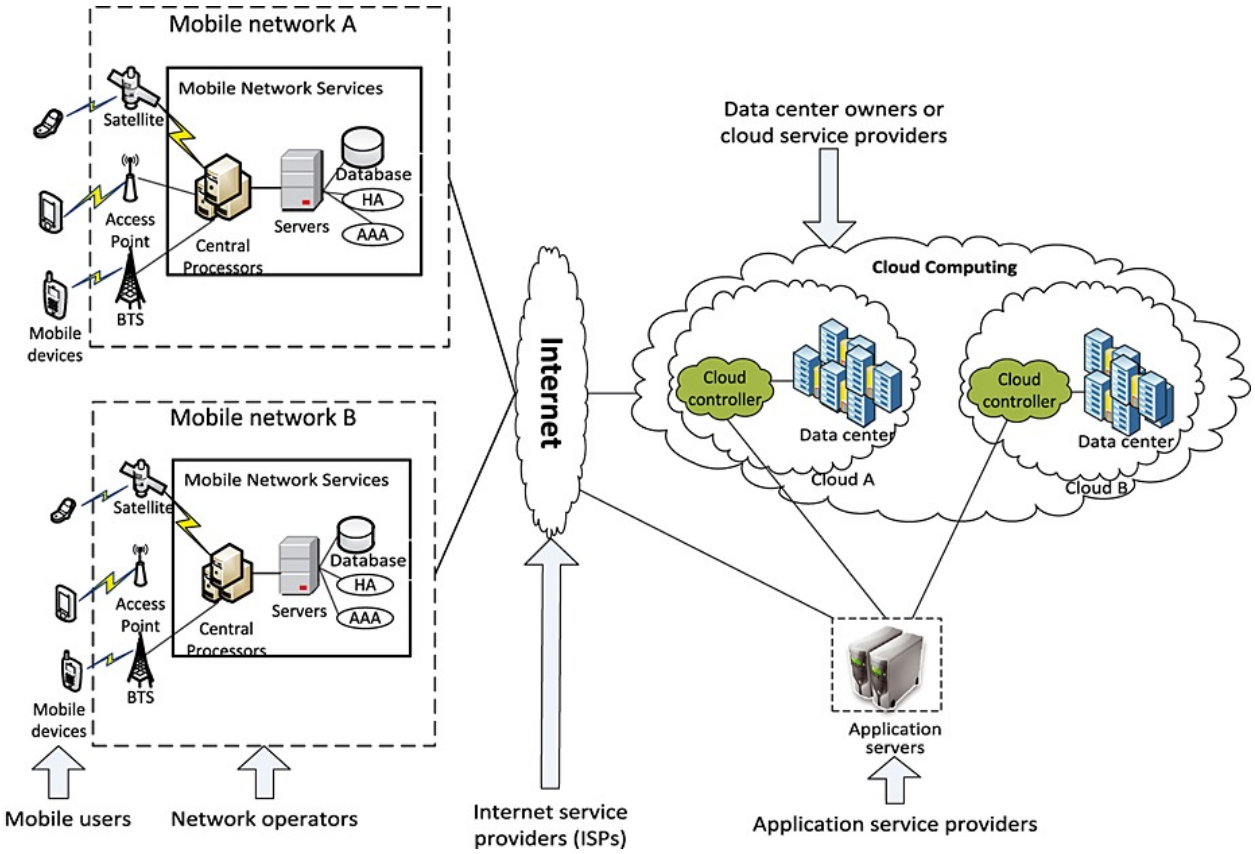

Figure 2: Architecture of Mobile Cloud Computing Dinh et al. (2013)

Figure [2] demonstrate the MCC architecture. Stations are the connecting point for all the mobile network to which the mobile devices are connected. Station helps to monitor the network and control the connection between the network and the mobile device Dinh et al. (2013). When there is a request which is generated by the mobile user it gets redirected to the central processor and it is the one which is connected to the service provider network. Authentication needs to be done by the network operator which helps to verify the exact mobile customer. Once the verification is done there is a request which gets a transfer through the internet to the cloud. The request which is sent by the customer needs to be addressed by cloud controller which processes the request and transmits the data to the customer who has initiated the request for the service.

3.2 BOINC Architecture

BOINC is an open source software which is used in volunteer computing as a middleware. To perform volunteer computing it needs to be installed on the mobile device which will be the volunteer node and also on the server side. The operation takes place once the setup is completed. Figure [3] The task which is requested by the client needs to be

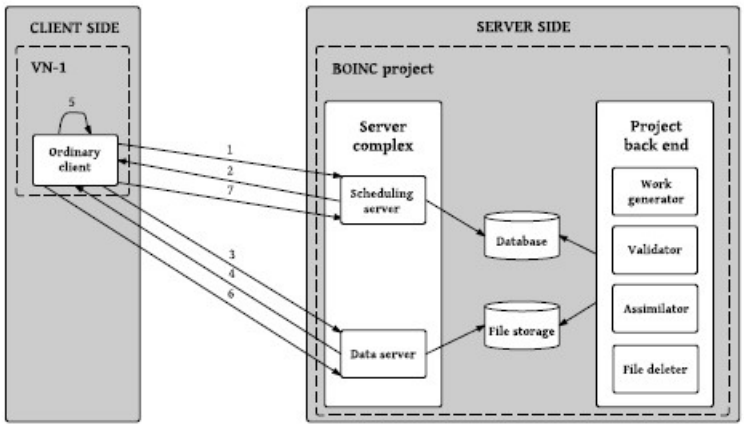

Figure 3: Architecture of BOINC Alonso-Monsalve et al. (2018)

transferred to the volunteer node through the BOINC server. The issue with BOINC is that the communication cannot be bi-directional that means the volunteer node can either communicate to BOINC server or BOINC server can communicate to the volunteer node but, at a time both cannot transmit information. Alonso-Monsalve et al. (2018) The existing process of BOINC is, client node request BOINC server for the task and only then the server assigns the task to the node, which means if the node is available and still does not request for the task then it will remain idle until it sends a request to the BOINC server requesting for the task. This creates wastage of resource hence load balancer needs to be implemented which will distribute the task to the node.

3.3 Proposed Architecture

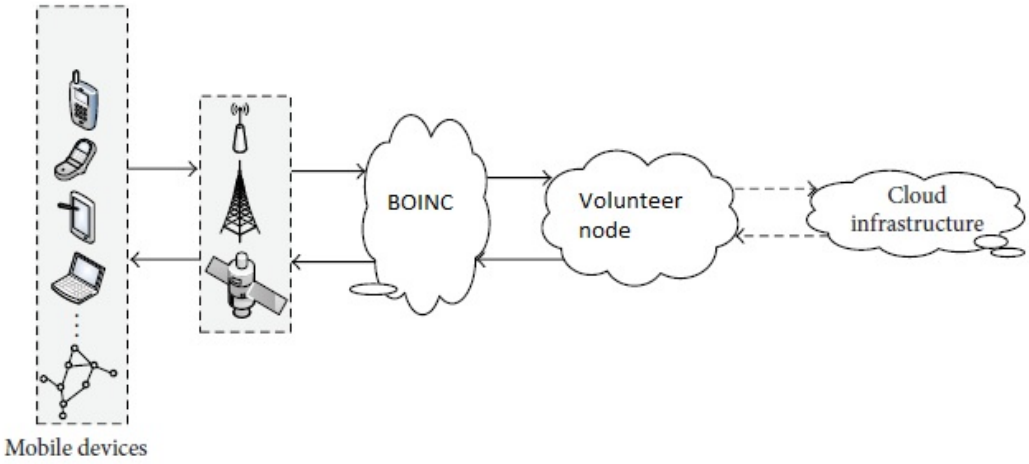

Figure 4: Basic Architecture

The basic architecture of the project is explained in Figure [4]. Here as we can see the first entity of the volunteer computing is mobile devices. The volunteers mobile devices are connected to the network station through which they will be able to communicate via the internet. The BOINC client end is installed on the mobile devices as the middleware which

will help to perform volunteer cloud computing. These volunteer nodes will communicate to the BOINC server to perform volunteer computing. As the concept of a volunteer is when the user wants to donate computing power of their mobile device only then the device becomes available for computing. In case the volunteer nodes are not available in such situation the operation will be performed on the cloud.

4 Design Specification

Figure [5] shows the workflow of the research. The very first step which initiates volunteer computing is the request from the mobile device. The client will first initiate the request to the middleware. Middleware will then send the request to load balancer to distribute the task to either the volunteer nodes or to the private cloud. The decision of load balancer will depend on the availability of the volunteer node, if the volunteer node is available then the task will be executed on it otherwise the operation will be carried out on the cloud. Either of the ways the task is performed finally, output will be generated which has been requested by the client.

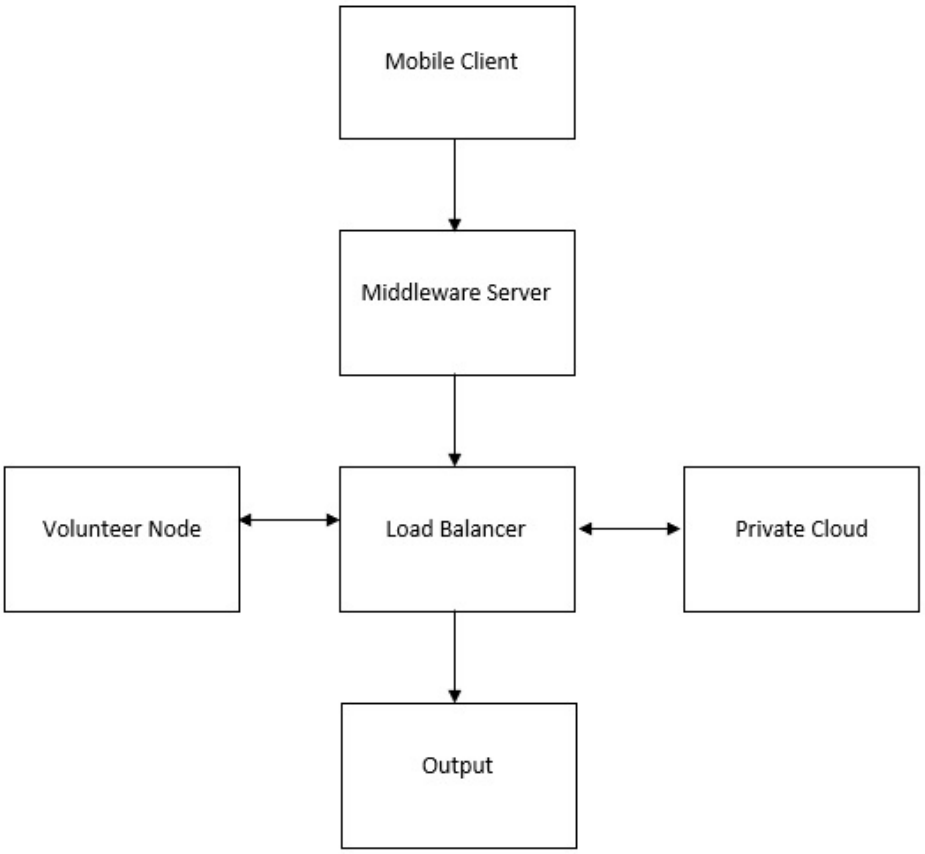

Figure 5: Work Flow Diagram

5 Implementation

Implementation of the research is performed on the basis of available volunteer node in MCC. There are certain criterias which are considered while implementation first if the

volunteer nodes are available for execution or not, second to locate the most appropriate node for execution as the memory size of the node needs to be considered while assigning certain task, if the node size is less than the process size in such situation the processing will not be possible, hence finding the most efficient node is crucial.

Chen et al. (2017) Agent-based load balancing algorithm has been implemented to distribute the task amongst various volunteer nodes. The agent-based algorithm aims to explore resources in the network also it collects information regarding the routing table. Search initiator node starts the search of the task, it sends numerous agents in the network to find out the intended node. The agent node has specific information through which they find out which node needs to be selected. The agent can get divided into a child agent which has the parent-child relationship which is used to find the intended node. This algorithm has been implemented to find the node.

Once the nodes are created, they are being arranged in ascending order this process is done because every process requires different memory to execute. If this process is not performed then if the first process which comes for execution requires a small amount of memory it will still be assigned with the first volunteer node this can lead to starvation of the nodes. Once the nodes are arranged in the ascending order it becomes easy for the load balancer to target the volunteer node to which the processing needs to be allocated.

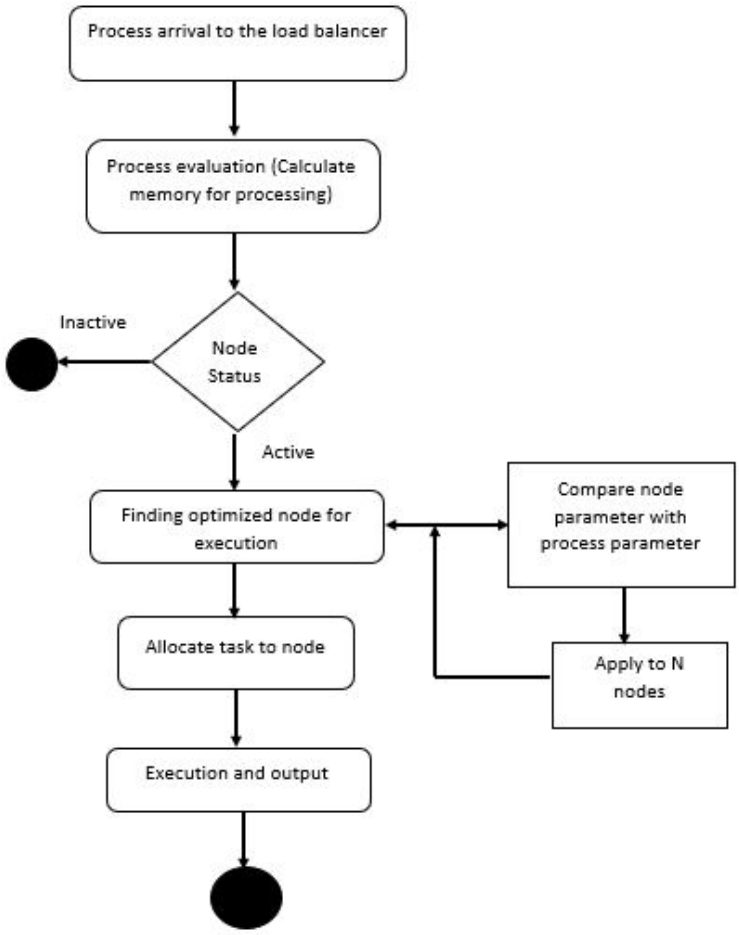

Figure 6: Working of Load balancer

Figure [6] explains the process of load balancer in detail. Once the process is received by the BOINC server it transmits them to the load balancer. Load balancer first evaluates the process by understanding the memory the process requires to execute. The next process is to check the status of the volunteer node if they are active or not. If the active nodes are located the next step is to find the node which is most suitable for the execution. This process is carried out by comparison of the parameters which are the memory size required for the execution should be greater than equal to the memory required by the process to execute It will perform if else loop which is explained in figure[7]. Finally, once the node is selected by the load balancer it allocates the task to the volunteer node.

```
if(noderam>=pram){

            alloted=true;
            //update node status to allloted
            PreparedStatement update_prstm=con.prepareStatement("update
node set isalloted=1 where nid="+rs34.getString(1));
            update_prstm.executeUpdate();
            //update process to execution
            PreparedStatement update_process=con.prepareStatement("update
process set isexecuted=1 where pid="+rs11.getString(1));
            update_process.executeUpdate();
            break;
        }
       }

       }else{

<tr>
<td><b><%out.println(processname); %></b></td>
<td><p class="label label-danger">Execute on Cloud</p></td>


       }

}
```

Figure 7: Code for node selection

In this research, there are two main components the volunteer nodes and the processes.

First volunteer node needs to be created Figure [8]. While this process is a simulation of the actual mobile node there is certain parameter which needs to be filled in order for the node creation. Node needs to have a title; RAM size needs to be specified in MB this is one of the important parameters which needs to be paid attention to as this will be the deciding factor for which process should be sent to which node depending on whether the RAM required for the task execution will be sufficient enough in the node which is created. The next is what time of the operating system is used, next is the processor type and then last is the node description.

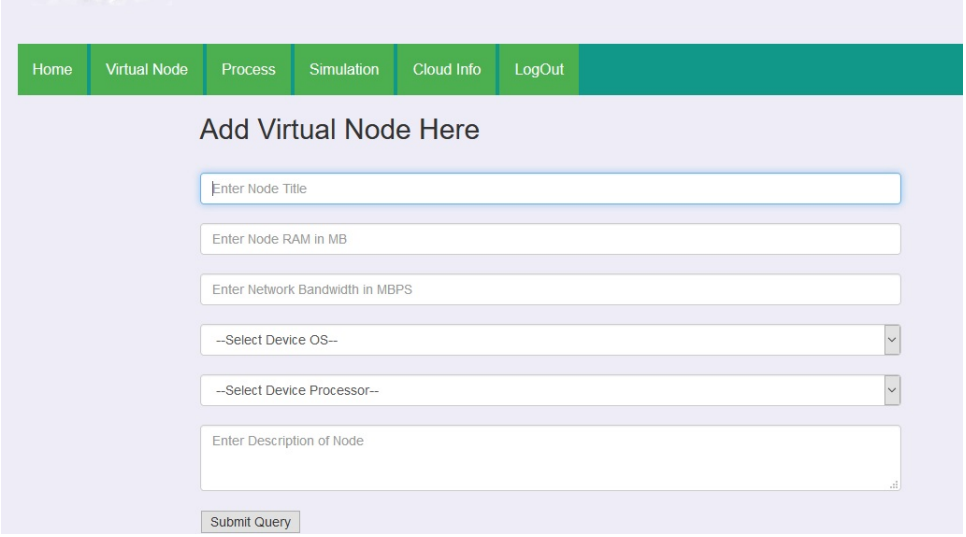

Figure 8: Volunteer Mobile Node creation

Figure [9] As the processing of the volunteer computing totally depends on the availability of the node, through this window it is possible to the user to make their volunteer device to be active or inactive. Complete control of the device remains with the user and they can easily decide if they want to provide their device for computing or not. Similarly, they can also remove their device from the project.

Figure 9: Status update of the volunteer node

Figure [10] As explained earlier for simulation purpose the nodes and process are being created. While creating the process the first information required is what is the function of the process, then the memory required for the process, similar to node creation step this is also an important information as the node selection will depend on the required size of the process. Next is the description of the process and last is what time is required to execute the process. Please note as this is not done in the real-time environment it will not affect the execution, however, in real time scenario, this input will determine after how much time the process needs to stop. The process can also be updated through the panel and it can also be removed if it is no longer required by the client.

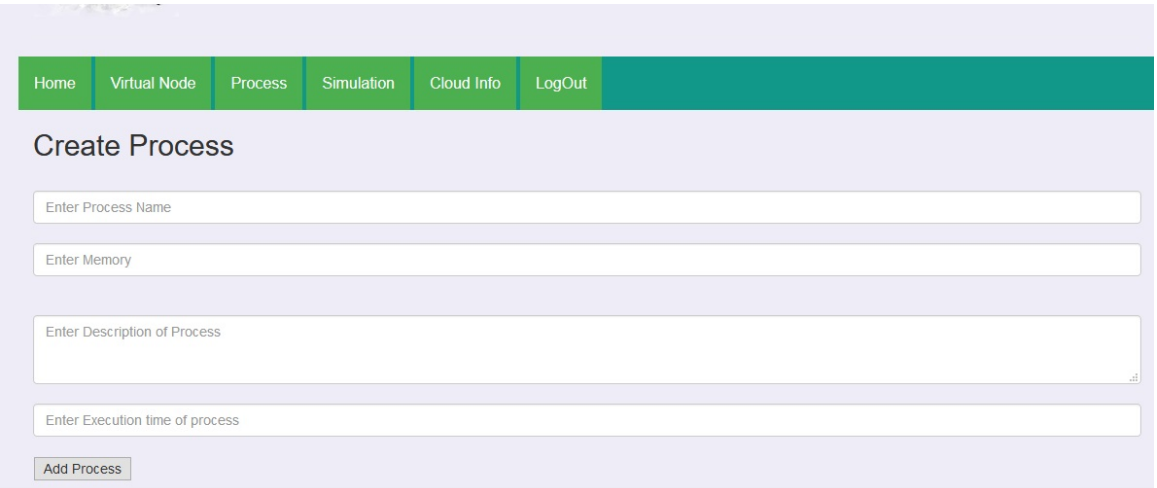

Figure 10: Process creation

6 Evaluation

There are 4 volunteer nodes which are created and there are 3 processes which are requested. Out of the 4 volunteer nodes one node is inactive Figure [11]

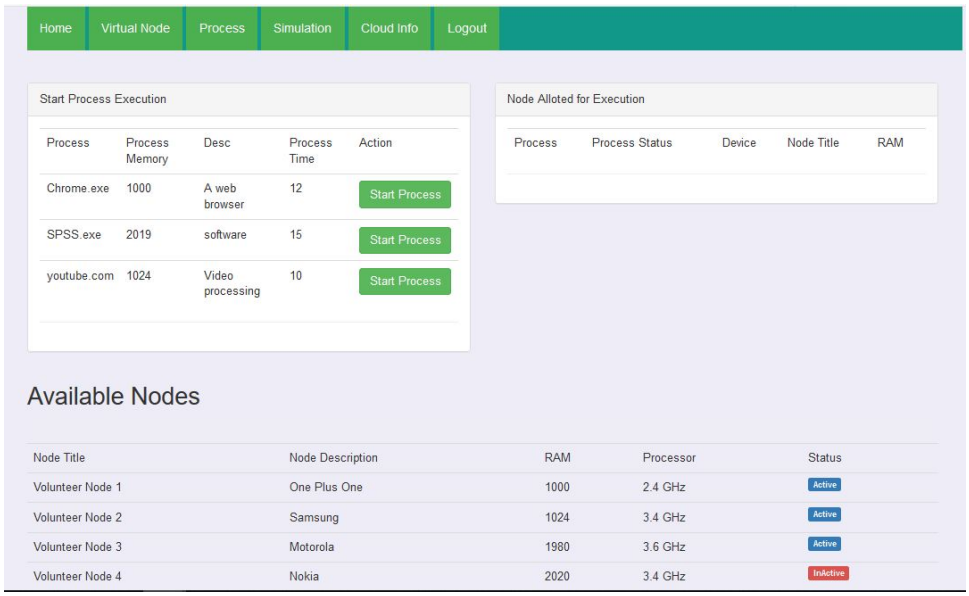

Figure 11: Volunteer node status

Figure [12] Initially a process is executed for a web browser which requires 1000 MB memory. It gets assigned to node 1 as it is the closed to the required memory size.

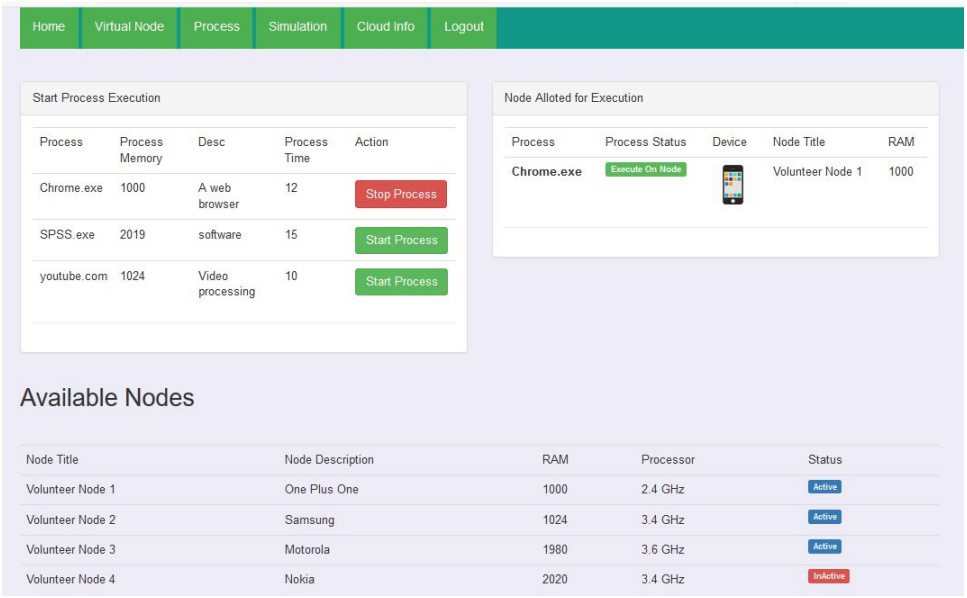

Figure 12: Execution of a web browser

Figure [13] However, when process SPSS.exe is requesting for a node, there is no suitable node available. Hence the operation is being performed on cloud.

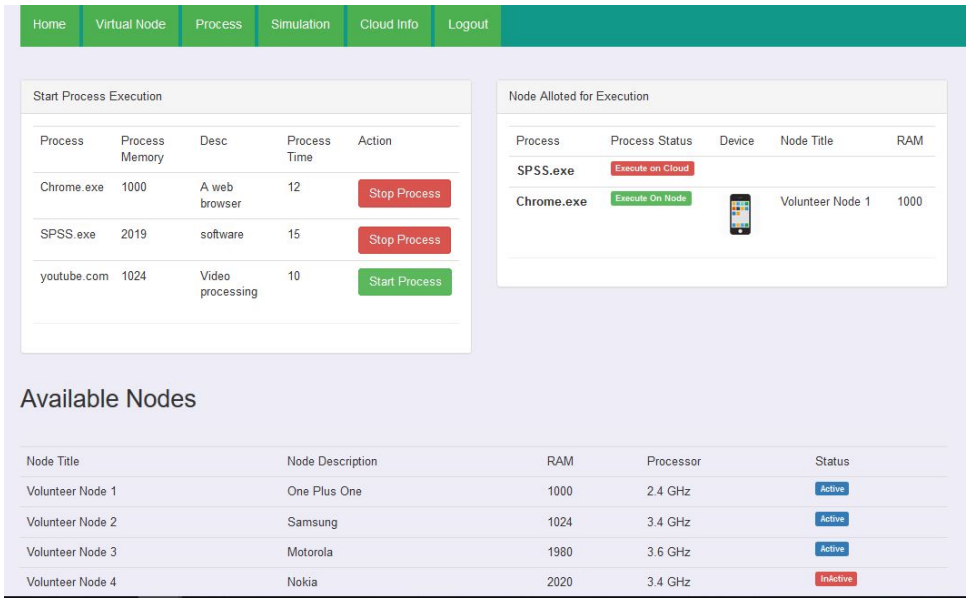

Figure 13: Execution of SPSS on cloud

Figure [14] Later the volunteer node becomes availabe whose memory is suitable for processing SPSS.exe then the procession is moved from cloud to volunteer node 4.

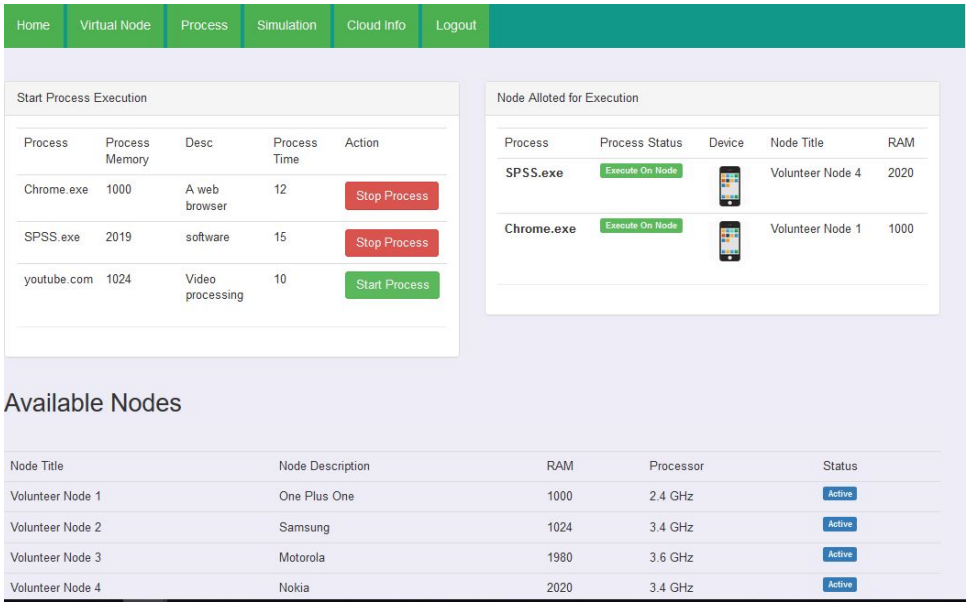

Figure 14: Processing of SPSS on Volunteer node

Figure [15] demonstrate the working of the load balancer stepwise as discussed in the Implementation section.

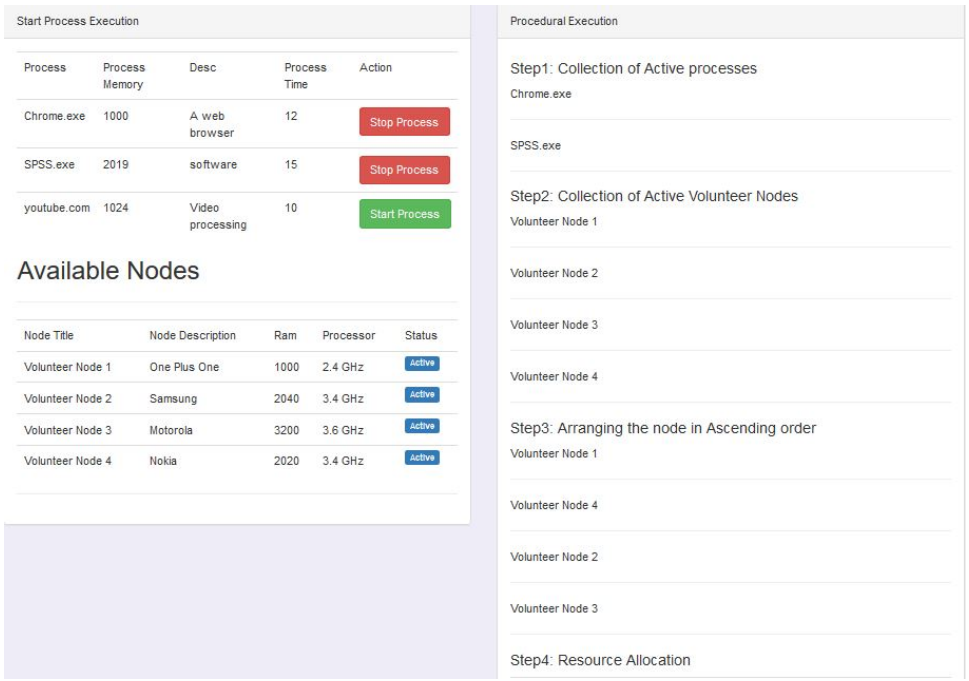

Figure 15: Load balancer execution steps

7 Conclusion

The main idea of this research has been to enhance the middleware used in volunteer computing to provide an improved solution to the user. BOINC is the middleware which was selected for this process whose drawback is it cannot allocate task efficiently as it can only function unidirectional. The load balancer has been added to improve the process

of task allocation which will not only increase the processing speed but will also reduce the wastage of the volunteer nodes. Through this, the performance of the volunteer computing will be enhanced.

8 Future Work

In the future to enhance the load balancing bandwidth and certain other parameters could be looked which will increase the accuracy to determine the exact node which is suitable for the processing. Also, there are various other algorithms available which could be used to check the efficiency to provide a better alternative to the existing solution

Acknowledgement

I would like to sincerely thank my supervisor Mr. Vikas Sahni, Professor in Computing at National College of Ireland who has encouraged and guided me to perform better throughout this research. His valuable inputs and suggestions have helped me to structure the research in the right direction. I would also like to take this opportunity to thank Dr. Nour Mawas who helped me by giving continuous feedbacks in the initial stages of research. Lastly, to all the staff of NCI for providing such wonderful guidance and support throughout my course.

References

Alonso-Monsalve, S., García-Carballeira, F. and Calderón, A. (2018). A heterogeneous mobile cloud computing model for hybrid clouds, *Future Generation Computer Systems* .

Bhadani, A. and Chaudhary, S. (2010). Performance evaluation of web servers using central load balancing policy over virtual machines on cloud, *Proceedings of the Third Annual ACM Bangalore Conference*, ACM, p. 16.

Chen, C.-A., Stoleru, R. and Xie, G. G. (2017). Energy-efficient load-balanced heterogeneous mobile cloud, *Computer Communication and Networks (ICCCN), 2017 26th International Conference on*, IEEE, pp. 1–9.

Cheng, C., Li, J. and Wang, Y. (2015). An energy-saving task scheduling strategy based on vacation queuing theory in cloud computing, *Tsinghua Science and Technology* **20**(1): 28–39.

Chopra, N. and Singh, S. (2013). Heft based workflow scheduling algorithm for cost optimization within deadline in hybrid clouds, *Computing, Communications and Networking Technologies (ICCCNT), 2013 Fourth International Conference on*, IEEE, pp. 1–6.

Cushman, I. J., Al Sadi, M. B., Chen, L. and Haddad, R. J. (2017). A framework and the design of secure mobile cloud with smart load balancing, *Mobile Cloud Computing, Services, and Engineering (MobileCloud), 2017 5th IEEE International Conference on*, IEEE, pp. 205–210.

Dinh, H. T., Lee, C., Niyato, D. and Wang, P. (2013). A survey of mobile cloud computing: architecture, applications, and approaches, *Wireless communications and mobile computing* **13**(18): 1587–1611.

Ergu, D., Kou, G., Peng, Y., Shi, Y. and Shi, Y. (2013). The analytic hierarchy process: task scheduling and resource allocation in cloud computing environment, *The Journal of Supercomputing* **64**(3): 835–848.

Etminani, K., Naghibzadeh, M. and Yanehsari, N. R. (2009). A hybrid min-min max-min algorithm with improved performance, *Department of Computer Engineering, University of Mashad* .

Geetha, P. and Robin, C. R. (2017). A comparative-study of load-cloud balancing algorithms in cloud environments, *2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS)*, IEEE, pp. 806–810.

Krishna, P. V. (2013). Honey bee behavior inspired load balancing of tasks in cloud computing environments, *Applied Soft Computing* **13**(5): 2292–2303.

Lee, Y. C., Wang, C., Zomaya, A. Y. and Zhou, B. B. (2012). Profit-driven scheduling for cloud services with data access awareness, *Journal of Parallel and Distributed Computing* **72**(4): 591–602.

Li, X., Mao, Y., Xiao, X. and Zhuang, Y. (2014). An improved max-min task-scheduling algorithm for elastic cloud, *Computer, Consumer and Control (IS3C), 2014 International Symposium on*, IEEE, pp. 340–343.

Lin, T.-C., Pai, M.-Y., Chen, C.-L. and Chen, C.-C. (2015). Load-balanced cloud service interface for the hiba mobile cloud environment, *Consumer Electronics-Taiwan (ICCE-TW), 2015 IEEE International Conference on*, IEEE, pp. 360–361.

Liu, X., Pan, L., Wang, C.-J. and Xie, J.-Y. (2011). A lock-free solution for load balancing in multi-core environment, *Intelligent Systems and Applications (ISA), 2011 3rd International Workshop on*, IEEE, pp. 1–4.

Lu, Y., Xie, Q., Kliot, G., Geller, A., Larus, J. R. and Greenberg, A. (2011). Join-idle-queue: A novel load balancing algorithm for dynamically scalable web services, *Performance Evaluation* **68**(11): 1056–1071.

Moradi, M., Dezfuli, M. A. and Safavi, M. H. (2010). A new time optimizing probabilistic load balancing algorithm in grid computing, *Computer Engineering and Technology (ICCET), 2010 2nd International Conference on*, Vol. 1, IEEE, pp. V1–232.

Nakai, A. M., Madeira, E. and Buzato, L. E. (2011). Load balancing for internet distributed services using limited redirection rates, *2011 Latin-American Symposium on Dependable Computing*, IEEE, pp. 156–165.

Ochi, K. and Fukushi, M. (2015). A group-based job scheduling method for parallel volunteer computing, *Computing and Networking (CANDAR), 2015 Third International Symposium on*, IEEE, pp. 571–575.

Randles, M., Lamb, D. and Taleb-Bendiab, A. (2010). A comparative study into distributed load balancing algorithms for cloud computing, *Advanced Information Networking and Applications Workshops (WAINA), 2010 IEEE 24th International Conference on*, IEEE, pp. 551–556.

Ranjan, R. and Buyya, R. (2010). Decentralized overlay for federation of enterprise clouds, *Handbook of Research on Scalable Computing Technologies*, IGI Global, pp. 191–217.

Rochwerger, B., Breitgand, D., Levy, E., Galis, A., Nagin, K., Llorente, I. M., Montero, R., Wolfsthal, Y., Elmroth, E., Caceres, J. et al. (2009). The reservoir model and architecture for open federated cloud computing, *IBM Journal of Research and Development* **53**(4): 4–1.

Zhang, Z. and Zhang, X. (2010). A load balancing mechanism based on ant colony and complex network theory in open cloud computing federation, *Industrial Mechatronics and Automation (ICIMA), 2010 2nd International Conference on*, Vol. 2, IEEE, pp. 240–243.

National College of Ireland

Configuration Manual

MSc Research Project
MSc in Cloud Computing

Gargee S Hiray
Student ID: x17154740

School of Computing
National College of Ireland

Supervisor: Mr.Vikas Sahni

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Gargee S Hiray
Student ID:	x17154740
Programme:	MSc in Cloud Computing
Year:	2018
Module:	MSc Research Project
Supervisor:	Mr.Vikas Sahni
Submission Due Date:	20/12/2018
Project Title:	Configuration Manual
Word Count:	400
Page Count:	5

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	
Date:	20th December 2018

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Gargee S Hiray
x17154740

1 System Overview

1.1 Software Requirement

- Operating system: Windows 10
- JDK 7.0
- Apache Tomcat 7.0
- MySql 5.5
- Eclipse Luna IDE

1.2 Hardware Requirement

- Processor: Intel Core i5-8250U CPU @1.60GHz 1.80 GHz
- Memory: 8.00 GB
- HDD: 1 TB
- MySql 5.5
- System Type: 64- bit operating system, x64 based processor

2 Installations

2.1 Installation of JDK 7.0

It is a stable version for eclipse luna
Step 1. Download JDK 7.0 from oracle link [1]
Step 2. Once the download is complete double click to start the installation.
Step 3. Accept the licence agreement
Step 4. Follow the instructions on the Installation wizard.

[1] https://www.oracle.com/technetwork/java/javase/downloads/java-archive-downloads-javase7-521261.html

2.2 Installation of Apache Tomcat 7.0

It is an open source and is compatible with javaEE
Step 1. Download apache tomcat from the link [2]
Step 2. Once the download is complete double click to start the installation
Step 3. Follow the instructions on the installation wizard.

2.3 Installation of Eclipse

Step 1. Download and Eclipse Luna from the link [3]
Step 2. Unzip the file, double click and follow the instruction on the installation widard.
Step 3. Click on eclipse icon to start
Step 4. Eclipse will Start

2.4 Installation of MySQL

Step 1. Download MySQL from the link [4]
Step 2. Double click to install the software

3 Configuration

3.1 How to configure tomcat Server with Eclipse

Step 1. Open eclipse.
Step 2. Click on Window next Preferences next Server.
Step 3. Click on runtime Environment.
Step 4. Click on Add next Select Apache Tomcat 7.0 from list.
Step 5. Give installation path
Step 6. Finish

[2] `https://tomcat.apache.org/download-70.cgi`
[3] `https://www.eclipse.org/downloads/packages/release/luna/sr2`
[4] `https://dev.mysql.com/downloads/mysql/5.5.html`

3.2 Create following tables in MySQL

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| nodemanagement     |
| performance_schema |
| test               |
+--------------------+
5 rows in set (0.00 sec)
```

Figure 1: Select the database

```
mysql> use nodemanagement;
Database changed
mysql> show tables;
+---------------------------+
| Tables_in_nodemanagement  |
+---------------------------+
| cloudinfo                 |
| node                      |
| process                   |
+---------------------------+
3 rows in set (0.00 sec)
```

Figure 2: Create 3 tables

```
mysql> desc process;
+------------+--------------+------+-----+---------+----------------+
| Field      | Type         | Null | Key | Default | Extra          |
+------------+--------------+------+-----+---------+----------------+
| pid        | int(11)      | NO   | PRI | NULL    | auto_increment |
| PName      | varchar(900) | YES  |     | NULL    |                |
| PMem       | varchar(900) | YES  |     | NULL    |                |
| PDesc      | varchar(900) | YES  |     | NULL    |                |
| Ptime      | varchar(900) | YES  |     | NULL    |                |
| pstatus    | varchar(90)  | YES  |     | NULL    |                |
| isexecuted | int(11)      | NO   |     | NULL    |                |
+------------+--------------+------+-----+---------+----------------+
7 rows in set (0.05 sec)
```

Figure 3: Create table process

```
mysql> desc node;
+-----------------+--------------+------+-----+---------+----------------+
| Field           | Type         | Null | Key | Default | Extra          |
+-----------------+--------------+------+-----+---------+----------------+
| nid             | int(11)      | NO   | PRI | NULL    | auto_increment |
| NodeTitle       | varchar(900) | YES  |     | NULL    |                |
| NodeDescription | varchar(900) | YES  |     | NULL    |                |
| Ram             | varchar(900) | YES  |     | NULL    |                |
| Process         | varchar(900) | YES  |     | NULL    |                |
| status          | varchar(90)  | YES  |     | NULL    |                |
| bandwidth       | varchar(90)  | YES  |     | NULL    |                |
| isalloted       | int(11)      | NO   |     | NULL    |                |
+-----------------+--------------+------+-----+---------+----------------+
8 rows in set (0.06 sec)
```

Figure 4: Create table Node

```
mysql> desc cloudinfo;
+-------------+--------------+------+-----+---------+----------------+
| Field       | Type         | Null | Key | Default | Extra          |
+-------------+--------------+------+-----+---------+----------------+
| cid         | int(11)      | NO   | PRI | NULL    | auto_increment |
| processname | varchar(900) | YES  |     | NULL    |                |
| pram        | varchar(900) | YES  |     | NULL    |                |
+-------------+--------------+------+-----+---------+----------------+
3 rows in set (0.05 sec)
```

Figure 5: Create table Cloud Info

4 Using this system

1 Open eclipse click on file next import
2 Import the project mobile cloud computing
3 Click on run
4 Open browser and enter the `http://localhost:8080/mobile_cloud_computing/`
5 Enter the username and password. Figure [6] The default username is admin and password is admin.
6 Create virtual node and processes.
7 Execute the project.

Figure 6: Amin Page