A Toolkit for Simulation of Desktop Grid Environment

by

PAYAM CHINI FOROUSHAN

17049

Dissertation submitted in partial fulfilment of
the requirements for the
Bachelor of Technology (Hons)
(Business Information System)
September 2014

Universiti Teknologi PETRONAS
Bandar Seri Iskandar
31750 Tronoh
Perak Darul Ridzuan

CERTIFICATION OF APPROVAL

A Toolkit for Simulation of Desktop Grid Environment

by

Payam Chini Foroushan
17049

A project dissertation submitted to the

Business Information Systems Programee

Universiti Teknologi PETRONAS

in partial fulfilment of the requirement for the

BACHELOR OF TECHNOLOGY (Hons)

(BUSINESS INFORMATION SYSTEMS)

Approved by, ____________________

(Dr Rohiza Binti Ahmad)

UNIVERSITI TEKNOLOGI PETRONAS

TRONOH, PERAK

Spetember 2014

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.

____

PAYAM CHINI FOROUSHAN

ACKNOWLEDGEMENTS

Firstly, I would like to thank the Almighty God for giving me the courage and wisdom to complete this paper. I would like to express my deep gratitude to my supervisor, Dr. Rohiza Binti Ahmad for the enthusiastic encouragement, for her advice and great assistance and guidance given during the completion of this project.

ABSTRACT

Peer to Peers, clusters and grids enable a combination of heterogeneous distributed recourses to resolve problems in different fields such as science, engineering and commerce. Organizations within the world wide grid environment network are offering geographically distributed resources which are administrated by schedulers and policies.

Studying the resources behavior is time consuming due to their unique behavior and uniqueness. In this type of environment it is nearly impossible to prove the effectiveness of a scheduling algorithm. Hence the main objective of this study is to develop a desktop grid simulator toolkit for measuring and modeling scheduler algorithm performance.

The selected methodology for the application development is based on prototyping methodology. The prototypes will be developed using JAVA language united with a MySQL database. Core functionality of the simulator are job generation, volunteer generation, simulating algorithms, generating graphical charts and generating reports.

A simulator for desktop grid environment has been developed using Java as the implementation language due to its wide popularity. The final system has been developed after a successful delivery of two prototypes. Despite the implementation of the mentioned core functionalities of a desktop grid simulator, advanced features such as viewing real-time graphical charts, generating PDF reports of the simulation result and exporting the final result as CSV files has been also included among the other features.

TABLE OF CONTENT

ACKNOWLEDGMENTS i
ABSTRACT ii
LIST OF FIGURES v
LIST OF TABLES vi
ABBREVIATIONS AND NOMENCLATURES vii
CHAPTER 1 INTRODUCTION 1
1.1 Background 1
1.2 Problem Statement 2
1.3 Objective 3
1.4 Methodology 4
1.5 Scope of study 5
CHAPTER 2 LITERATURE REVIEW 6
2.1 Introduction to simulation 6
2.2 Desktop grid environment 6
2.3 General Simulator for Distributed Computing 8
2.3.1 SimGrid 8
2.3.2 GridSim 11
2.3.3 GangSim 13
2.3.4 SimBOINC 17
2.3.5 SimBA 17
2.4 Reasons to Develop a New Simulation Toolkit 18
CHAPTER 3 METHODOLOGY 19
3.1 Research Methodology 19
3.2 Research Procedure 20
3.3 Development Methodology 20
3.3.1 Planning 22
3.3.2 Analysis 23
3.3.3 Design 23
3.3.4 Prototyping 24
3.3.5 Implementation & Monitoring 24

3.4 Tools required 25
3.5 Gantt Chart..26
3.6 Key Milestones 27
CHAPTER 4 RESULT AND DISCUSSION..28
4.1 Requirement Engineering 28
4.2 Prototype Design...29
4.2.1 Prototype 1.0 Activities...29
4.2.2 Problems Encountered With Prototype 1.0 31
4.2.3 Prototype 2.0 Activities...34
4.2.4 Problems Encountered With Prototype 2.0 35
4.3 Design of Final System and Graphical User Interface.....................................40
4.3.1 Job Generation...42
4.3.2 Volunteer Generation Panel 42
4.3.3 Volunteer Setting..43
4.3.4 Viewing Graphical Charts 44
4.3.5 Generate PDF Reports and Export CSV Files.....................................46
4.4 Implementation of Final System...46
CHAPTER 5 CONCLUSION...49
5.1 Conclusion 49
5.2 Future works 50
REFERENCES 52
APPENDICES 57

LIST OF FIGURES

Figure 1.1: Grid computing environment.. 2
Figure 1.2: Research methodology 5
Figure 2.1: Conceptual model of Desktop Grid environment..................................... 7
Figure 2.2: Different types of API available for SimGrid application....................... 10
Figure 2.3: GridSim architectural design 13
Figure 3.1: Prototype based methodology 21
Figure 3.2: Desktop Grid Simulator UseCase Diagram... 23
Figure 3.3: Gantt Chart for FYPI and FYPII 26
Figure 3.4: Key Milestones of FYP I and FYP II 27
Figure 4.1: Activity diagram for job generation in prototype 1.0 29
Figure 4.2: Activity diagram for host generation in prototype 1.0 30
Figure 4.3: Structure of ‘Avail_trace’ Table 31
Figure 4.4: Duration to Run the Query for Volunteer Availability............................ 32
Figure 4.5: Java code for adding partitions to “avail_trace” table 33
Figure 4.6: Duration to Run the Same Query After Committing Changes 33
Figure 4.7: Activity diagram for remaining components in prototype 2.0 34
Figure 4.8: Result of Netbean Profiler 36
Figure 4.9: Summary of Java Heap Dump... 37
Figure 4.10: Heap Dump of Classes 38
Figure 4.11: Sample of normal *try-catch* statement... 38
Figure 4.12: Sample of *try-with-resources* statement 39
Figure 4.13: Result of Final Memory Leakage Testing 40
Figure 4.14: Main Panel 41
Figure 4.15: Job Generation Panel 42
Figure 4.16: Volunteer Generation Panel... 43
Figure 4.17: Setting Panel 43
Figure 4.18: Bar Chart for Quantity of Grouping 44
Figure 4.19: Pie Chart of Successful and Unsuccessful Job Submission 45
Figure 4.20: Linear Chart for Quantity of Engaged Volunteers 46
Figure 4.21: GUI issue Detected During Final Meeting 47
Figure 4.22: Official Website of University Technology Petronas’ HPC center....... 48

LIST OF TABLES

Table 2.1: Comparison of GangSim,GridSim and SimGrid simulator 16
Table 3.1: Descrption of Tools Required Desktop Grid Simulator 25
Table 4.1: User and system requirements 28
Table 4.2: Summary of Released Prototypes and Issues.. 40

ABBREVIATIONS AND NOMENCLATURES

API	Application Programming Interface
SMP	Symmetric Multiprocessing
JVM	Java Virtual Machine
GVP	Guarantee of Victorious Probability
VO	Virtual Organization
LS	Local Schedulers
ES	External Schedulers
DS	Data Schedulers
PEPs	Policy Enforcement Points
S-PEPs	Site Policy Enforcement
V-PEPs	Virtual Organization Policy Enforcement
BOINC	Berkeley Open Infrastructure for Network Computing
SimBa	Simulator of BOINC Applications
UTP	Universiti Teknologi Petronas
HPC	High Performance Computing
PDF	Portable Document Format
GUI	Graphical User Interface
HCI	Human-Computer Interaction
CSV	Comma Separated Variable
GC	Garbage Collector
Mb	Megabyte
SQL	Search and Query Language
CPU	Central Processing Unit

CHAPTER 1

INTRODUCTION

1.1 Background

A computational grid has been described as “A hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities.” [1].

As explained by Foster and Kesselman [1], Grid computing is a concept which inherits the grid’s environment characteristics such as volatility, distribution and share-ability. This infrastructure and technology is meant to connect heterogeneous and geographically distributed computers across the network. Figure 1.1 demonstrates the world wide infrastructure of grid computing. Referring to Figure 1.1, it is shown that different types of resources could be shared and connected on the grid environment such as databases (R3), servers (R4), personal computers and clusters. In grid computing it is required for resources to follow the “push based” model in order to execute tasks. The jobs need to be pushed from client to the resources in order to execute jobs [2].

Desktop grid computing is one type of distributed computing which uses the idle computers around the world for computing power, this type of computing is following a “pull based” behavior, meaning that the idle resources will push the jobs from servers to the clients (a vise-versa concept compared to grid computing) in order to execute them. The concept of desktop grid is that the resources of idle computers throughout the network would be detected and used [3]. An assemblage of idle computer resources on the network, which are normally unused, will be used for a low cost unit computing which is able to perform high-performance tasks [4].

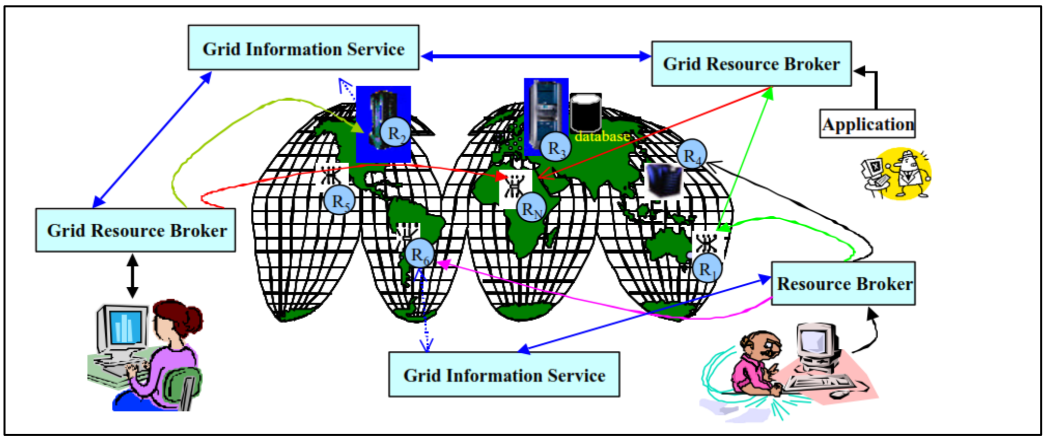


Figure 1.1: Grid computing environment

Foster and Iamnitchi [3] organized the desktop grid computing into two different categories as, enterprise computing and volunteer computing. In the enterprise desktop grid computing model, the idle computer resources from one or more organization would be used through a high-speed local-area network. On the other hand, in the volunteer computing model, idle random geographically distributed recourses over the internet world wide would be used.

In recent years, there has been a significant growth of interest in desktop grid computing mainly due to the successful outcome of most desktop grid related projects [5]. Various new algorithm and policies are introduced by researchers in order to increase the performance of this environment, this new algorithms will be tested in Desktop Grid Simulators in order to verify its efficiency [6].

1.2 Problem Statement

As similar with other systems, simulation of the algorithms used in the scheduler is one of the most important steps before the actual usage of the algorithm in the real world. This is to ensure the efficiency of the job scheduling algorithm prior to its implementation. However, for desktop grid environment, the challenge is quite different from other distributed environments.

The main challenge of desktop grid environment is the unrepeatable nature of the volatile resources since no one has the capability to fully control all available resources and tasks [7]. Desktop grid computing environment consists of a large number computing nodes (volunteer) with highly dynamic behavior (the node may be come up or down in random moments of time) therefore reliable models for planning of the whole computing (load balancing, algorithms or fault recovery strategies) are highly desired. In order to use the strengths and contain the weakness of desktop grid environments for different projects, project designers need to study the efficiency of different project parameters and scheduling policies without affecting the desktop grid community.

Several challenges arise in doing this in desktop grid environments:

- The time required to measure project throughput, total number of results delivered to the scientists in a given amount of time, under different parameter settings can be significant.
- Problems due to testing might upset volunteers, even to the degree that they leave a project [8].
- Limited "desktop grid" simulators are available. The complexity of simulators are relatively high, to an extend which impossible for normal users without programming knowledge to fully understand the simulator toolkit.

1.3 Objective

In order to solve the problems or challenges mentioned in this study, it is necessary to conduct "performance" studies in a simulated environment. Within a simulated enviornment it is possible to test a wide range of hypotheses in a short period of time without affecting the desktop grid community. Hence, the objective for this study can be broken into:

- To study the requirements for desktop grid environment simulator
- To design the desktop grid environment simulation toolkit
- To test the simulator for evaluating algorithms of schedulers with relatively easy complexity usable for general users.

1.4 Methodology

To achieve the objectives following research methodology has been conducted:

- Conducting literature review related to grid environment simulation
- Collect information related to desktop grid environments from experts in order to analyze the desktop grid environment
- Constructing the framework for simulation of desktop grid environment
- Implementation of the constructed framework into a simulation toolkit to test the simulation process

Figure 1.2 demonstrates the four step research methodology process used in this study.

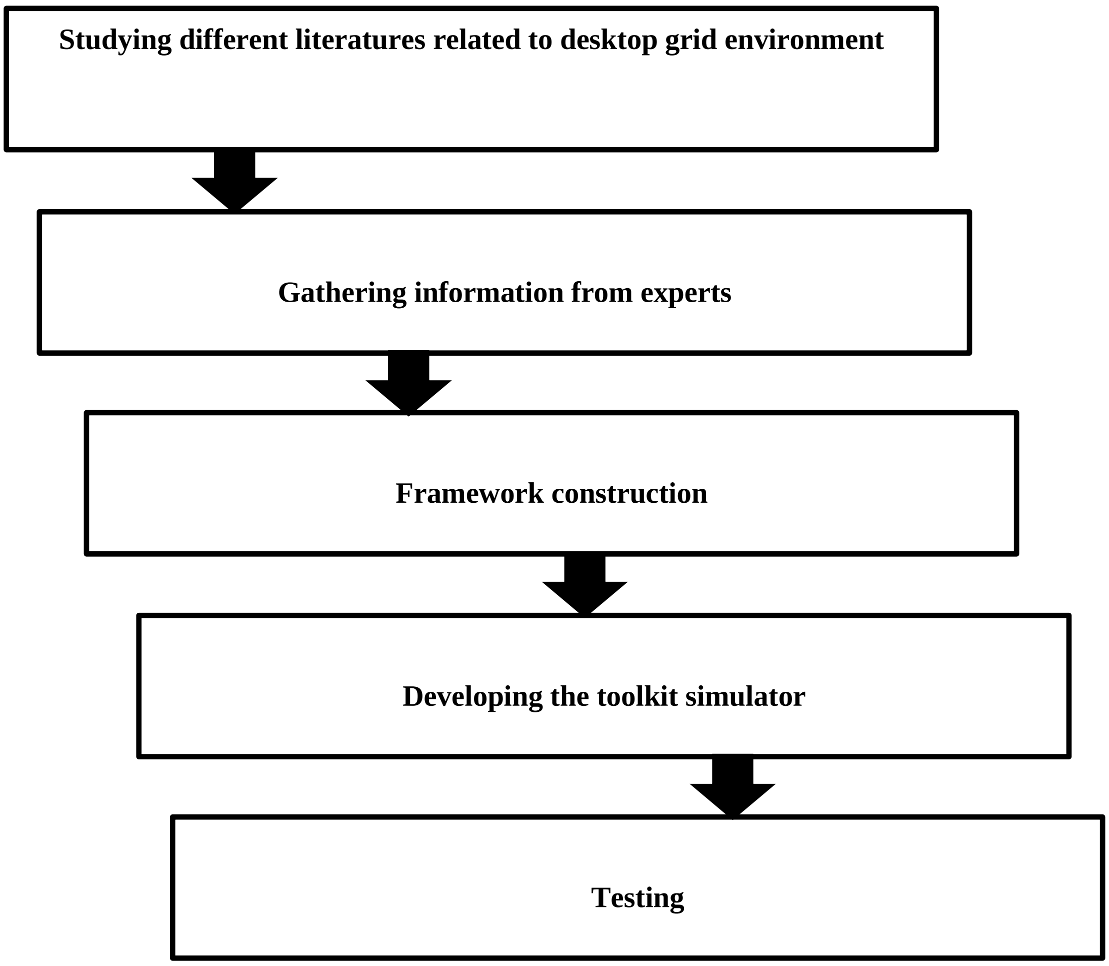

Studying different literatures related to desktop grid environment
Gathering information from experts
Framework construction
Developing the toolkit simulator
Testing

Figure 1.2: Research methodology

1.5 Scope of study

This study is limited to the development of a simulator toolkit for desktop grid environment. The scope will be focusing on the current simulator toolkits. The scope is mainly focused on analysis of the simulator system and constructing the simulator's framework. Eventually, the simulator will be developed.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction to simulation

Simulation can be a very influential tool if it is used appropriately within an academically correct context. As defined by Shannon [9] simulation is “The process of designing a model of a real system and conducting experiments with this model for purpose either of understanding the behavior of the system or of evaluation various strategies (within the limits imposed by a criterion or set of criteria) for the operation of the system”.

According to this definition, discrete event is a type of simulation which could be followed during architecture design. A massive advantage of discrete-event approach is the ability to follow the real system dynamics [10]. On the other hand various other types of simulation models, such as high-powered optimization models are unable to mimic the variable and dynamics of a system. This mimic ability of discrete-event model gives the system a unique way to structure, function and analyze result.

2.2 Desktop grid environment

A desktop grid computing environment contains nodes (volunteer), brokers and schedulers [11]. The main target of this environment is to complete huge bulk of jobs which is submitted through the server. Figure 2.1 demonstrates a conceptual model of Desktop Grid environment.

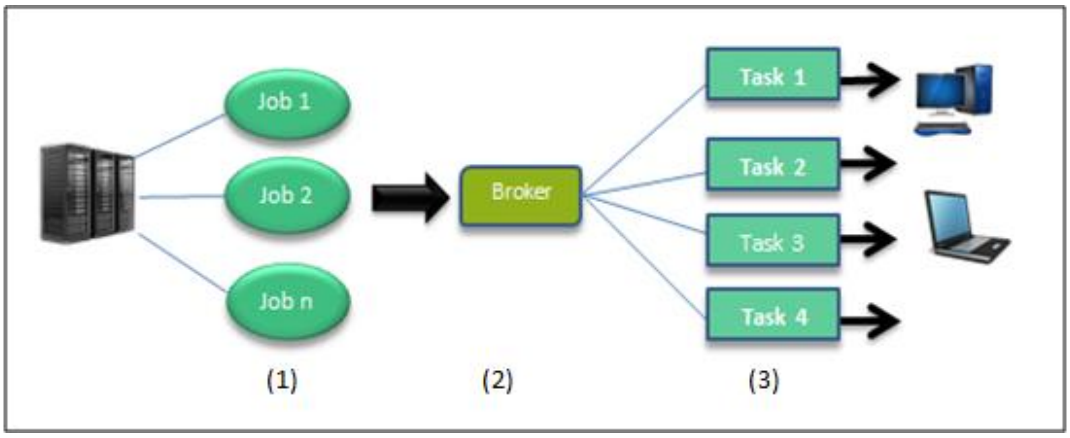


Figure 2.1: Conceptual model of Desktop Grid environment

According to above figure, the process of completing jobs within this environment can be described as following:

1. Job gets assigned to the broker
2. The broker breaks the job into smaller independent portions (referred as sub-jobs or tasks) according to a pre-defined policy
3. Tasks will be assigned to the volunteers and executed
4. Executed tasks will be send back to the server

In the last decade, studies on desktop grid computing have shown that there is the an opportunity use to the idle CPU cycle within a network of million computers distributed worldwide [12]. The fact that only one tenth of the entire processing power of idle computer resources is being used, confirms that the resources in both personal computers and organizational computers are underutilized [13]. Hence, the interest has been elevated on using the available ide resource over the Internet. This new technology has been referred to as Internet computing, other alternative names given are peer to peer (P2P) computing [14], distributed computing and enterprise/desktop grid computing [15].

Grid desktop computing could be helpful in resolving large-scale issues, as an instance SETI@home is the first project which attempts to use distributed computing as a tool to analyze the radio signals from space using millions of computers worldwide [16].

Anderson, et al. [17] claimed that SETI@home project performed 221 million of work units with an average throughput of 27.36 TFLOPS during a 12 month period which is a significant result and also demonstrates the power of Desktop grid computing. Anderson, et al. [17] further argues there are enough available computers connected through the internet to support 100 projects as big as SETI@home, this indicates the vast potential of expansion of this field.

2.3 General Simulator for Distributed Computing

According to Donassolo, et al. [7] grid computing and volunteer computing platforms include thousands of volunteers which have heterogeneous specifics and volatile behavior. Due to this two specifications, experimenting with real-world platforms of volunteer computing is quite challenging and hence most of the researches will be based on simulation [18]. There are various different types of simulation method and modeling depending on the discipline and area which the research is focused. It is difficult to re-use simulation models from different disciplines due to the lack of mutual criteria's in each discipline [10]. There are numerous simulators developed and proposed for grid computing, although only few of them specified for simulation of the desktop grid environment [19].

The main purpose of following sections is to review literature related to general simulators of volunteer computing and grid environment. The outcome of the literature shall identify the minimum requirements of a complete desktop grid simulation toolkit.

2.3.1 SimGrid

SimGrid Is the first simulator developed in 1998 and is still widely been used up till today. The main core functionality of SimGrid is to simulate distributed application in heterogeneous and distributed environment [19].

SimGrid provides a set of core functionalities that enables the user to simulate domains of application and computing environment topologies. This simulator uses an event-based approach for its simulation. The main and most important section in the simulation process is to simulate the resources (also referred as volunteers). The assumption of this simulator is that each individual resource has two performance characteristics as: latency (the amount of seconds required to access a resource) and service rate (quantity of work units completed in one second). Two mechanisms are provided by SimGrid for simulating the mentioned performance characteristics, first mechanisms is based on the actual traces driven form the real platform and second is a set of constant values for each resource. Real world traces are able to generate via different volunteer computing monitoring tools. Unfortunately, using only traces is not sufficient to simulate every behavior of the resources due to many hidden characteristics of each individual resource; however, it will be a first step and guidance for current simulators which could be improved over time [20].

Correct implementation of a simulator is highly crucial for good performance and accurate result, the implementation of Simgrid and its test suit consist of approximately 10,000 lines of C code. The main target of the developers were to implement techniques to improve the speed and memory usage, as an instance, traces in SimGrid will only be loaded only when needed and un-used traces will be distracted. These techniques are crucial to be used in due to the large quantity of resources throughout the simulation process.

Another technique introduced by Casanova [20] to improve the CPU performance and efficacy during simulation is the use of a multi-level trace model where each additional level is using a coarser time scale rather than traditional singular-level trace model. Experiments on a Pentium II 360MHZ indicated that with the use of a multi-level trace model the ratio between Simgrid CPU time and virtual time simulated will decrease from 10^{-6} to 10^{-10} which is a significant improvement. However, maintaining a multi-level trace requires an extra memory cost due to its complexity but it the cost is well worth considering the performance improvement.

The first application of SimGrid is PSTSim, a simulator mainly targeted to simulate and evaluate scheduling strategies for parameter sweep application. The second application or simulator based on SimGrid is DagSim which is focused on simulating scheduling algorithms for applications which are based on DAG structure. These two simulators are completely different in the means of purpose, structure, architecture and design. However, SimGrid's flexibility was amendable within both of the simulators.

Casanova [20] argues that, although it is possible to generate accurate results using SimGrid toolkit, however, it is still not user friendly enough and a user with limited programming skills and knowledge might face difficulties using this toolkit. The main strength of SimGrid, which is not available on other simulators, is the ability to simulate computation and background traffic of the recourses. The models are constructed using C files and XML which might be challenging for non-expert users to understand. According to Donassolo, et al. [7] versions of SimGrid has been released on a regular basis, in the current version there are four APIs provided (MSG, GRAS, SimDAG, and SMPI) each of these APIs are provided for a particular use. Figure 2.1 shows different types of APIs available for current release.

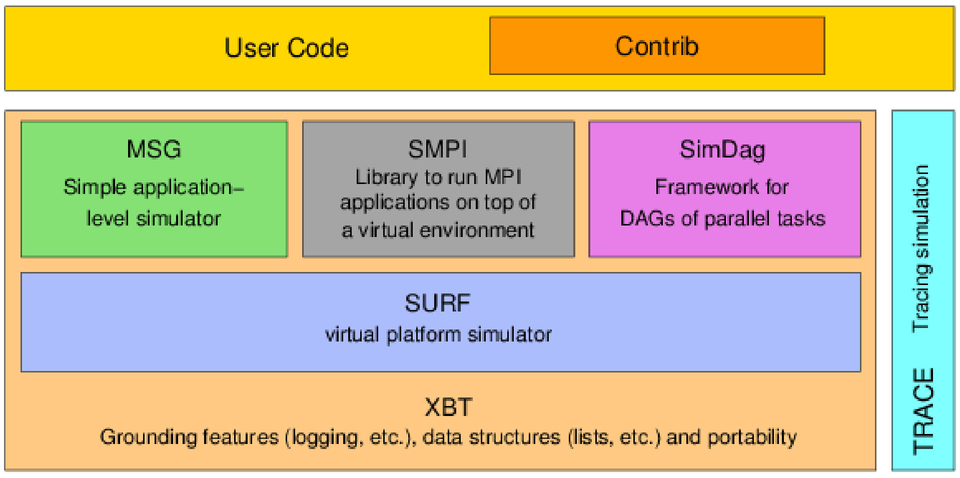

User Code
Contrib
MSG
Simple application-level simulator
SMPI
Library to run MPI applications on top of a virtual environment
SimDag
Framework for DAGs of parallel tasks
SURF
virtual platform simulator
XBT
Grounding features (logging, etc.), data structures (lists, etc.) and portability
TRACE
Tracing simulation

Figure 2.2: Different types of API available for SimGrid application.

In a study on redistribution a SimGrid application was used in order to generate and simulate a two-step algorithm for parallel mixed structured application [21]. Wen, et al. [22] proposed a new algorithm for grid scheduling, the algorithm was experimented and verified by a simulation using SimGrid toolkit. In another study based on distributed environment, SimGrid was the tool used in order to simulate task reallocation in different clusters [23].

2.3.2 GridSim

Another widely used grid simulator which has been updated to version 5.0 recently is the GridSim simulator. According to Buyya and Murshed [24], Gridsim is a java-based district-event toolkit mainly developed for simulating heterogeneous resources as well as models and users. The main advantage of this highly flexible toolkit is the ability to simulate various different types of classes including machines and schedulers. The GridSim toolkit supports simulation and modeling of PCs, SMPs, workstations, and distributed different configured clusters.

Buyya and Murshed [24] further described the grid computing environment as a peer-to-peer computing network. An interesting fact in this paper is the emphasis on the broker’s role within the grid computing architecture. For proving the scalability and reliability of SimGrid, Buyya and Murshed [24] developed a grid recourse broker, referred as Nimrod-G which performed scheduling of task-framing application on geographically distributed resources. It supports budget scheduling based on market-based economic models.

Different classes of heterogeneous users, resources and brokers could be simulated using GridSim toolkit. Brokers, also referred as schedulers, have the responsibility to select the resources for each user and deliver the tasks to the aggregated for each individual user. Hence each user has its own private broker which is responsible to submit jobs to the central scheduler. This enables the central scheduler is targeted for a global optimization such as higher performance and system utilization of different resources based on their broker policy.

Main features of Gridsim include:

- Ability to simulate dynamic and static schedulers
- It is possible to locate the resources at any time zone
- Tasks of the application could be heterogeneous
- No limits are set for job submission to resources

GridSim’s architecture is a multilayered architecture as demonstrated in Figure 2.2; its first layer is related to the java’s runtime environment also referred as JVM (JAVA virtual Machine). The second layer is mainly for using the interfaces created at the first layer and build an elementary discrete-event infrastructure; Simjava is among the most popular discrete-event infrastructure in JAVA which is also been used in SimGrid’s implementation. The third layer is concerned with simulating and modeling of fundamental Grid entities for instance, information services, resources and etc. The fourth layer is for modeling the Grid brokers and schedulers, the fifth and final layer focusing on simulating and modeling application and recourse based on different scenarios.

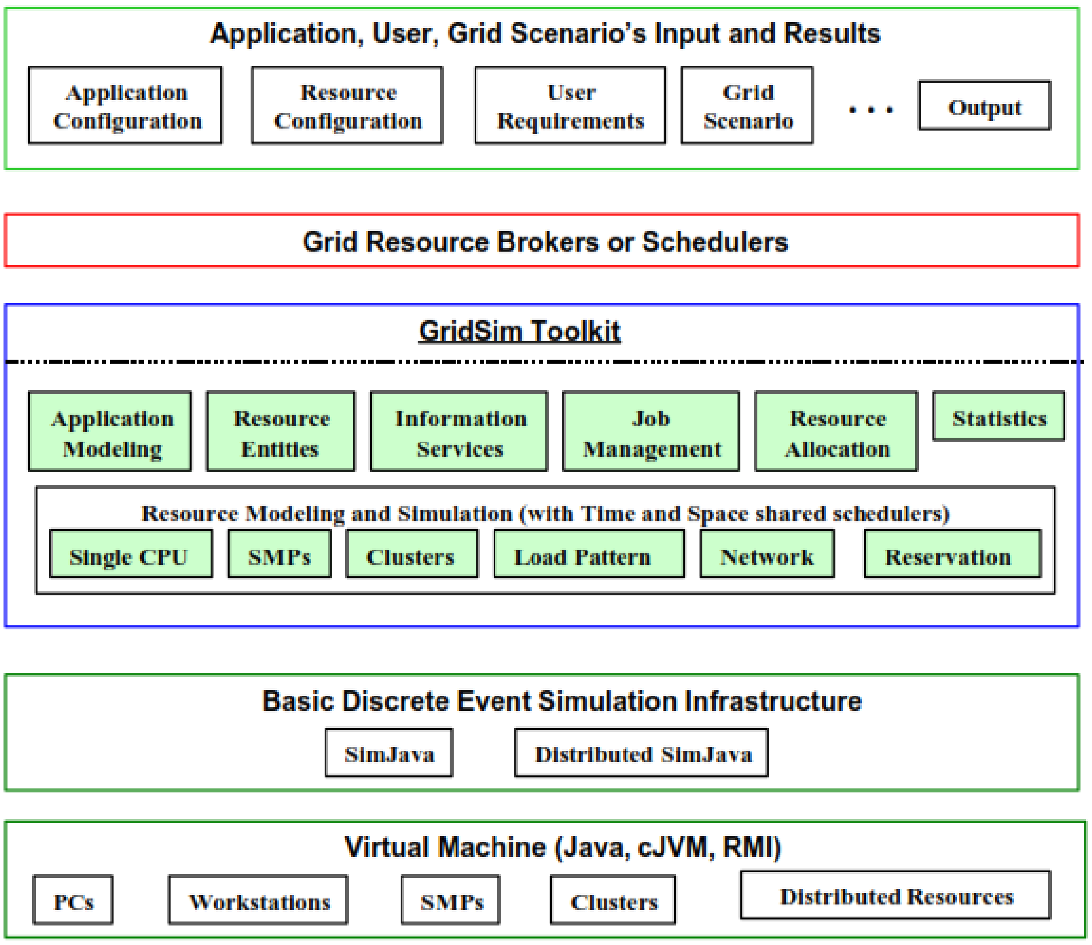


Figure 2.3: GridSim architectural design

Yu and Buyya [25] used GridSim simulator in order to simulate a grid environment which enables them to test their proposed genetic algorithm in order to schedule scientific workflow application. In grid environment idle resources are constantly competing for getting a task. Yao, et al. [26] introduced a Guarantee of Victorious Probability (GVP) algorithm to increase the efficiency of task submissions to the resources. Another research used the GridSim simulator simply to create a grid environment in order to experiment on statistical perspective of job scheduling and job allocation [27].

2.3.3 GangSim

GangSim is another tool used for grid desktop simulation. It was developed to support simulation of schedulers in grid environment with a particular emphasis on examination of interaction between community and local resource allocation algorithm [28]. One of the main advantages of GangSim is the ability to simulate virtual organization users and planers beside the normal ability to model site policies. Another advantage is that GangSim enables to run and combine different simulation components on parallel basis [29]. However, the basic assumption of GangSim is that each job runs for minimum of 100 seconds which might be too long for some models, specially service related models, Hence it is not possible to simulate all type of models using GangSim [30].

GangSim is able to model following processes: job submission process, resource monitoring and usage of policy infrastructure. Dumitrescu and Foster [28] listed 7 different components included in GangSim toolkit.

- Site: characterized by its disk space, size of CPU and memory, and networks. Each of the characteristics is loaded and stored into a configuration file during the startup.
- Virtual Organization (VO): composes from a set of users who are submitting jobs, which might be grouped into workloads. Each job has its own requirements to get completely executed; these requirements will be distributed in files among the sites in the beginning.
- Local Schedulers, External Schedulers and Data Schedulers (LS, ES, DS): These schedulers are mainly representing points where scheduling decision is performed.
- Monitor Data Points (MDPs): Represent the monitoring component of the simulator also referred as nodes which consist of information gathered from internal and external schedulers.
- Policy Enforcement Points (PEPs): This component is responsible for policy enforcement based on the gathered information from related operation and this information would be used to set resources allocation [31, 32].
- Site Policy Enforcement (S-PEPs): Is mainly responsible for validation of the entire process. As an instance, jobs are immediately removed if they are not following the policy requirements.

- Virtual Organization Policy Enforcement (V-PEPs): Have the same context as S-PEPs but mainly are focused on virtual organizations rather than sites.

GridSim simulation toolkit has not been widely used in studies due to in-popularity and limitations. However, numerous reliable sources acknowledge GridSim as toolkit for grid environment simulations [28, 29, 33, 34].

Table 2.1 indicates a comparison between SimGrid, GridSim and GangSim toolkit, a comparison between different types of simulators could eventually lead us to find the mutual components in each simulator. These mutual components are fundamental of simulation application which is crucial to be implemented into the grid desktop simulator toolkit.

Table 2.1: Comparison of GangSim,GridSim and SimGrid simulator

Name	Advantage	Disadvantage	Key Specialty
SimGrid	Follows a multi-level trace model which increases the performance of the application.	Too complex structure due to various types of API's.	simulate and evaluate scheduling strategies for parameter sweep application
GridSim	Since each user has its own broker, central scheduler is targeted for a global optimization such as higher performance and system utilization of different resources based on their broker policy	With an increase in user population, the quantity of brokers will increase which decreases the memory space and CPU performance.	simulating heterogeneous resources as well as models and users
GangSim	Ability to simulate virtual organization users and planers beside the normal ability to model site policies.	Unsuitable to simulate every model since the basic assumption is that each job runs for a minimum 100 seconds which might be too long for some particular models.	Simulation of schedulers in grid environment with a particular emphasis on examination of interaction between community and local resource allocation algorithm

2.3.4 SimBOINC

SimBOINC [35] is a simulator for desktop grids systems. Berkeley Open Infrastructure for Network Computing (BOINC) is a well-known framework used in volunteer computing projects [36], SimBOINC is a simulator in order to test and evaluate new scheduling policies in BOINC and other desktop grid systems. SimBOINC’s infrastructure is based on the SimGrid and its main target is to simulate distributed and parallel computing systems [35].

2.3.5 SimBA

SimBA (Simulator of BOINC Applications) [8] is a discrete event simulator which models the main functions of BOINC. SimBA is able to model main components of desktop grid environment such as job generation, workload generation and tasks distribution. The entire simulation process in SimBA will be performed in a virtual environment which is volatile, heterogeneous, and distributed as how it is in the real grid environment. SimBA simulates the creation, characterization, and termination of volunteers by using trace files obtained from real BOINC projects.

A trace file contains information such as generate time of jobs, type of operating system and the availability duration of volunteers. Fundamental components of SimBA are to generate work-units, to create a number of instances for each work-unit or replicas based on the project replication policy; to dispatch instances work unit to volunteers according to scheduling algorithm policy; to model behavior of volunteer availability and unavailability, heterogeneity of volunteers by characterization which is obtained from the real trace file; to determine the status of a returned results from volunteers using the characterization of volunteer’s unsuccessful rate that are obtained from the trace file; To determine the legitimacy of completed results by the project’s confirmation policy; and finally the component of performance computation in terms of throughput [8, 37, 38].

2.4 Reasons to Develop a New Simulation Toolkit

Some of the main components of desktop grid environment such as availability of volunteers and reliability of the volunteers are simulated through statistical formulas and distributions, which is a major disadvantage of current simulators. These formulas and distributions are reliable for simulations yet not as accurate as the real grid environment. Hence, developing a simulator which can use a set of ‘trace data’ to model the availability and reliability of volunteers will significantly increase the reliability of the simulation process [39, 40].

This simulator will be developed according to the collected trace by the expert in our second level of methodology (Referring to Figure 1.2 in Section 1.4). The trace is gained from real trace data set of SETI@home project for a period of 10 months retrieved from http://setiathome.berkeley.edu/index.php.

CHAPTER 3

METHODOLOGY

3.1 Research Methodology

In order to have a better understanding of the simulation processes and simulator's component, a qualitative research has been conducted. Desktop grid computing is precisely a newly introduced concept in computer science and limited researches have been conducted related to this field, hence conducting a quantitative research related to desktop grid computing is quiet difficult and nearly impossible.

Firstly, it is important to understand the behavior of volunteers and jobs in reality which will provide the researcher a full understanding of the simulation process and the requirements. In order to fulfill this step, a study on a 10-month real life data set generated from SETI@home has been conducted. The main analysis on the data was conducted by High Performance Computing (HPC) Service Center located at University Technology PETRONAS Malaysia campus.

Secondly, a conceptual model is required in order to ease the process of development of the simulator toolkit. This conceptual model integrates a desktop grid simulator which is able to simulate volunteer behavior in a huge time frame (eg. 10 months) in a relatively shorter time frame without interrupting their process in the server and storing the results into the workstation in user friendly format (such as PDF) so that it could be used for future references.

3.2 Research Procedure

There are five phases involved in conducting this research, in the first phase the author identifies the process and activities involved in developing a desktop grid simulator toolkit, this is done through considering and reviewing various literatures related to desktop grid environments simulator.

The second phase is to collect information related to desktop grid prototype from experts in order to analyze the desktop grid environment; this is done by analyzing the result of the semi-structured interviews taken from UTP's HPC staffs that have some expertise with desktop grid computing. This information from the semi-structured interviews could also be helpful to identify functions of the desktop simulator toolkit.

The third phase is constructing the framework of desktop grid environment, which is then followed by the fourth step. The methodology will be applied here, whereby the prototype will be developed. In the fifth phase developed prototypes should be evaluated and tested by experts on a regular basis. The feedbacks should be documented properly and implemented in the next prototype version. Each prototype of the desktop grid simulator toolkit would be tested and evaluated by HPC staff on monthly basis for verification purposes. It is also necessary to conduct different types of testing to ensure functionality, reliability, durability and flexibility of the finished simulator toolkit. The analysis of the feedbacks should be reviewed before the final user acceptance testing.

3.3 Development Methodology

The main development strategy for the grid desktop simulator used is prototyping strategy. According to Sommerville [41], prototypes are initial versions of an application which demonstrate the core functionalities; the main purpose of a prototype is to give more information about the main problem.

This development strategy will repeat the 3 main phases of analysis; design and implementation concurrently until the final release get accepted and verified [42]. This enables the users to test and evaluate prototypes before the finial application release which is an advantage.

Prototyping approach includes 5 main phases as following:

1. Planning
2. Analysis
3. Design
4. Develop prototype
5. Testing & maintenance
6. Release the final system

Each phase needs to follow the sequence demonstrated in figure 3.1.

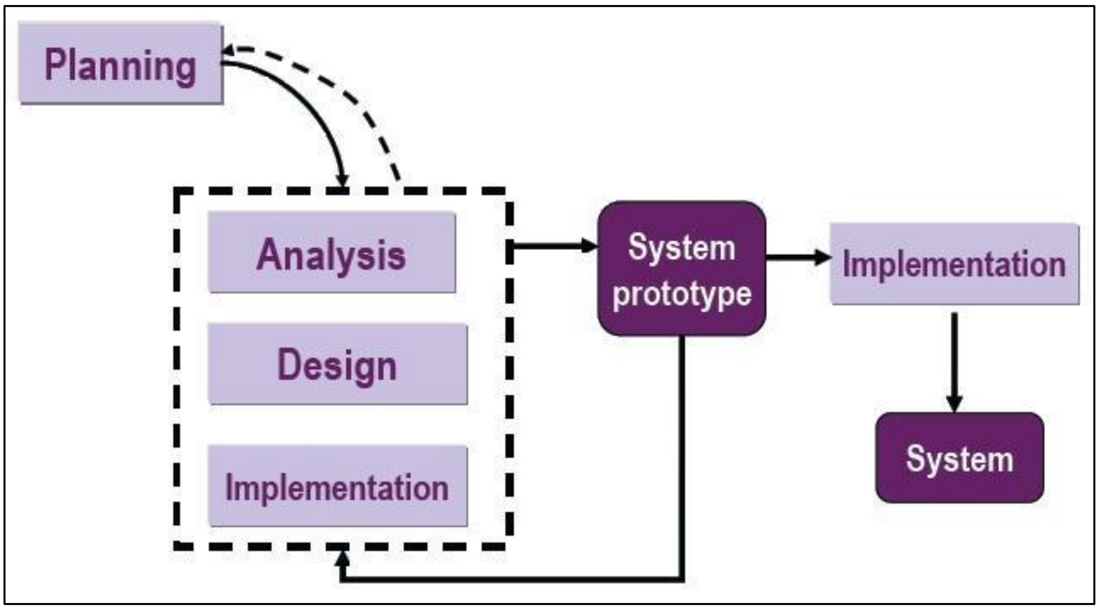


Figure 3.1: Prototype based methodology

3.3.1 Planning

Planning is one of the most important steps in project development since within this phase the scope, requirement, specification and objectives should be clarified before moving to further phases. An understanding of Software Development Lifecycle is crucial to ease the planning process. It is possible to break down this phase into three main categories.

I) Studying and understanding different literature: Detail analysis and deep understanding could be gained from reading related literatures written by researchers related to the scope of topic. The literature review (Referring Chapter 2) is mainly focused on breaking down and evaluating other simulators related to volunteer computing. This enables the developer to find and gather all the mutual factors within different simulators, these mutual factors are crucial to exist in the desktop grid simulator toolkit hence all of them should be included in the planning.

II) Evaluation of similar simulators: This step can be helpful to get ideas on designing the GUI. User interface is one of the most important components within each application which should follow HCI rules and standards. Testing and evaluating similar and related applications to desktop grid computing, such as gridsim and simgrid could be helpful for determining a suitable GUI design.

III) Information gathering from experts: This is conducted through semi-structured interviews with UTP's HPC staff that are expertise in volunteer computing field. It is recommended to discuss and evaluate the entire gathered project requirement with the experts in order to confirm and validate and then only compile the project plan based on the validated requirements.

3.3.2 Analysis

In this phase, it is required to analyze all the gathered information from the resources and describe important facts and assumptions. The main activity in this stage is to research more on proposed system such as the feasibility of the system and time constraints. Within this phase the technical requirements will be finalized to get used in later stages.

3.3.3 Design

In this phase development of the basic architecture design will be conducted which will be based on the analyzed information gathered in previous phases. The design should be explained in detail and documented for future references. Following is the conceptual model of the desktop grid simulator toolkit:

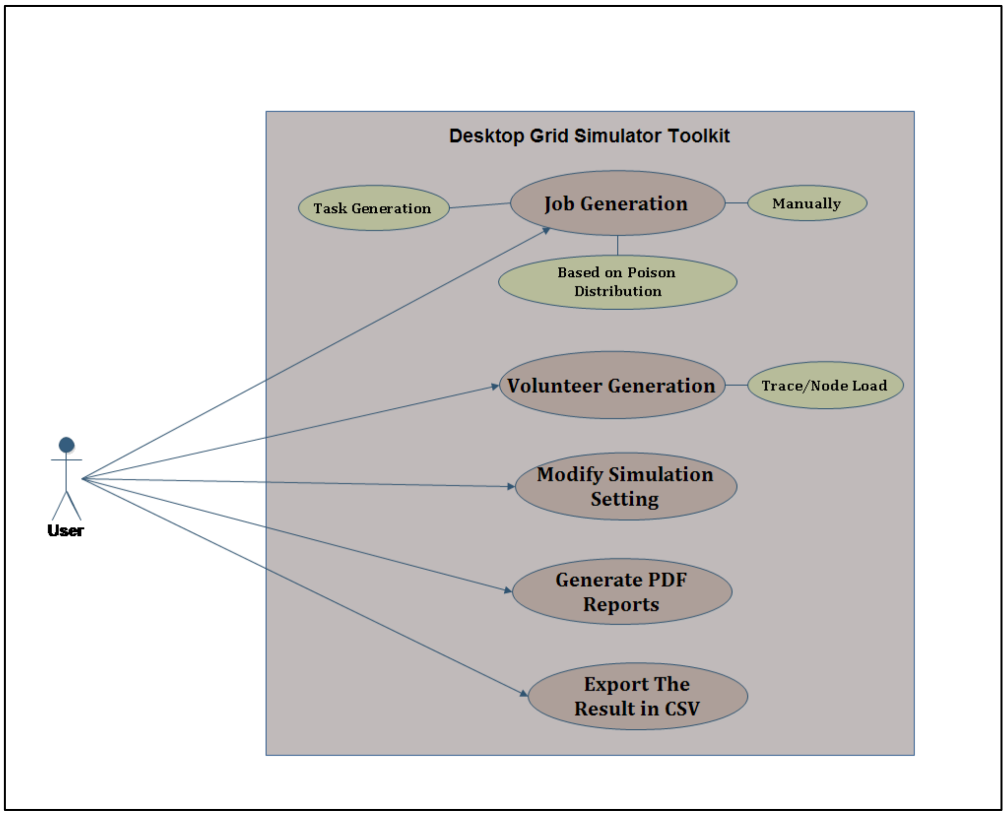


Figure 3.2: Desktop Grid Simulator UseCase Diagram

3.3.4 Prototyping

In this step an initial prototype should be developed based on the architecture design in section 3.3.3. The flow of the processes should follow a sequence; firstly the prototype is required to generate jobs and tasks. Job arrival in a desktop grid environment should be simulated according to a passion process or it should be possible to manually change the arrival rate [39]. Second the user is required to generate volunteer's (resources), with the volunteer generation the respective notes of the volunteer should be also loaded into the system. Third, user should be able to change the simulation setting (Time zone, RAM distribution etc.). The second step is to implement more advanced features of the simulator such as generating PDF reports, implementing graphical analysis and exporting result into CSV file in later releases of the prototype.

3.3.5 Implementation & Monitoring

Implementation mainly focuses on programming and testing. In each developed prototype, all the functions will be tested and evaluated. Once all the function perform as required by the requirements, the entire application components will be integrated and a final version gets compiled. This final version would be given to the target users to test including feedback forms. All the gathered information from the feedbacks would be compared back to the planning, analysis and design process if there is any mismatch or satisfaction the process gets repeated till the user has a complete satisfaction.

3.4 Tools required

Table 3.1: Descrption of Tools Required Desktop Grid Simulator

Purpose	Application Name	**Version**	Release Date/ Update
System Development Tool	NetBeans	8.0.1	9-Sep-2014
Database query and administration	Aqua Data Studio	13.0.4-5	18-Oct-2013
Database design	MySQL Workbench	6.2	23-Sep-2014
Data migration from MS excel to database	MySQL for excel	6.2	23-Sep-2014
Drawing diagrams	Edraw Max	6.0	8-Aug-20133
Drawing diagrams	Gliffy	Online based	2013
Generating Reports	Jaspersoft iReport Designer	5.6.0	10-Feb-2014

3.5 Gantt Chart

Date Activities	Duration (weeks)	1	2	3	4	5	6	7	8	9	10	11	12	13	14			1	2	3	4	5	6	7	8	9	10	11	12	13	14
SEMESTER		**FYPI**																**FYPII**													
PLANNING																															
1- Identify Problem and Objectives	2	▨	▨																												
2- Feasilibilty Analysis	2		▨	▨																											
ANALYSIS																															
1- Review Litritures	1				▨																										
2- Interview Experts	2				▨	▨																									
3-Methodology Design	2						▨	▨																							
DESIGN																															
1- Architecture Design	2								▨	▨																					
2- Database Design	3									▨	▨	▨	▨	▨	▨																
3- GUI Design	3									▨	▨	▨	▨	▨	▨																
4- Interim Report	1											▨																			
5- Developing Fonctionalities	5																	▨	▨	▨	▨	▨									
TESTING																															
1- Unit Testing	5																	▨	▨	▨	▨	▨									
2- Beta Testing	3																						▨	▨	▨						
3- User Acceptance Testing (UAC)	1																								▨						
IMPLEMENTATION																															
1- Implementing the System in HPC cente	2																								▨	▨					
2- Dissertation	2																										▨	▨			
3- VIVA	3																												▨	▨	▨

Figure 3.3: Gantt Chart for FYPI and FYPII

3.6 Key Milestones

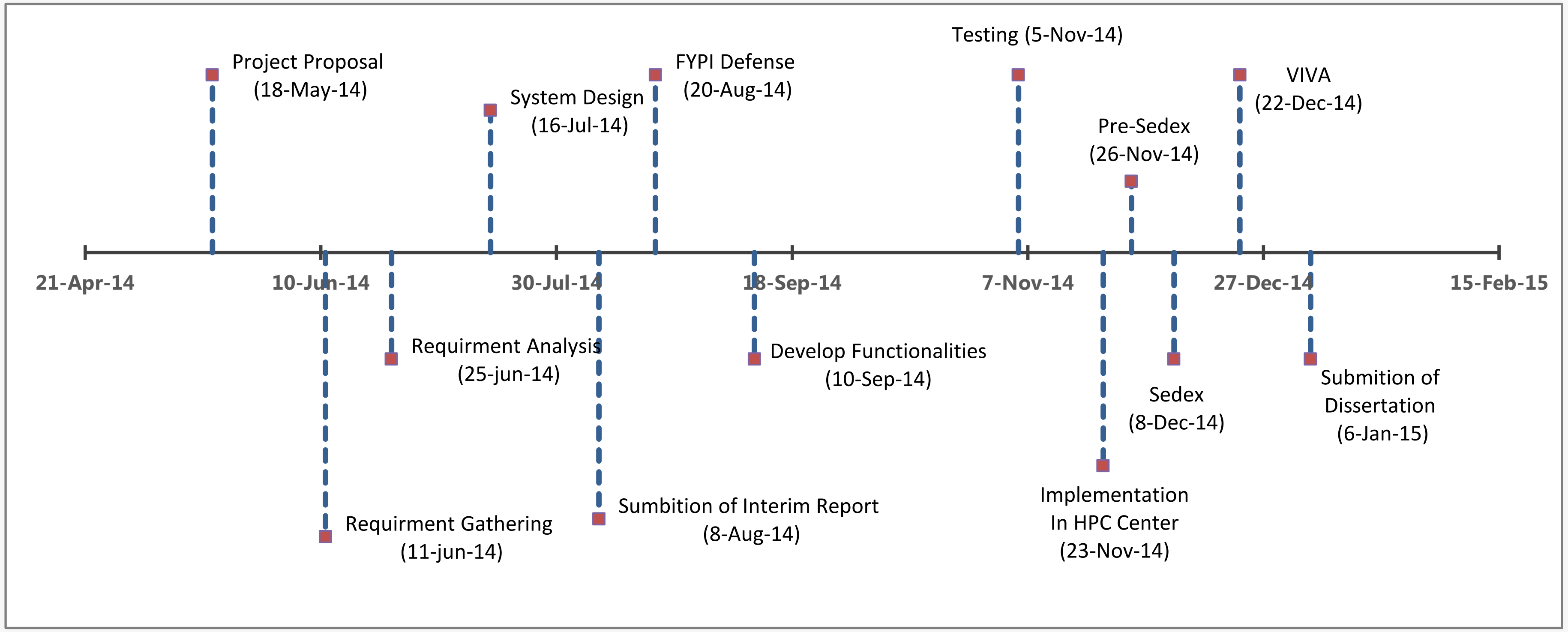


Figure 3.4: Key Milestones of FYP I and FYP II

CHAPTER 4

RESULT AND DISCUSSION

4.1 Requirement Engineering

The requirements of a system displays the purpose of a system, it is also concerned to identify different functionalities which the system is required to perform. This information is necessary for completing the analysis phase. User requirement is the main service which software is supposed to perform according to the outcome of semi-structured interviews taken. System requirement is the detailed description of software's functionalities and services which is the based on literatures reviewed. It is possible to set the most important sections of a desktop grid environment as the initial requirement. Table 4.1 shows both user requirement and system specification for the simulation toolkit.

Table 4.1: User and system requirements

User Requirement	1 Simulate and evaluate scheduling policies
System Requirement	1.1 User shall be able to generate workloads, jobs and tasks 1.2 User shall be able to generate volunteers 1.3 User shall be able to change default volunteer generation Setting 1.4 User shall be able to view graphical charts of the result 1.5 User shall be able to create PDF reports of the result 1.6 User shall be able to extract the entire simulation details as a CSV file on their local system

4.2 Prototype Design

In order to deliver a fully functional desktop grid simulator toolkit, prototypes shall be released to be tested and verified by the experts. After the verification test the final system might be released and implemented in HPC center. Through a total period of 8 months two prototypes have been released.

4.2.1 Prototype 1.0 Activities

The first step to develop prototype 1.0 is by evaluating each functionality and activity involved in the application. This is possible through transferring all the gathered information to a conceptual design. Figure 4.1 is the activity diagram for job generation in prototype 1.0. In the first step the user should be able to access the system; second step is that the user should open the job generation tab implemented in the main page (or press CTR + J). Third, the user should be able modify parameters required for job generation and finally in the fourth step the process should successfully be finished.

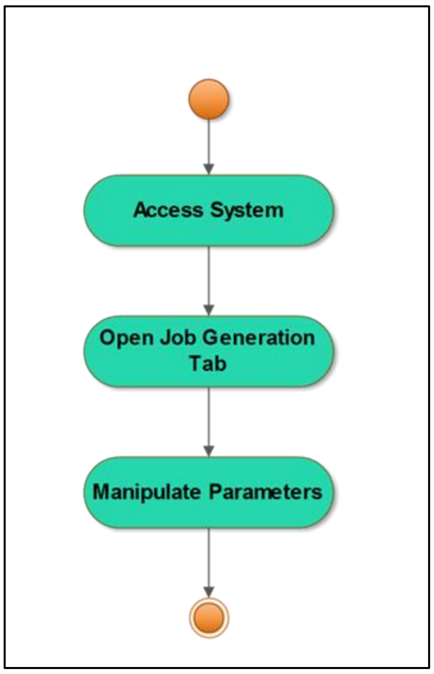


Figure 4.1: Activity diagram for job generation in prototype 1.0

Figure 4.2 indicates the process of generating available volunteers, First the user should access the system; the second step is to open the host generation tab from the main page. In the third step user is required to create an archive which is a storage implemented within the application to store all the previous generated volunteers. In the fourth stage, it is optimal to change the settings for the host generation process. Any necessary changes to the default settings could be performed under the setting tab accessible from the host generation page.

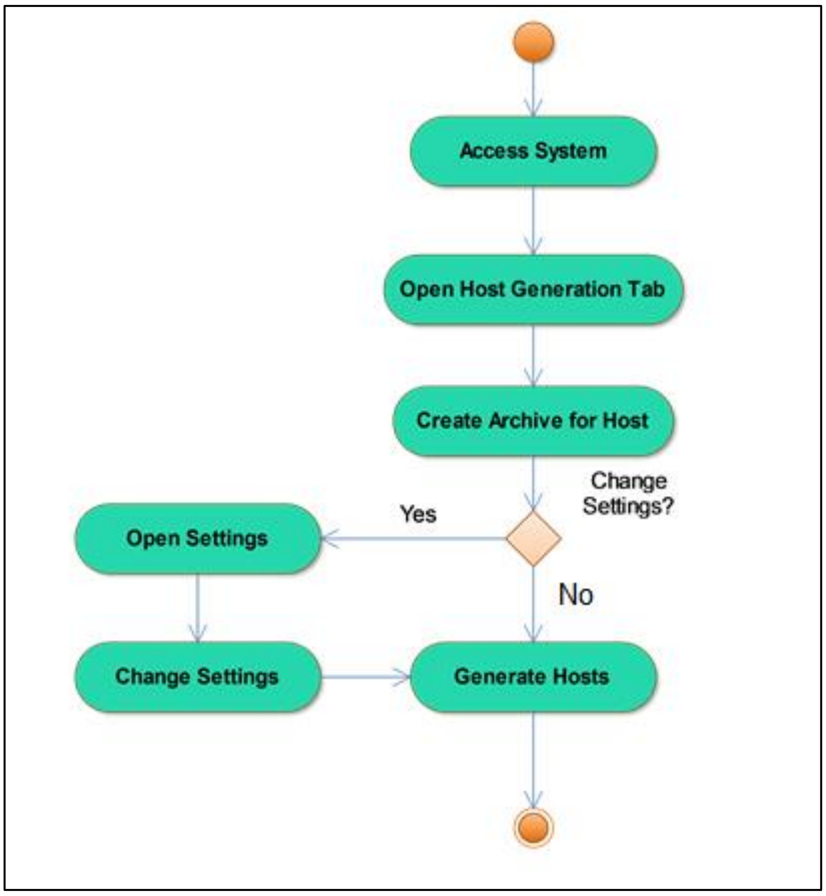
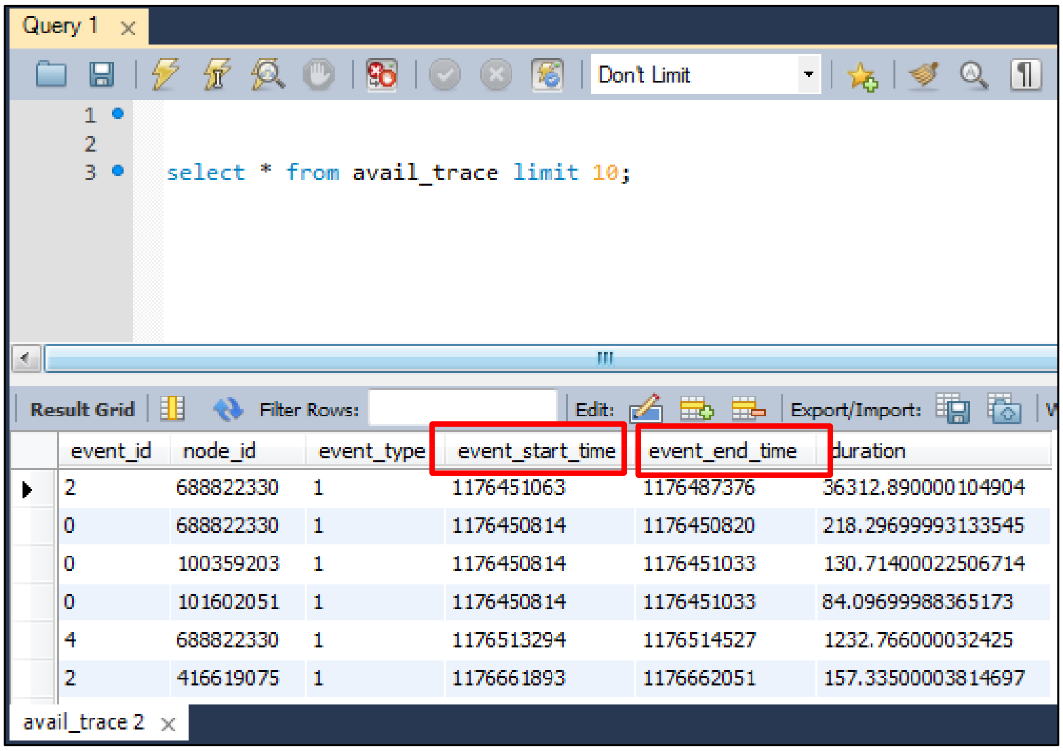


Figure 4.2: Activity diagram for host generation in prototype 1.0

At this point, the simulator requires a database in order to keep the information about the volunteers, workload archives, jobs, tasks, the data trace and the result of the simulation. Knowing the mentioned requirements, a database containing 6 tables has been created using MySql workbench as shown in Appendix A.

4.2.2 Problems Encountered With Prototype 1.0

As explained in section 2.4, this simulator uses a 'real trace' set of data in order to perform its simulation. This data has been stored in a database table named as "avail_trace" (referring to Appendix A). This table contains 8,497,274 rows of data storing information about each volunteer. Figure 4.3 shows all the information stored in 'avail_trace' table.



```
select * from avail_trace limit 10;
```

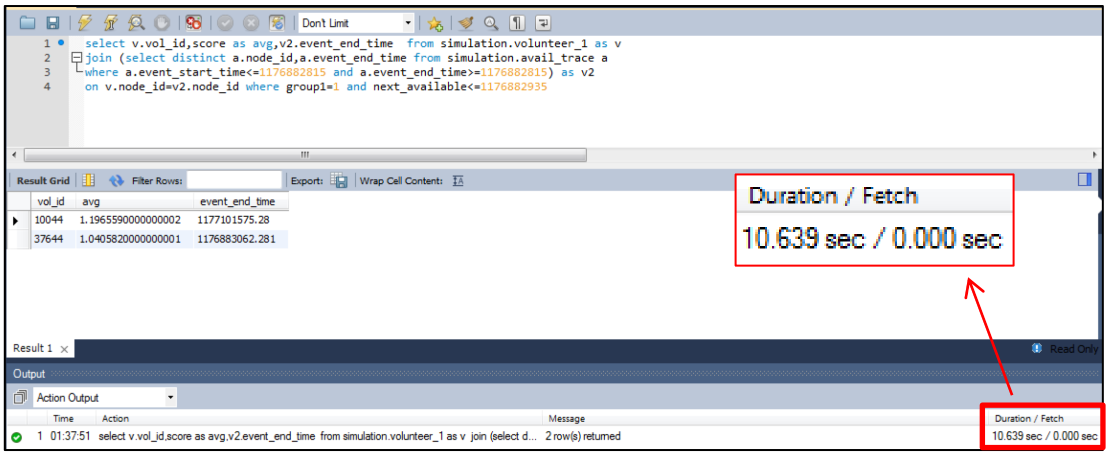
event_id	node_id	event_type	event_start_time	event_end_time	duration
2	688822330	1	1176451063	1176487376	36312.890000104904
0	688822330	1	1176450814	1176450820	218.29699993133545
0	100359203	1	1176450814	1176451033	130.71400022506714
0	101602051	1	1176450814	1176451033	84.09699988365173
4	688822330	1	1176513294	1176514527	1232.766000032425
2	416619075	1	1176661893	1176662051	157.33500003814697

Figure 4.3: Structure of 'Avail_trace' Table

The column "event_start_time" in Figure 4.3 refers to the time and date which a volunteer started to be available in the grid desktop environment (all the dates are saved as epoch time), on the other hand, "event_end_time" refers to the time and date which a volunteer became unavailable due to any reason. Hence, it is self-explanatory that it is only possible to submit tasks to volunteers within the period of their availability which starts from the "event_start_time" and lasts till the "event_end_time". Thus, within the simulation process, prior to each task submission a query should be run in order to check the availability of the volunteer.

This query requires joining information from "volunteer" table and "avail_trace" in order to find the availability status of the volunteers in a specific time before each task submission.

As mentioned in section 2.4, the data acquired from SETI@Home project, contains information about 38,166 volunteers which is saved in "volunteer" table. Hence, the simulator is required to join two tables containing 38,166 and 8,497,274 rows of information before each task submission. As expected, at first this join would consume massive time and memory of the system. Testing proved that 10.639 seconds of time was required for this query to be run (Figure 4.4).



```
select v.vol_id,score as avg,v2.event_end_time  from simulation.volunteer_1 as v
join (select distinct a.node_id,a.event_end_time from simulation.avail_trace a
where a.event_start_time<=1176882815 and a.event_end_time>=1176882815) as v2
on v.node_id=v2.node_id where group1=1 and next_available<=1176882935
```

vol_id	avg	event_end_time
10044	1.1965590000000002	1177101575.28
37644	1.0405820000000001	1176883062.281

Duration / Fetch
10.639 sec / 0.000 sec

	Time	Action	Message	Duration / Fetch
1	01:37:51	select v.vol_id,score as avg,v2.event_end_time from simulation.volunteer_1 as v join (select d...	2 row(s) returned	10.639 sec / 0.000 sec

Figure 4.4: Duration to Run the Query for Volunteer Availability

The quantity of task submissions depends on the user requirement and can be calculated using Equation 1. In a standard grid simulation environment a total number of 4 tasks will be submitted per 10 minute of simulation time [43-45]. Thus, assuming for a simulation period of 1 month 4,320 task submissions are required.

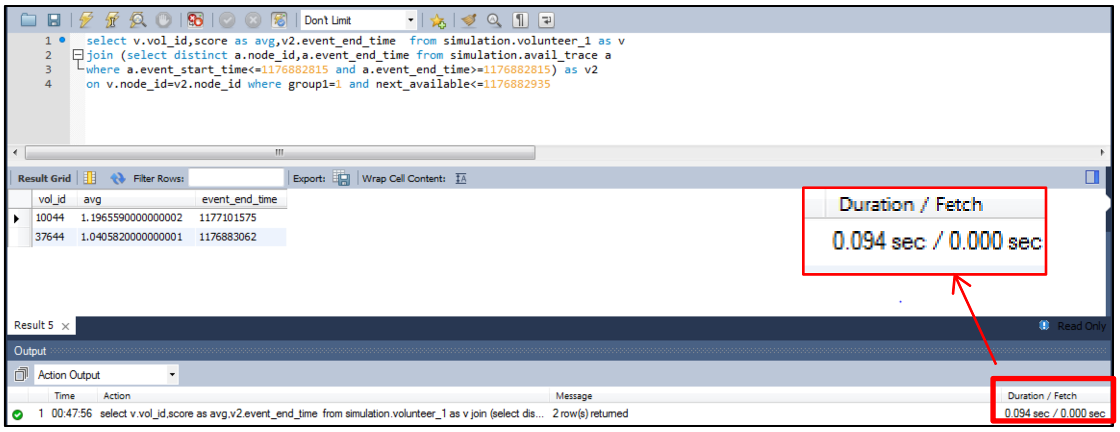
$$Total\ Number\ of\ Task\ Submition = \frac{Total\ Duration\ of\ Simulation\ (Minute)}{Task\ Arrival\ Intravel\ (Minute)} \qquad (1)$$

It is compulsorily to check the availability of volunteers before each task submission. Hence assuming that each availability check consumes 10.639 seconds of the simulation time, the total 4,320 checks before the entire tasks submissions would take up to 45,960.48 seconds (4,320 x 10.639) summing up to an approximate duration of enormous 12 hours, only for this section of simulation which is definitely not efficient. In order to overcome this issue, two B-TREE indexes had been implemented on “event_start_time” and “event_end_time” column since both of the mentioned columns were involved in the *where* condition of the query. Moreover, a total number of 1,000 partitions have been added to “avail_trace” table in order classify the table. The partitions were implemented on “event_start_time” column using a java *for loop* as shown in figure 4.5.

```
  String query;

      long Time=1176450815; //starting date of simulation (based on epoch time)

      query ="ALTER TABLE avail_trace PARTITION BY RANGE (event_start_time)  (";

       for (int i=0; i<1000;i++){ //This for loop runs for 1,000 times

         time+=600; //time interval of arriving tasks (depending on user)

        query+="PARTITION trace_"+i+" VALUES LESS THAN ("+time+") ENGINE =MyISAM,";

         }

      query+="PARTITION avil_max VALUES LESS THAN MAXVALUE ENGINE = MyISAM);";

System.out.println(query);//The ‘alter table’ query will be printed in the console
```

Figure 4.5: Java code for adding partitions to “avail_trace” table

After committing above changes, the query shall be run again. Figure 4.6 demonstrated the new outcome of the same query:



```
select v.vol_id,score as avg,v2.event_end_time  from simulation.volunteer_1 as v
join (select distinct a.node_id,a.event_end_time from simulation.avail_trace a
where a.event_start_time<=1176882815 and a.event_end_time>=1176882815) as v2
on v.node_id=v2.node_id where group1=1 and next_available<=1176882935
```

vol_id	avg	event_end_time
10044	1.1965590000000002	1177101575
37644	1.0405820000000001	1176883062

Duration / Fetch
0.094 sec / 0.000 sec

	Time	Action	Message	Duration / Fetch
1	00:47:56	select v.vol_id,score as avg,v2.event_end_time from simulation.volunteer_1 as v join (select dis...	2 row(s) returned	0.094 sec / 0.000 sec

Figure 4.6: Duration to Run the Same Query After Committing Changes

By adding two indexes and 1,000 partitions the duration of execution an "Availability" search on volunteers has been fallen from 10.639 to 0.094 which is a decrease of 10.545 seconds. This change is able to save time up to 10.545 seconds before each task submission; hence in total there is a potential decrease of 45,554.4 seconds (4,330 x 10.545) over the entire duration of simulation.

4.2.3 Prototype 2.0 Activities

The activities of Prototype 2.0 are same as Prototype 1.0; however, three additional components must be integrated within the system. As explained in section 4.1, viewing graphical charts, generating PDF reports and exporting the result as CSV files are part of the system requirements which have not been implemented in Prototype 1.0. The activity diagram of remaining components is shown in Figure 4.7 users are first required to access the system, the next step is to perform a simulation. After the simulation users are able to view graphical analysis of the simulation in the simulator, generate PDF reports about the simulated result and lastly to dump the entire result in a CSV file.

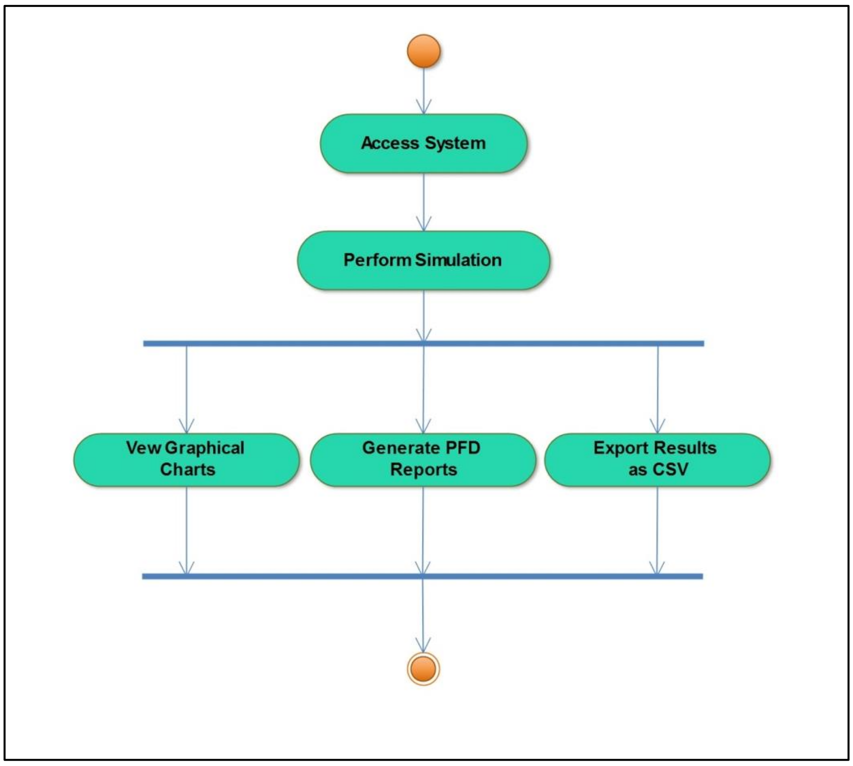


Figure 4.7: Activity diagram for remaining components in prototype 2.0

4.2.4 Problems Encountered With Prototype 2.0

During the development process of prototype 2.0, two major bugs were discovered by the HPC experts. First the application would crush after running for long time of periods with an error as *java.lang.OutOfMemoryError: Java heap space* prompted through the console following by an immediate crash. The second issue was a bit more complicated to cope with; the application performance would decrease significantly over time.

The first error was encountered due to an insufficient size of heap allocated to the JVM environment. The default heap size given to any java program is 64 MB which can be modified by customizing the *–Xmx* parameter prior to the running of the application [46, 47]. Since simulations normally run for a long duration, allocating 1024 MB of heap size is sufficient for the application [48] and the issue was resolved.

The second issue encountered due to memory leakage of objects within different classes. As described by Findeisen and Seidman [49] memory leak occurs when a section of the program is being held by the memory although that particular section has no use and is out of value. Despite the implementation of garbage collection technology in JAVA, memory leaks are still a frequent occurrence within JAVA applications. In JAVA, a memory leak occurs when there is a hidden reference to an unused object within the application. Despite the fact that the objects is no longer usable, the garbage collector would not erase the object from the memory due to existence of the reference [50].

In order to overcome this issue three steps need to be followed:

1. Confirm if there is any memory leakage
2. Take a heap dumb of the application using Netbeans Profiler
3. Resolve the issue

Netbeans Profiler can be attached to a running JVM in order to analyze the heap and garbage collection behavior during the simulation process. Any suspicious behavior of these two parameters can confirm a memory leakage. Figure 4.8 is a screenshot of the profiling result.

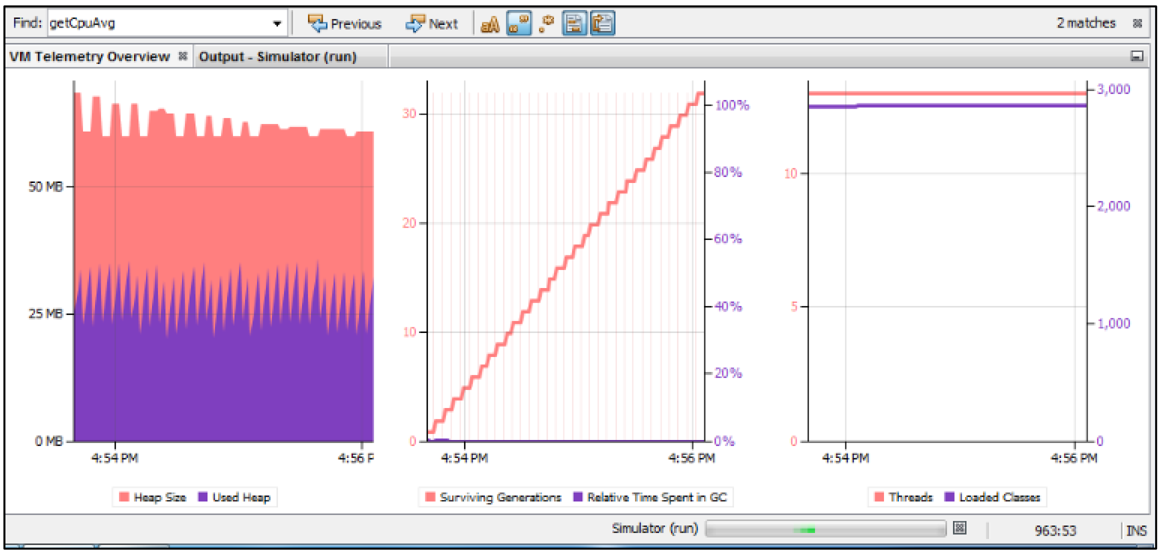
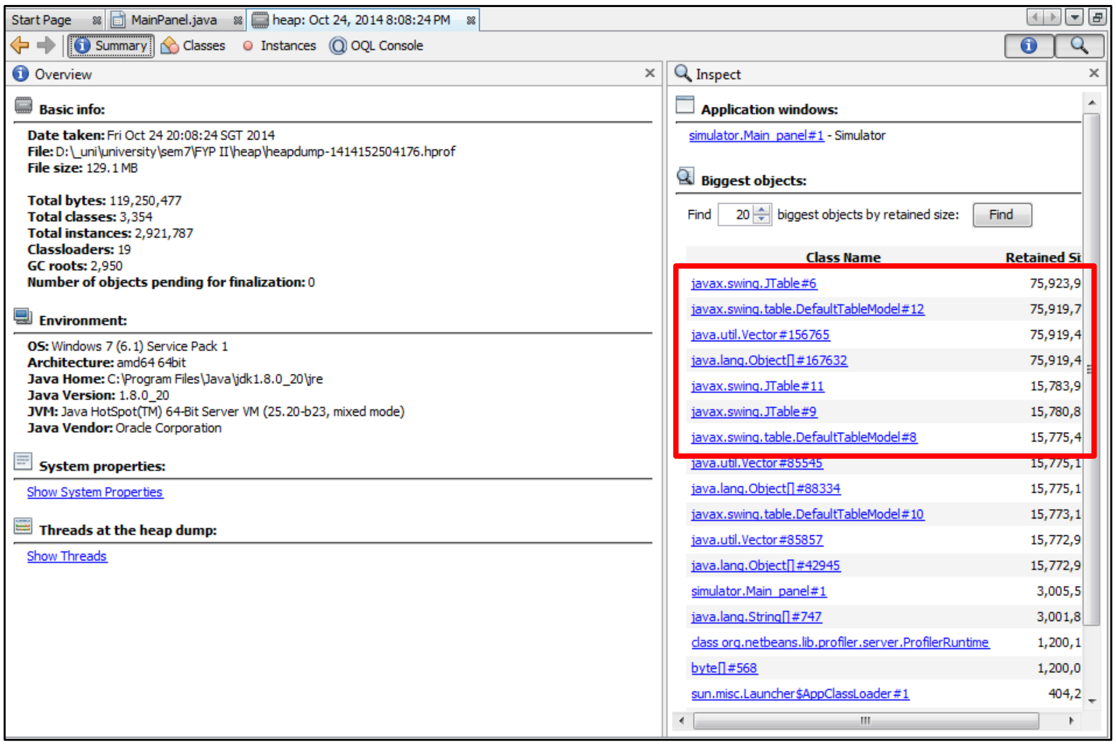


Figure 4.8: Result of Netbean Profiler

As seen in the figure above, three parameters can be evaluated using the built-in Netbeans profiler. The first diagram represents the total free and allocated heap size dynamically throughout the simulation process. The second shows the JAVA Garbage Collector (GC) behavior which in this case is indeed behaving incorrectly. “Surviving Generations” is referred to all objects allocated on the JVM heap since the profiling session started. In this particular case, objects are surviving each garbage collection and they are remaining in the heap (memory) which confirms the memory leak. Since the JAVA memory leak has been confirmed, a heap dump is required to be generated. The source of the leaking JAVA object can only be detected by evaluating a generated heap dump from the application. Thus, a heap dumb of prototype 2.0 had been generated as shown in Figure 4.9.



Basic info:

Date taken: Fri Oct 24 20:08:24 SGT 2014
File: D:_uni\university\sem7\FYP II\heap\heapdump-1414152504176.hprof
File size: 129.1 MB

Total bytes: 119,250,477
Total classes: 3,354
Total instances: 2,921,787
Classloaders: 19
GC roots: 2,950
Number of objects pending for finalization: 0

Environment:

OS: Windows 7 (6.1) Service Pack 1
Architecture: amd64 64bit
Java Home: C:\Program Files\Java\jdk1.8.0_20\jre
Java Version: 1.8.0_20
JVM: Java HotSpot(TM) 64-Bit Server VM (25.20-b23, mixed mode)
Java Vendor: Oracle Corporation

System properties:

Show System Properties

Threads at the heap dump:

Show Threads

Application windows:

simulator.Main_panel#1 - Simulator

Biggest objects:

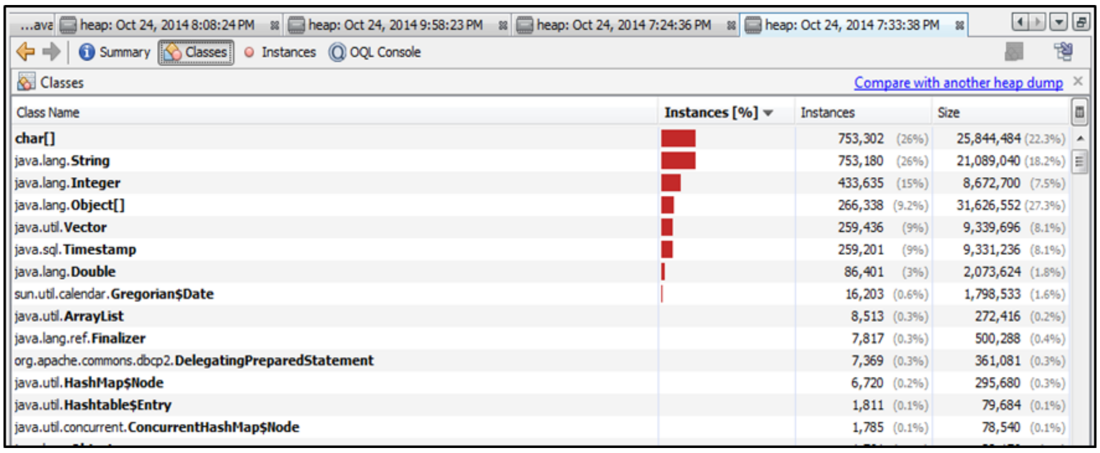
Find 20 biggest objects by retained size: Find

Class Name	Retained Si
javax.swing.JTable#6	75,923,9
javax.swing.table.DefaultTableModel#12	75,919,7
java.util.Vector#156765	75,919,4
java.lang.Object[]#167632	75,919,4
javax.swing.JTable#11	15,783,9
javax.swing.JTable#9	15,780,8
javax.swing.table.DefaultTableModel#8	15,775,4
java.util.Vector#85545	15,775,1
java.lang.Object[]#88334	15,775,1
javax.swing.table.DefaultTableModel#10	15,773,1
java.util.Vector#85857	15,772,9
java.lang.Object[]#42945	15,772,9
simulator.Main_panel#1	3,005,5
java.lang.String[]#747	3,001,8
class org.netbeans.lib.profiler.server.ProfilerRuntime	1,200,1
byte[]#568	1,200,0
sun.misc.Launcher$AppClassLoader#1	404,2

Figure 4.9: Summary of Java Heap Dump

This heap dump includes much useful information, a search of biggest objects ordered by size shows that 75 MB of the memory was occupied with a *JTable* object. After searching for the source, it was discovered that this *JTable* was implemented in the GUI of Prototype 1.0 for showing the entire 38,166 volunteers. It was determined that throughout the simulation GUI objects are the most memory consumers, hence the solution is to deactivate the memory consuming objects before the simulation and reload the after the simulation process.

Netbean Profiler’s heap dump enables users to view the garbage collection behavior for the entire classes. Further investigation on the heap dump revealed that some default java classes such as *char[]*, *java.lang.String* and *java.lang.Integer* are not being properly removed from the heap by the Garbage Collector. Figure 4.10 indicates the investigation process, basically using the heap it is possible to view the percentage of memory usage of each class throughout the simulation process. Using this information, it is possible to indicate which class is the most memory consuming class and henceforth that class’ Garbage Collection is not working properly.


...ava
heap: Oct 24, 2014 8:08:24 PM
heap: Oct 24, 2014 9:58:23 PM
heap: Oct 24, 2014 7:24:36 PM
heap: Oct 24, 2014 7:33:38 PM
Summary
Classes
Instances
OQL Console
Classes
Compare with another heap dump

Class Name	Instances [%]	Instances		Size	
char[]	🟥	753,302	(26%)	25,844,484	(22.3%)
java.lang.**String**	🟥	753,180	(26%)	21,089,040	(18.2%)
java.lang.**Integer**	🟥	433,635	(15%)	8,672,700	(7.5%)
java.lang.**Object[]**	🟥	266,338	(9.2%)	31,626,552	(27.3%)
java.util.**Vector**	🟥	259,436	(9%)	9,339,696	(8.1%)
java.sql.**Timestamp**	🟥	259,201	(9%)	9,331,236	(8.1%)
java.lang.**Double**	🟥	86,401	(3%)	2,073,624	(1.8%)
sun.util.calendar.**Gregorian$Date**	🟥	16,203	(0.6%)	1,798,533	(1.6%)
java.util.**ArrayList**		8,513	(0.3%)	272,416	(0.2%)
java.lang.ref.**Finalizer**		7,817	(0.3%)	500,288	(0.4%)
org.apache.commons.dbcp2.**DelegatingPreparedStatement**		7,369	(0.3%)	361,081	(0.3%)
java.util.**HashMap$Node**		6,720	(0.2%)	295,680	(0.3%)
java.util.**Hashtable$Entry**		1,811	(0.1%)	79,684	(0.1%)
java.util.concurrent.**ConcurrentHashMap$Node**		1,785	(0.1%)	78,540	(0.1%)

Figure 4.10: Heap Dump of Classes

After tracking the source of the leaking classes within the project, it was discovered that the leakage occurs due to the JAVA database connection and *prepared statements*. The core functionality of *prepared statement* is to retrieve data from the database[51]. It is compulsory to enclose *prepared statements* within JAVA *try-catch* blocks as shown In Figure 4.11. However, the disadvantage of using *try-catch* is that the objects within the blocks will not be properly closed after the end of each block [52].

```
    String sql="Any Query";

    try {

        con = datasource.getConnection();  //get the database connection

        pst=con.prepareStatement(sql);  // declaration of prepare statement

        rs=pst.executeQuery(sql);} // execution of the query

        catch (SQLException e ) {

         JOptionPane.showMessageDialog(null, e); // catch any exception

         log.debug(e); // generation logs

        } finally { //to close each statement

                    try { if (rs != null) rs.close(); } catch(Exception e){ }

                    try { if (stmt != null) stmt.close(); } catch(Exception e)
        {}

                    try { if (con != null) con.close(); } catch(Exception e)
        {}

                    try { if (pst != null) pst.close(); } catch(Exception e)
        {}}
```

Figure 4.11: Sample of normal *try-catch* statement

The solution of this problem is to use *try-with-recourses* statement rather than *try-catch* statement. The try-with-resources statement is a try statement that declares one or more resources. A resource is an object that must be closed after the program is finished with it. The *try-with-resources* statement ensures that each resource is closed at the end of the statement hence it eliminates any risk of memory leakage. Overall, through the process of developing any large JAVA project, it is recommended to use private access objects rather than public objects since private objects are forced to be closed after usage which allows the garbage collector to work properly.

A sample of *try-with-recourses* statement which has been implemented in the system is represented in Figure 4.12.

```
String sql="Any Query";

     try (connection con) { // take note that 'con' is not public anymore

         con = datasource.getConnection();  //get the database connection

         PreparedStatement  pst=con.prepareStatement(sql);  // declaration of
        //prepare statement

         ResultSet rs=pst.executeQuery(sql); // execution of the query

           //perform any task}

      catch (SQLException e ) {

           JOptionPane.showMessageDialog(null, e); // catch any exception

           log.debug(e); // generation logs

} //finally {} block is no more required and "con", "pst", "rs" objects get
closed at this point.
```

Figure 4.12: Sample of *try-with-resources* statement

At this stage, after all the changes have been committed a final testing is required to be performed in order to confirm there is no more memory leakage. Same as the initial testing, the built-in Netbeans Profiler will be attached to the JVM while running the simulator. Figure 4.13 shows the outcome of the testing.

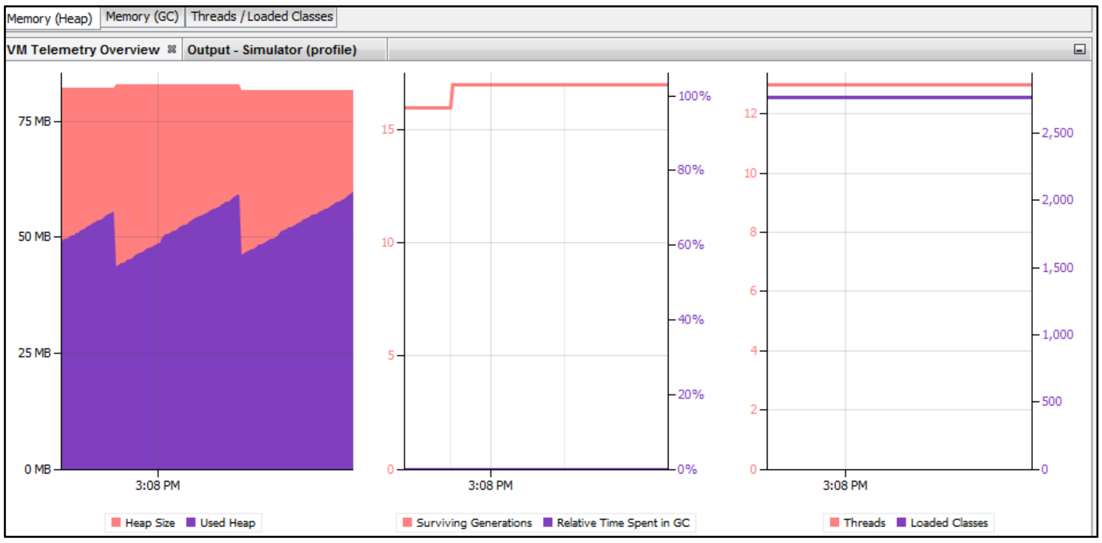


Figure 4.13: Result of Final Memory Leakage Testing

In summary, two prototypes have been released throughout duration of 6 months. Various bugs and issues have been confronted throughout the process of developing these prototypes; however, the major issues are summarized in Table 4.2.

Table 4.2: Summary of Released Prototypes and Issues

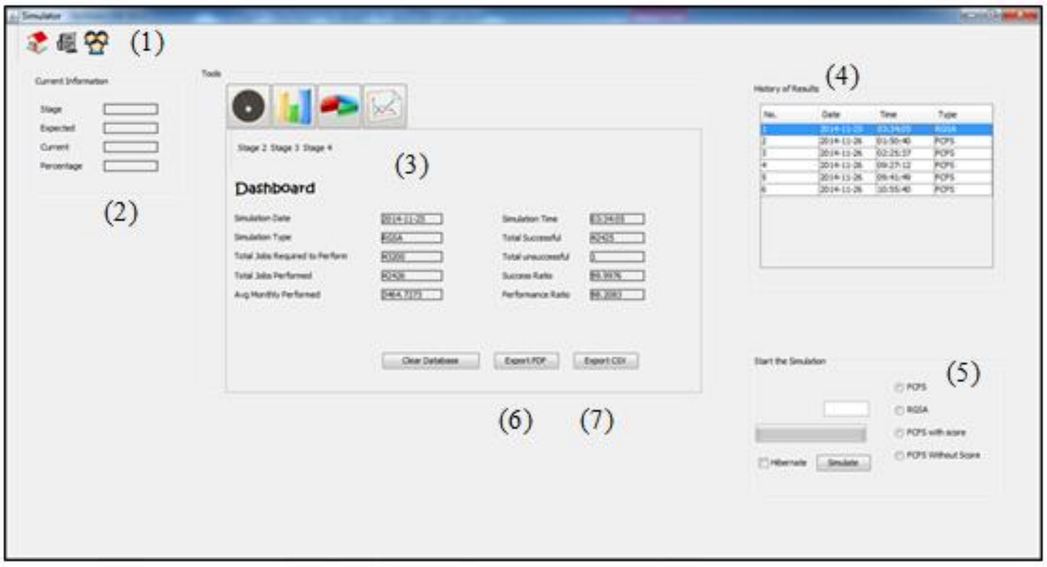
Version	Functionalities	Issues	Resolution
1.0	• Job Generation • Volunteer Generation	Query for checking volunteer availability takes too long	Implement two indexes and partitioning.
2.0	• View Graphical Charts • Generate PDF reports • Export Result in CSV	1. Heap size is too small 2. Program slows down over time	1. Configurator –Xmx parameter 2. Using heap dumps to locate the leaking objects

4.3 Design of Final System and Graphical User Interface

After the development of each prototype, semi-structured interviews have been conducted with the HPC experts in order to gather their feedbacks. Without the acceptance of the experts the development would not go into further stages, this enables the system developer to conduct unit testing on the system meanwhile the development stage. Sommerville [41] explains that unit testing could be performed throughout the development process with testing each developed component before moving to the next component. Duration of three weeks was requested from the HPC expert in order to use the system fully and test each single component of the system under different conditions. Several feedbacks have been given, however, majority of the feedbacks were related to improvements of the GUI.

Graphical User Interface is one of the most important outcomes; the GUI should be usable, reliable and easy to understand. Figure 4.14 demonstrates the GUI for the main panel of the simulator. The three icons on the toolbar (1) are referring to home panel, job generation panel and volunteer generation panel respectively. Under "Current Information" (2) all the information about any running simulation will be displayed such as the stage of simulation, how long it is expected to be run, current position of the loop (this is mainly implemented for debugging purposes and lastly the completed percentage of the simulation process.

On the dashboard (3) information about the previous simulation results will be displayed, however, prior to this a result history shall be selected from the "History of Results" (4) table. In the "Start of Simulation" (5) column, users are able to start their required simulation with entering the number of required dates in to the text-box and clicking "Simulate" button. With checking the hibernate checkbox beside the "Simulate" button, once the simulation is done the system will automatically get into hibernate condition after all saving the result.



Dashboard

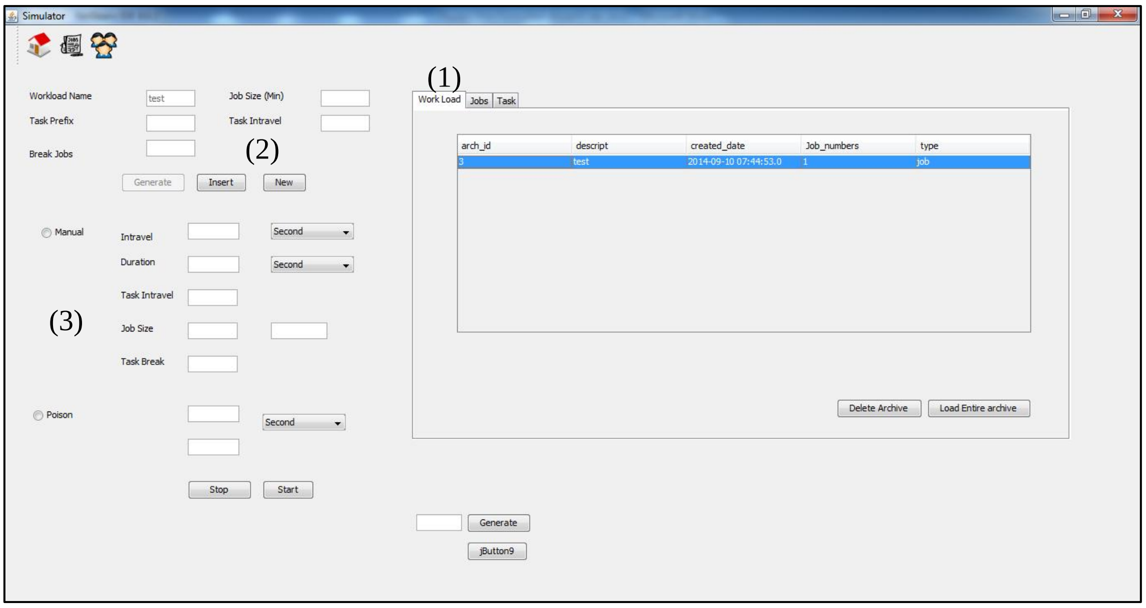
Simulation Date	2014-11-25	Simulation Time	03:34:03
Simulation Type	RGSA	Total Successful	82425
Total Jobs Required to Perform	83200	Total unsuccessful	1
Total Jobs Performed	82426	Success Ratio	99.9976
Avg Monthly Performed	3464.7375	Performance Ratio	99.2083

No.	Date	Time	Type
1	2014-11-25	03:34:03	RGSA
2	2014-11-26	01:50:40	FCFS
3	2014-11-26	02:25:37	FCFS
4	2014-11-26	09:27:12	FCFS
5	2014-11-26	09:41:49	FCFS
6	2014-11-26	10:55:40	FCFS

Figure 4.14: Main Panel

4.3.1 Job Generation

Figure 4.15 demonstrates the GUI for job generation. “Workload” (1) means to .archive generated jobs at one particular time frame. A workload is automatically created before jobs can be generated; this workload basically functions as an archive of generated jobs. Parameters (2) such as task prefix, job size, break job and task interval could be manipulated from this screen. The job arrival could be either in batches (manually configurator) or according to poison distribution (3).



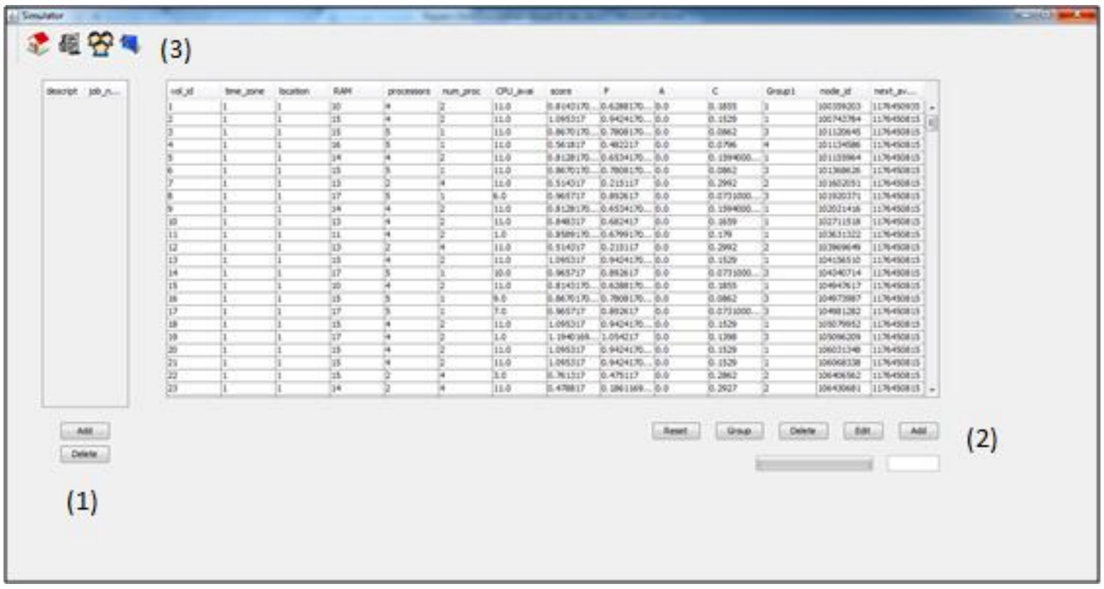
Workload Name: test

arch_id	descript	created_date	Job_numbers	type
3	test	2014-09-10 07:44:53.0	1	job

Figure 4.15: Job Generation Panel

4.3.2 Volunteer Generation Panel

Figure 4.16 indicates the GUI of host generation tab; it is possible to add archives by clicking on the “add” (1) button on the left side. In order to generate new hosts, the quantity of the required amount should be inserted into the text box following by clicking on the “add” (2) button on the right side which will generate new hosts. By default, the default settings of the grid desktop simulator will be used unless the user changes the settings by clicking on the “Setting” (3) button on the ribbon at the top.

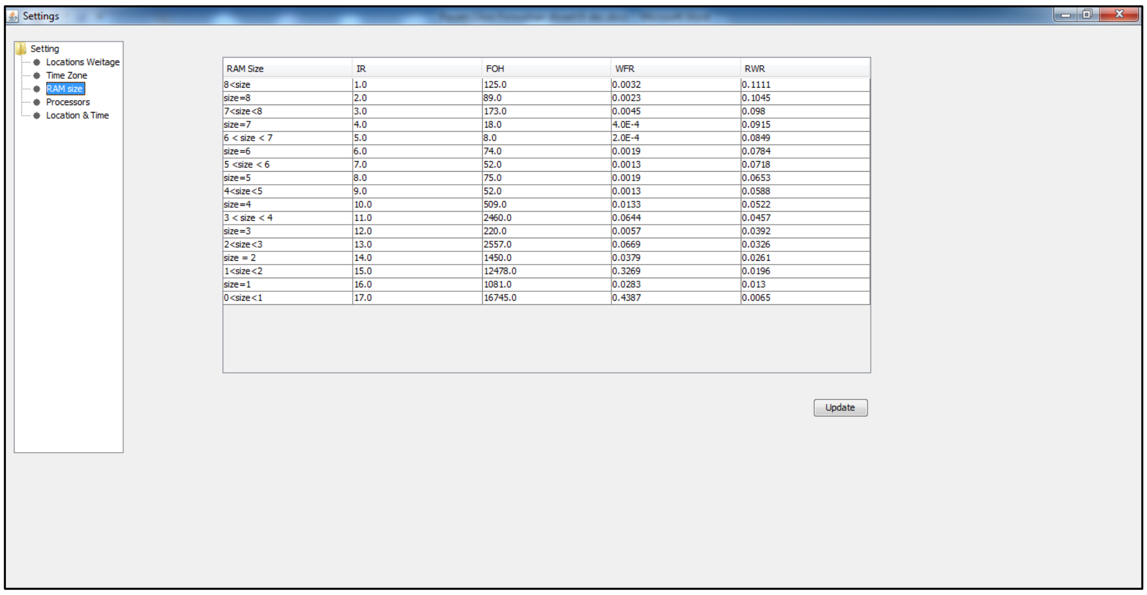


vol_id	time_zone	location	RAM	processors	num_proc	CPU_avai	score	F	A	C	Group1	node_id	next_av...
1	1	1	10	4	2	11.0	0.8140170...	0.6288170...	0.0	0.1855	1	100359303	1176450935
2	1	1	15	4	2	11.0	1.095317	0.9424170...	0.0	0.1529	1	100742764	1176450815
3	1	1	15	5	1	11.0	0.8670170...	0.7808170...	0.0	0.0862	3	101120645	1176450815
4	1	1	16	5	1	11.0	0.561817	0.482217	0.0	0.0796	4	101134586	1176450815
5	1	1	14	4	2	11.0	0.8128170...	0.6534170...	0.0	0.1594000...	1	101155964	1176450815
6	1	1	15	5	1	11.0	0.8670170...	0.7808170...	0.0	0.0862	3	101368626	1176450815
7	1	1	13	2	4	11.0	0.514317	0.215117	0.0	0.2992	2	101602051	1176450815
8	1	1	17	5	1	6.0	0.965717	0.892617	0.0	0.0731000...	3	101920371	1176450815
9	1	1	14	4	2	11.0	0.8128170...	0.6534170...	0.0	0.1594000...	1	102021416	1176450815
10	1	1	13	4	2	11.0	0.848317	0.682417	0.0	0.1659	1	102711518	1176450815
11	1	1	11	4	2	1.0	0.8589170...	0.6799170...	0.0	0.179	1	103631322	1176450815
12	1	1	13	2	4	11.0	0.514017	0.210117	0.0	0.2992	2	103969649	1176450815
13	1	1	15	4	2	11.0	1.095317	0.9424170...	0.0	0.1529	1	104156510	1176450815
14	1	1	17	5	1	10.0	0.965717	0.892617	0.0	0.0731000...	3	104340714	1176450815
15	1	1	10	4	2	11.0	0.8140170...	0.6288170...	0.0	0.1855	1	104647617	1176450815
16	1	1	15	5	1	9.0	0.8670170...	0.7808170...	0.0	0.0862	3	104973987	1176450815
17	1	1	17	5	1	7.0	0.965717	0.892617	0.0	0.0731000...	3	104981282	1176450815
18	1	1	15	4	2	11.0	1.095317	0.9424170...	0.0	0.1529	1	105079952	1176450815
19	1	1	17	4	2	1.0	1.1940169...	1.054217	0.0	0.1398	3	105096209	1176450815
20	1	1	15	4	2	11.0	1.095317	0.9424170...	0.0	0.1529	1	106021348	1176450815
21	1	1	15	4	2	11.0	1.095317	0.9424170...	0.0	0.1529	1	106068338	1176450815
22	1	1	15	2	4	3.0	0.761317	0.475117	0.0	0.2862	2	106406562	1176450815
23	1	1	14	2	4	11.0	0.478817	0.1861169...	0.0	0.2927	2	106430681	1176450815

Figure 4.16: Volunteer Generation Panel

4.3.3 Volunteer Setting

Figure 4.17 demonstrates the settings page implemented within the application, the user is able to manipulate all the simulation settings (mainly host generation volunteer setting). Settings such as volunteer's location, time zone, RAM and processor weightage could be modified using this functionality of the application.



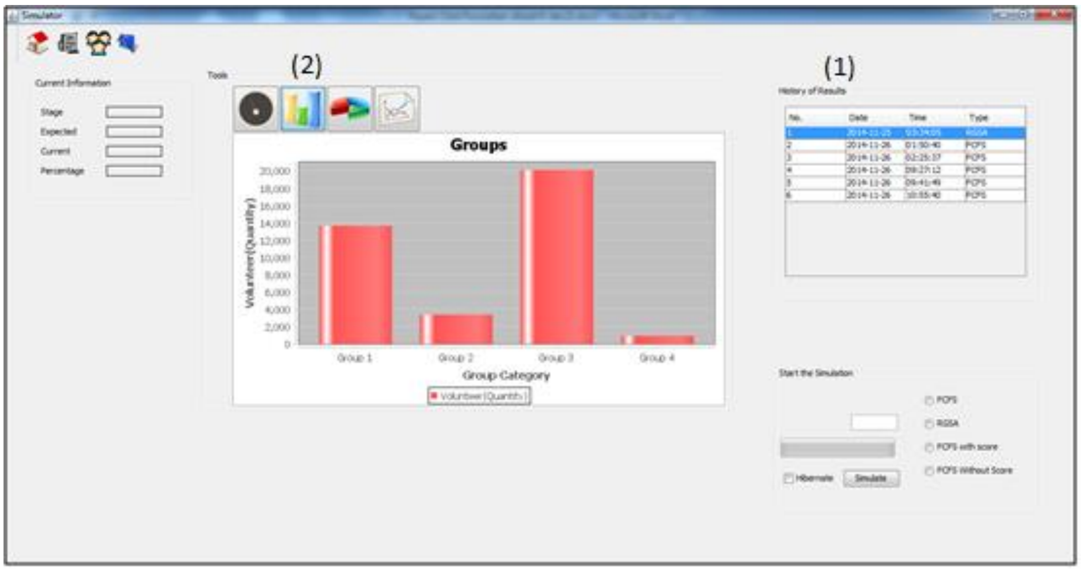
RAM Size	IR	FOH	WFR	RWR
8<size	1.0	125.0	0.0032	0.1111
size=8	2.0	89.0	0.0023	0.1045
7<size<8	3.0	173.0	0.0045	0.098
size=7	4.0	18.0	4.0E-4	0.0915
6 < size < 7	5.0	8.0	2.0E-4	0.0849
size=6	6.0	74.0	0.0019	0.0784
5 <size < 6	7.0	52.0	0.0013	0.0718
size=5	8.0	75.0	0.0019	0.0653
4<size<5	9.0	52.0	0.0013	0.0588
size=4	10.0	509.0	0.0133	0.0522
3 < size < 4	11.0	2460.0	0.0644	0.0457
size=3	12.0	220.0	0.0057	0.0392
2<size<3	13.0	2557.0	0.0669	0.0326
size = 2	14.0	1450.0	0.0379	0.0261
1<size<2	15.0	12478.0	0.3269	0.0196
size=1	16.0	1081.0	0.0283	0.013
0<size<1	17.0	16745.0	0.4387	0.0065

Figure 4.17: Setting Panel

4.3.4 Viewing Graphical Charts

Graphical charts are representation of data in a visual format, which are easier for users to digest and understand. The aim of including a graphical chart within your project is to demonstrate a set of data which in this case is the result of the simulation. In each simulation there are three essential factors which the user is required to know after each simulation process. The first important factor is to compare the volunteer's grouping before and after simulation.

Grouping of volunteers has been used in various researches in order to increase the reliability of desktop grid environment [45, 53, 54]. Using the grouping approach, volunteers are classified into different groups based on their computing characteristics such as RAM size, CPU size and quantity of processors. These groupings could be dynamic based on your policy configuration; it is necessarily to compare the quantity of volunteers at the end of the simulation in order to discover the efficiency of the policy.

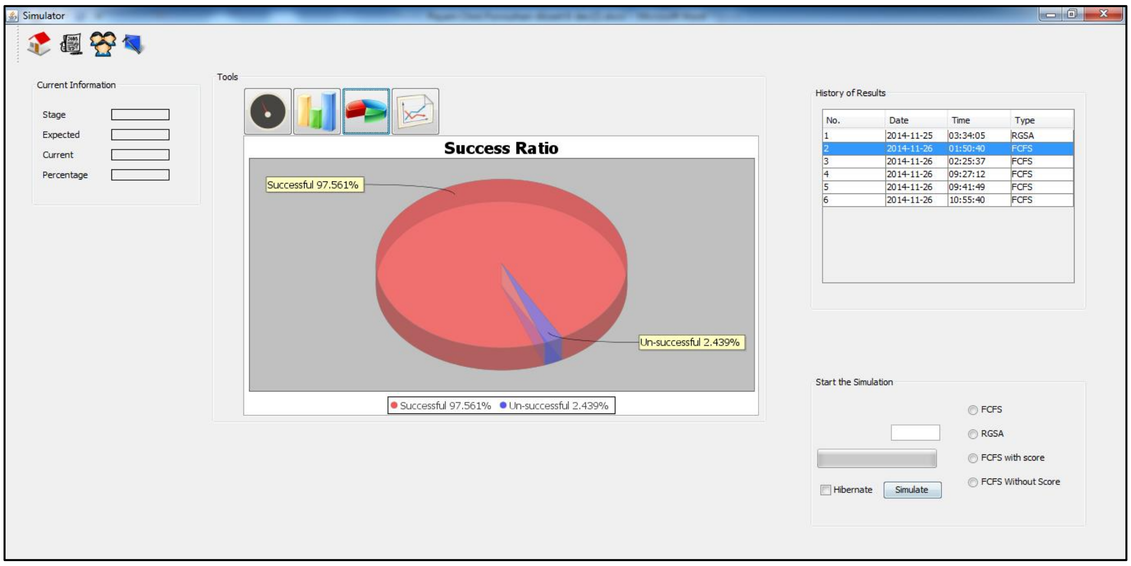


No.	Date	Time	Type
1	2014-11-25	03:34:05	RGSA
2	2014-11-26	01:50:40	FCFS
3	2014-11-26	02:25:37	FCFS
4	2014-11-26	09:27:12	FCFS
5	2014-11-26	09:41:49	FCFS
6	2014-11-26	10:55:40	FCFS

Figure 4.18: Bar Chart for Quantity of Grouping

The figure above shows the bar chart which is available to generate with selecting a simulation history (1) followed by clicking on the bar chart button (2). This graph, demonstrates the group category horizontally and quantity of volunteers vertically. Hence, users can compare the quantity of volunteers within a group prior and post simulation process.

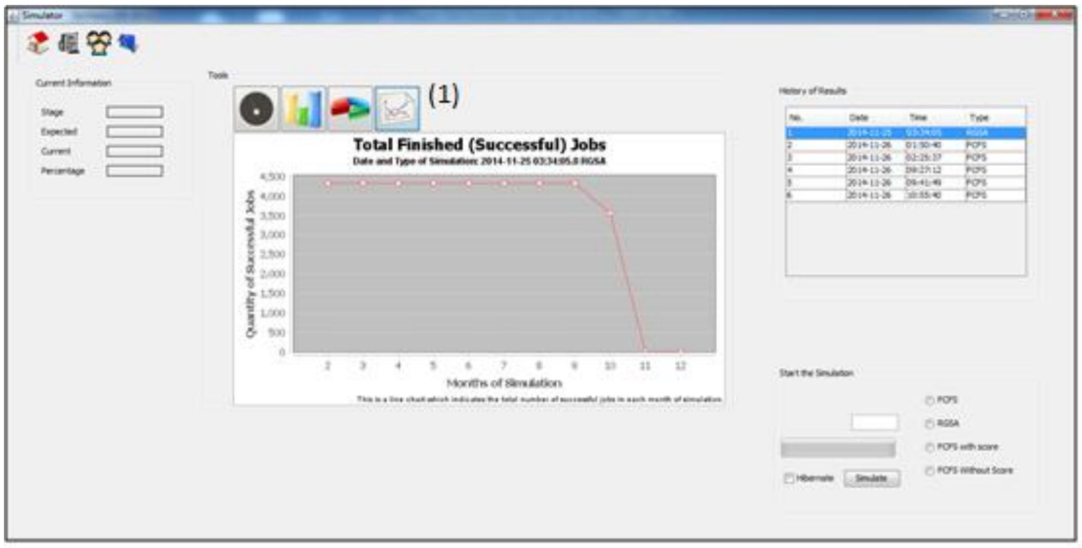
The second important result factor is the success ratio of the simulation process. Success ratio refers to the successful/unsuccessful quantity of jobs compared to the entire job submission quantity. This can be demonstrated using a pie chart represented in Figure 4.19.



No.	Date	Time	Type
1	2014-11-25	03:34:05	RGSA
2	2014-11-26	01:50:40	FCFS
3	2014-11-26	02:25:37	FCFS
4	2014-11-26	09:27:12	FCFS
5	2014-11-26	09:41:49	FCFS
6	2014-11-26	10:55:40	FCFS

Figure 4.19: Pie Chart of Successful and Unsuccessful Job Submission

The last factor which is compulsory for a user to know after each simulation is the quantity of participated volunteers throughout the simulation period. This can be demonstrated in a simple line graph as in Figure 4.20. By clicking on the Linear Graph Button (1) a linear chart will be constructed which contains the information about the quantity of used volunteers throughout the simulation process. A volunteer is considered as “used” or “engaged” if at least one task submission occurred to the volunteer (regardless the outcome of it). The horizontal axis refers to the number of simulated months throughout the simulation, the vertical axis indicated the quantity of engaged volunteers.


Simulator
Current Information
Stage
Expected
Current
Percentage
Tools
(1)
Total Finished (Successful) Jobs
Date and Type of Simulation: 2014-11-25 03:34:05.0 RGSA
Quantity of Successful Jobs
Months of Simulation
This is a line chart which indicates the total number of successful jobs in each month of simulation
History of Results
Start the Simulation
FCFS
RGSA
FCFS with score
FCFS Without Score
Hibernate
Simulate

No.	Date	Time	Type
1	2014-11-25	03:34:05	RGSA
2	2014-11-26	01:50:40	FCFS
3	2014-11-26	02:25:37	FCFS
4	2014-11-26	09:27:12	FCFS
5	2014-11-26	09:41:49	FCFS
6	2014-11-26	10:55:40	FCFS

Figure 4.20: Linear Chart for Quantity of Engaged Volunteers

4.3.5 Generate PDF Reports and Export CSV Files

As drawn in section's 3.3.3 Use Case Diagram (Figure 3.2) and also demonstrated in section 4.1's System Requirement Table (Table 4.1) the system shall be able to generate PDF Reports and export the end result as CSV file on your local computer. These two functionalities could be perform through the system's dashboard located at the home panel as in Figure 4.14, in this panel by clicking on "Export PDF" button (6) or "Export CSV" button (7) user's demanded action would be performed. Appendix B indicates an instance of a PDF report and Appendix C an example of an exported result in CSV format.

4.4 Implementation of Final System

In this phase, the finished version of the system should be evaluated by the end user. In this case, the simulator was presented to the experts in HPC center at UTP in order to get feedback and comments. In the first semi-structured interview the feedbacks were both positive and negative. The negative feedbacks were about the application's GUI.

In prototype 1.0, the GUI would freeze during the process due to the heavy load of work. This issue has been resolved by implementing a java swingWork framework for improving the concurrency. However, this issue was resolved in the final version of the simulator but the HPC experts demanded a major change of the layout behavior. Previously, a normal JAVA layout was implemented to the system. Hence, new JAVA forms would appear above the previous forms. As instance, in Figure 4.21 it is visible that the user is requesting for the "Volunteer Generation Tab" (1) from the "Main Panel", by clicking on that a new JAVA form (JForm) would appear in front of your current view (2). As mentioned by HPC experts, this behavior is not pleasant to the users according to basic Human Computer Interaction (HCI) rules

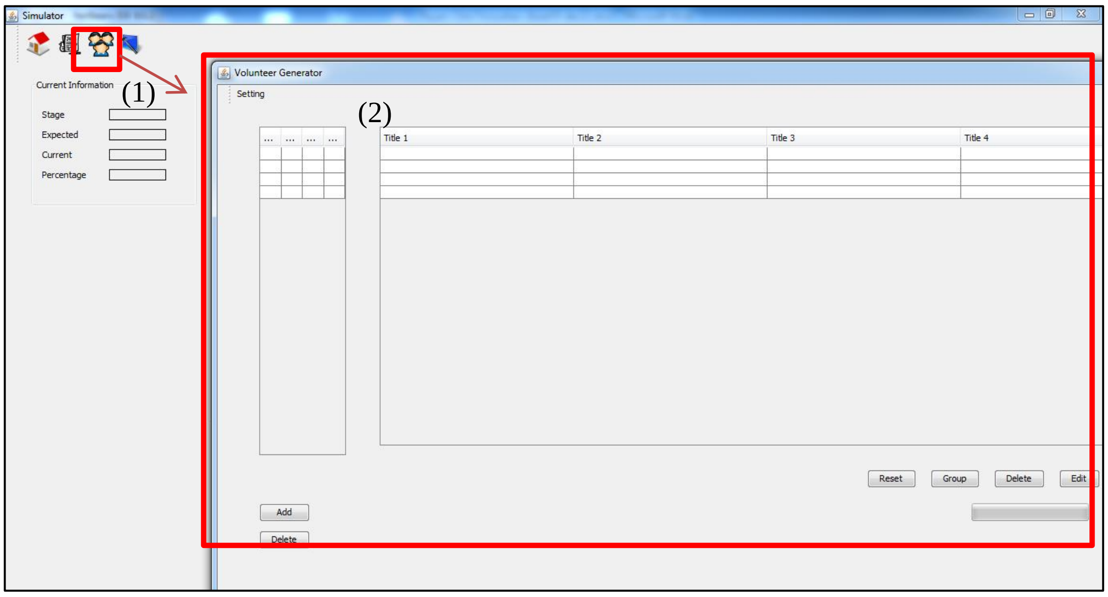
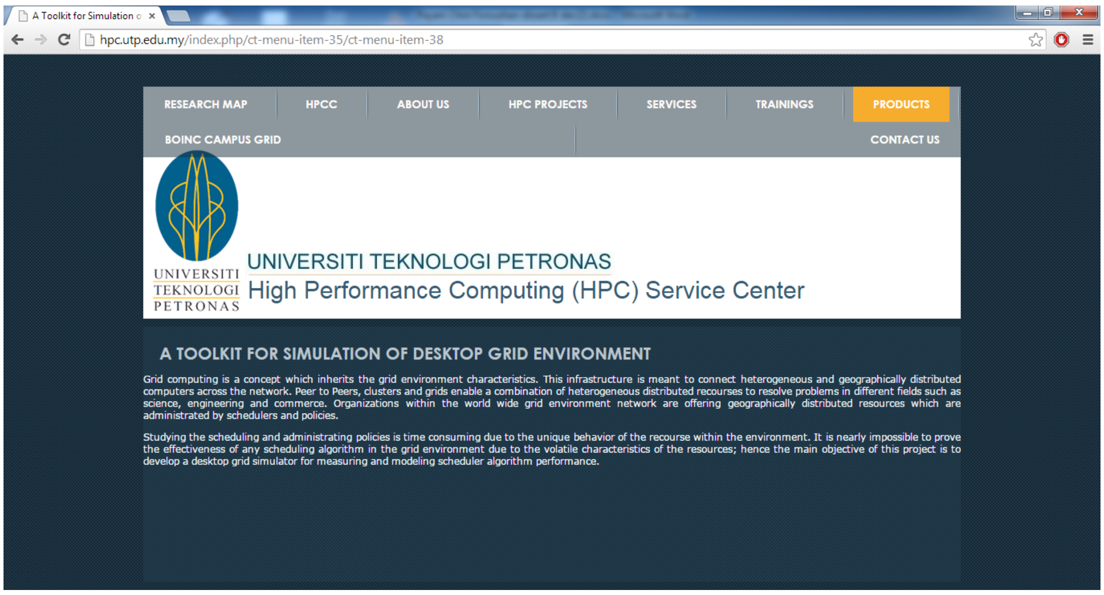


Figure 4.21: GUI issue Detected During Final Meeting

Hence, the implementation was postponed due to the new requested changes. In order to overcome this "interference" layer issue in JAVA, CardLayout flow could be implemented into the system [55, 56]. Using CardLayout enables an integrated and embedded view of all forms only in one page in other words, only one card (form) is visible at a time, and the container acts as a stack of cards. Once the changes have been committed a replacement meeting had been arranged.

On 26-November-2014, HPC experts officially accepted the project. Experts agreed that the system is fully functional and all the requirements have been fulfilled. Ever since, this desktop grid environment simulator is used by the HPC experts for personal research purposes. Figure 4.22 shows the official UTP HPC center website accessible through http://hpc.utp.edu.my/. This simulator has been listed as one of the official projects of HPC computing center after the final implementation. Implementing the system in HPC service center was not an objective of this paper, however, it is an enormous advantage for the system reliability level since grid computing researchers could modify and improve the system if required. Any academic activity should be aimed to increase knowledge and assist the advancement of science, thus it is an honor if the developed desktop grid environment simulator could assist any other research.


A Toolkit for Simulation o
hpc.utp.edu.my/index.php/ct-menu-item-35/ct-menu-item-38
RESEARCH MAP
HPCC
ABOUT US
HPC PROJECTS
SERVICES
TRAININGS
PRODUCTS
BOINC CAMPUS GRID
CONTACT US
UNIVERSITI TEKNOLOGI PETRONAS
UNIVERSITI TEKNOLOGI PETRONAS
High Performance Computing (HPC) Service Center

A TOOLKIT FOR SIMULATION OF DESKTOP GRID ENVIRONMENT

Grid computing is a concept which inherits the grid environment characteristics. This infrastructure is meant to connect heterogeneous and geographically distributed computers across the network. Peer to Peers, clusters and grids enable a combination of heterogeneous distributed recourses to resolve problems in different fields such as science, engineering and commerce. Organizations within the world wide grid environment network are offering geographically distributed resources which are administrated by schedulers and policies.

Studying the scheduling and administrating policies is time consuming due to the unique behavior of the recourse within the environment. It is nearly impossible to prove the effectiveness of any scheduling algorithm in the grid environment due to the volatile characteristics of the resources; hence the main objective of this project is to develop a desktop grid simulator for measuring and modeling scheduler algorithm performance.

Figure 4.22: Official Website of University Technology Petronas' HPC center

CHAPTER 5

CONCLUSION

5.1 Conclusion

Desktop Grid environment’s main functionality is to break down enormous bulk of jobs into smaller tasks and distributes these tasks throughout its entire network. Within the grid network, distributed and heterogeneous volunteers perform these tasks and return it to the server. Having a large network of volunteers worldwide enables scientists to perform huge jobs using this grid network rather than using super computers.

The grid network could have an enormous size (depending on the number of volunteers), any changes on the parameters (job size, quantity of volunteers or etc.) of this network might be costly and time consuming. Hence, scientists and researchers tend to simulate an instance of the grid environment virtually using a simulator and any further testing or research would be performed on the simulated environment rather than the actual environment. Currently, only few desktop grid simulators are accessible for research purposes and researches agree that it is troublesome to use the current simulators due to their complexity.

A study conducted on other related simulators such as SimGrid, GridSim, GangSim, SimBOINC and SimBa proved that the current simulators use mathematical formulas and distributions in order to simulate some characteristics of the environment. Hence, it was concluded that in order to make the simulator more accurate, it is recommended to use a set of ‘real trace’ data rather than statistical information.

The research methodology used in this study was based on studying other related projects and interviewing experts from UTP’s HPC service center. According to all the gather information from reviewing other literatures and semi-structured interviews, the requirements of the system could be gathered and finalize. The simulator was developed using JAVA programming language based on the requirements using a prototyping development methodology. Two different prototypes have been developed throughout a period of 7 months; various challenges had been raised through this process which could be overcome with the assistance of HPC experts.

The desktop grid simulator toolkit is meant to simulate and model scheduling policy, job generation and host generation in a grid environment. Additional features such viewing graphical charts, generating PDF reports and exporting result as CSV files has also been implemented into the simulator. The process of the entire simulation process will take few minutes’ up to maximum few hours depending on the selected model size. The implemented GUI decreased the system’s complexity, as stated in the objectives, due to the implemented CardLayout. Since the simulator toolkit is based on an offline JAVA programming, there is no downtime or interruption resources involved in the grid environment.

The final system was implemented only after the HPC experts confirmed that the desktop grid environment is fully functional. This implementation was not among the objectives of this study, however, it increases the reliability of the simulator since researchers may improve and maintain the system.

5.2 Future works

Currently, it is among desktop grid simulators’ assumption that there is no network transmission time throughout the process of simulation. Network transmission time refers to the delay of job transmission due to any network difficulty. Unfortunately, SETI@home does not provide any data concerning this network delays throughout the grid environment.

It is among one of the current hypothesis of desktop grid environment simulators that this network transmission time equals to zero and does not exist, however, in does exist in the actual desktop grid environment. Thus, further researches could be conducted in order to discover a novel mathematical distribution for mimicking this characteristic of the actual grid environment and implement it in any desktop grid environment simulator to increase its reliability.

REFERENCES

[1] I. Foster and C. Kesselman, "Globus: A metacomputing infrastructure toolkit," *International Journal of High Performance Computing Applications,* vol. 11, pp. 115-128, 1997.

[2] Z. Balaton, G. Gombas, P. Kacsuk, A. Kornafeld, J. Kovács, A. C. Marosi*, et al.*, "Sztaki desktop grid: a modular and scalable way of building large computing grids," in *Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International*, 2007, pp. 1-8.

[3] I. Foster and A. Iamnitchi, "On death, taxes, and the convergence of peer-to-peer and grid computing," in *Peer-to-Peer Systems II*, ed: Springer, 2003, pp. 118-128.

[4] L. F. Sarmenta, "Volunteer computing," Massachusetts Institute of Technology, 2001.

[5] S. Choi, M. Baik, C. Hwang, J. Gil, and H. Yu, "Volunteer availability based fault tolerant scheduling mechanism in desktop grid computing environment," in *Network Computing and Applications, 2004.(NCA 2004). Proceedings. Third IEEE International Symposium on*, 2004, pp. 366-371.

[6] F. Costa, L. Silva, G. Fedak, and I. Kelley, "Optimizing data distribution in desktop grid platforms," *Parallel Processing Letters,* vol. 18, pp. 391-410, 2008.

[7] B. Donassolo, H. Casanova, A. Legrand, and P. Velho, "Fast and scalable simulation of volunteer computing systems using simgrid," in *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, 2010, pp. 605-612.

[8] M. Taufer, A. Kerstens, T. Estrada, D. A. Flores, and P. J. Teller, "SimBA: A Discrete Event Simulator for Performance Prediction of Volunteer Computing Projects," in *PADS*, 2007, pp. 189-197.

[9] R. E. Shannon, *Systems simulation: the art and science*: Prentice-Hall, 1975.

[10] R. G. Ingalls, "Introduction to simulation," in *Proceedings of the 40th Conference on Winter Simulation*, 2008, pp. 17-26.

[11] I. Foster and C. Kesselman, *The Grid 2: Blueprint for a new computing infrastructure*: Elsevier, 2003.

[12] P. Domingues, B. Sousa, and L. Moura Silva, "Sabotage-tolerance and trust management in desktop grid computing," *Future Generation Computer Systems,* vol. 23, pp. 904-912, 2007.

[13] M. W. Mutka and M. Livny, "The available capacity of a privately owned workstation environment," *Performance Evaluation,* vol. 12, pp. 269-284, 1991.

[14] A. Oram, "Peer-to-peer: Harnessing the power of disruptive technologies," *SIGMOD Record,* vol. 32, p. 57, 2003.

[15] A. Chien, B. Calder, S. Elbert, and K. Bhatia, "Entropia: architecture and performance of an enterprise desktop grid system," *Journal of Parallel and Distributed Computing,* vol. 63, pp. 597-610, 2003.

[16] E. Korpela, D. Werthimer, D. Anderson, J. Cobb, and M. Lebofsky, "SETI@ HOME—massively distributed computing for SETI," *Computing in science & engineering,* vol. 3, pp. 78-83, 2001.

[17] D. P. Anderson, J. Cobb, E. Korpela, M. Lebofsky, and D. Werthimer, "SETI@ home: an experiment in public-resource computing," *Communications of the ACM,* vol. 45, pp. 56-61, 2002.

[18] S. Bagchi, "Simulation of grid computing infrastructure: challenges and solutions," in *Proceedings of the 37th conference on Winter simulation*, 2005, pp. 1773-1780.

[19] A. Manacero, R. S. Lobato, P. H. Oliveira, M. A. Garcia, A. I. Guerra, V. Aoqui, *et al.*, "iSPD: an iconic-based modeling simulator for distributed grids," in *Proceedings of the 45th Annual Simulation Symposium*, 2012, p. 5.

[20] H. Casanova, "Simgrid: A toolkit for the simulation of application scheduling," in *Cluster Computing and the Grid, 2001. Proceedings. First IEEE/ACM International Symposium on*, 2001, pp. 430-437.

[21] S. Hunold, T. Rauber, and F. Suter, "Redistribution aware two-step scheduling for mixed-parallel applications," in *Cluster Computing, 2008 IEEE International Conference on*, 2008, pp. 50-58.

[22] X.-m. Wen, W. Zhao, and F.-x. Meng, "Research of Grid Scheduling Algorithm Based on P2P_Grid Model," in *Electronic Commerce and Business Intelligence, 2009. ECBI 2009. International Conference on*, 2009, pp. 41-44.

[23] Y. Caniou, G. Charrier, and F. Desprez, "Analysis of tasks reallocation in a dedicated grid environment," in *Cluster Computing (CLUSTER), 2010 IEEE International Conference on*, 2010, pp. 284-291.

[24] R. Buyya and M. Murshed, "Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing," *Concurrency and computation: practice and experience*, vol. 14, pp. 1175-1220, 2002.

[25] J. Yu and R. Buyya, "Scheduling scientific workflow applications with deadline and budget constraints using genetic algorithms," *Scientific Programming,* vol. 14, pp. 217-230, 2006.

[26] L. Yao, G. Dai, H. Zhang, S. Ren, and Y. Niu, "A novel algorithm for task scheduling in grid computing based on game theory," in *High Performance Computing and Communications, 2008. HPCC'08. 10th IEEE International Conference on*, 2008, pp. 282-287.

[27] B. Lu and H. Zhang, "Grid load balancing scheduling algorithm based on statistics thinking," in *Young Computer Scientists, 2008. ICYCS 2008. The 9th International Conference for,* 2008, pp. 288-292.

[28] C. L. Dumitrescu and I. Foster, "GangSim: a simulator for grid scheduling studies," in *Cluster Computing and the Grid, 2005. CCGrid 2005. IEEE International Symposium on*, 2005, pp. 1151-1158.

[29] M. R. K. Grace, S. S. Priya, and S. Surya, "A Survey on Grid Simulators," 2012.

[30] F. Berman, G. Fox, and A. J. G. Hey, *Grid Computing: Making the Global Infrastructure a Reality*: Wiley, 2003.

[31] D. Kosiur, *Understanding Policy-Based Networking*: Wiley, 2001.

[32] K. Ranganathan and I. Foster, "Decoupling computation and data scheduling in distributed data-intensive applications," in *High Performance Distributed Computing, 2002. HPDC-11 2002. Proceedings. 11th IEEE International Symposium on*, 2002, pp. 352-358.

[33] N. A. Singh and M. Hemalatha, "High performance computing network for cloud environment using simulators," *arXiv preprint arXiv:1203.1728,* 2012.

[34] S. Sotiriadis, N. Bessis, E. Asimakopoulou, and N. Mustafee, "Towards Simulating the Internet of Things," in *Advanced Information Networking and Applications Workshops (WAINA), 2014 28th International Conference on*, 2014, pp. 444-448.

[35] D. Kondo, "SimBOINC: A simulator for desktop grids and volunteer computing systems," ed, 2007.

[36] D. P. Anderson, "Boinc: A system for public-resource computing and storage," in *Grid Computing, 2004. Proceedings. Fifth IEEE/ACM International Workshop on*, 2004, pp. 4-10.

[37] F. Borrajo, Y. Bueno, I. De Pablo, B. Santos, F. Fernández, J. García, *et al.*, "SIMBA: A simulator for business education and research," *Decision Support Systems,* vol. 48, pp. 498-506, 2010.

[38] D. A. Flores, T. Estrada, M. Taufer, P. J. Teller, and A. Kerstens, "Simba: a discrete event simulator for performance prediction of volunteer computing projects," in *Proceedings of the 2006 ACM/IEEE conference on Supercomputing*, 2006, p. 168.

[39] B. Javadi, D. Kondo, J.-M. Vincent, and D. P. Anderson, "Discovering Statistical Models of Availability in Large Distributed Systems: An Empirical Study of SETI@home," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, pp. 1896-1903, 2011.

[40] J. Brevik, D. Nurmi, and R. Wolski, "Automatic methods for predicting machine availability in desktop grid and peer-to-peer systems," in *Cluster Computing and the Grid, 2004. CCGrid 2004. IEEE International Symposium on*, 2004, pp. 190-199.

[41] I. Sommerville, *Software Engineering*: Pearson Education, 2011.

[42] A. Dennis, R. M. Roth, and B. H. Wixom, *System [sic] Analysis and Design*: John Wiley & Sons, Limited, 2013.

[43] A. Galstyan, K. Czajkowski, and K. Lerman, "Resource allocation in the grid using reinforcement learning," in *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 3*, 2004, pp. 1314-1315.

[44] M. Tang, B.-S. Lee, X. Tang, and C.-K. Yeo, "The impact of data replication on job scheduling performance in the Data Grid," *Future Generation Computer Systems*, vol. 22, pp. 254-268, 2006.

[45] J. H. Abawajy, "Fault-tolerant scheduling policy for grid computing systems," in *Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International*, 2004, p. 238.

[46] J.-S. Kim and Y. Hsu, "Memory system behavior of Java programs: methodology and analysis," in *ACM SIGMETRICS Performance Evaluation Review*, 2000, pp. 264-274.

[47] J.-S. Kim and Y. Hsu, "Analyzing Memory Reference Traces of Java Programs," in *Workload Characterization for Computer System Design*, ed: Springer, 2000, pp. 25-48.

[48] J. G. Fernández, "Performance improvement of multithreaded java applications execution on multiprocessor systems," Universitat Politècnica de Catalunya, 2005.

[49] P. Findeisen and D. I. Seidman, "Identifying memory leaks in computer systems," ed: Google Patents, 2008.

[50] M. D. Bond and K. S. McKinley, "Tolerating memory leaks," in *ACM Sigplan Notices*, 2008, pp. 109-126.

[51] R. Greenwald, R. Stackowiak, and J. Stern, *Oracle essentials: Oracle9 i, Oracle8 i & Oracle8*: O'Reilly & Associates, Inc., 2001.

[52] M. Konda, *What's New in Java 7?*: " O'Reilly Media, Inc.", 2011.

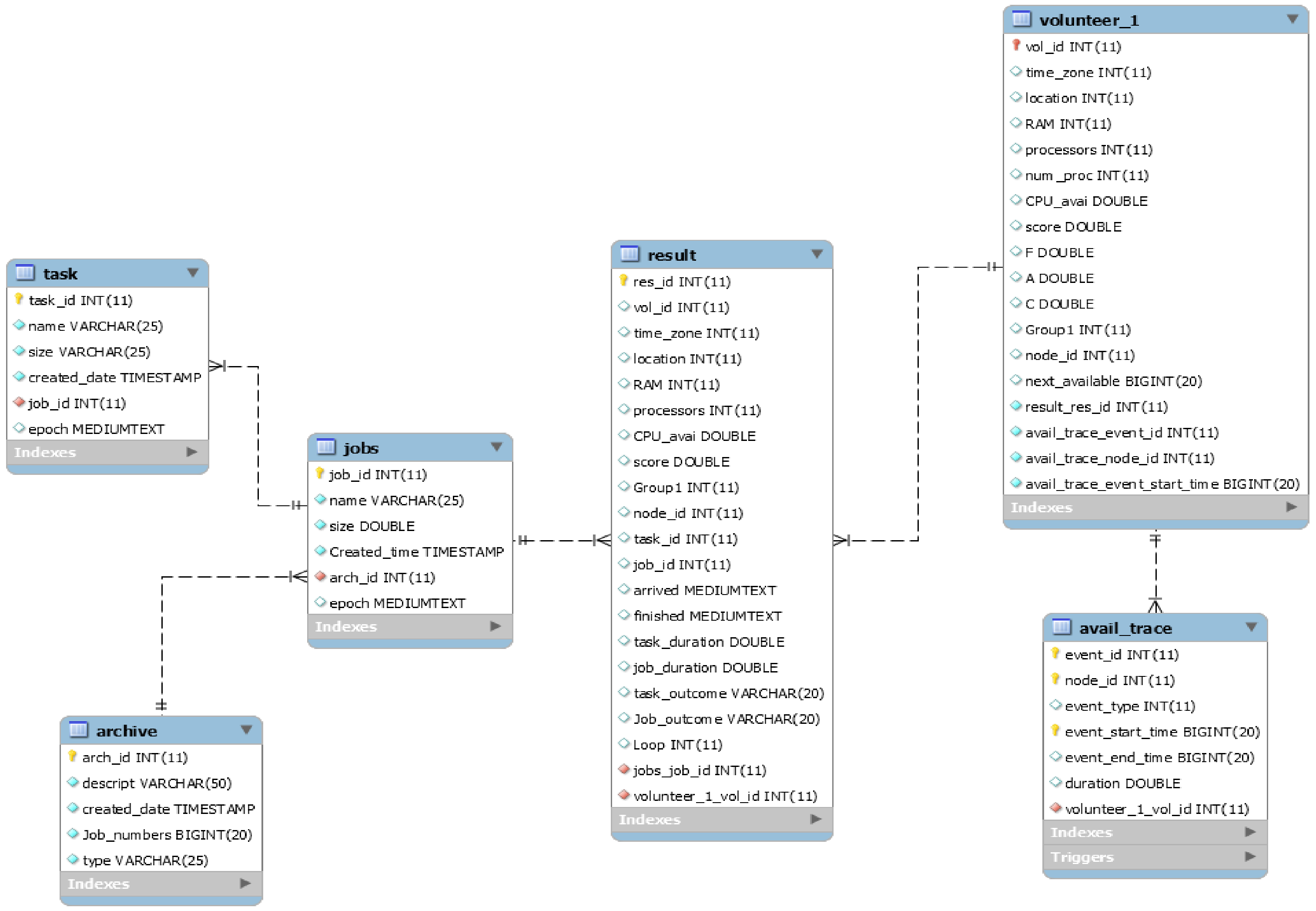
[53] M. Khan, I. Hyder, B. Chowdhry, F. Shafiq, and H. Ali, "A novel fault tolerant volunteer selection mechanism for volunteer computing," *Sindh University Research Journal—Science Series*, vol. 44, pp. 138-143, 2012.

[54] M. K. Khan, S. I. Hyder, G. U. Ahmed, S. Begum, and M. Aamir, "A Group Based Replication Mechanism to Reduce the Wastage of Processing Cycles in Volunteer Computing," *Wireless Personal Communications*, vol. 76, pp. 591-601, 2014.

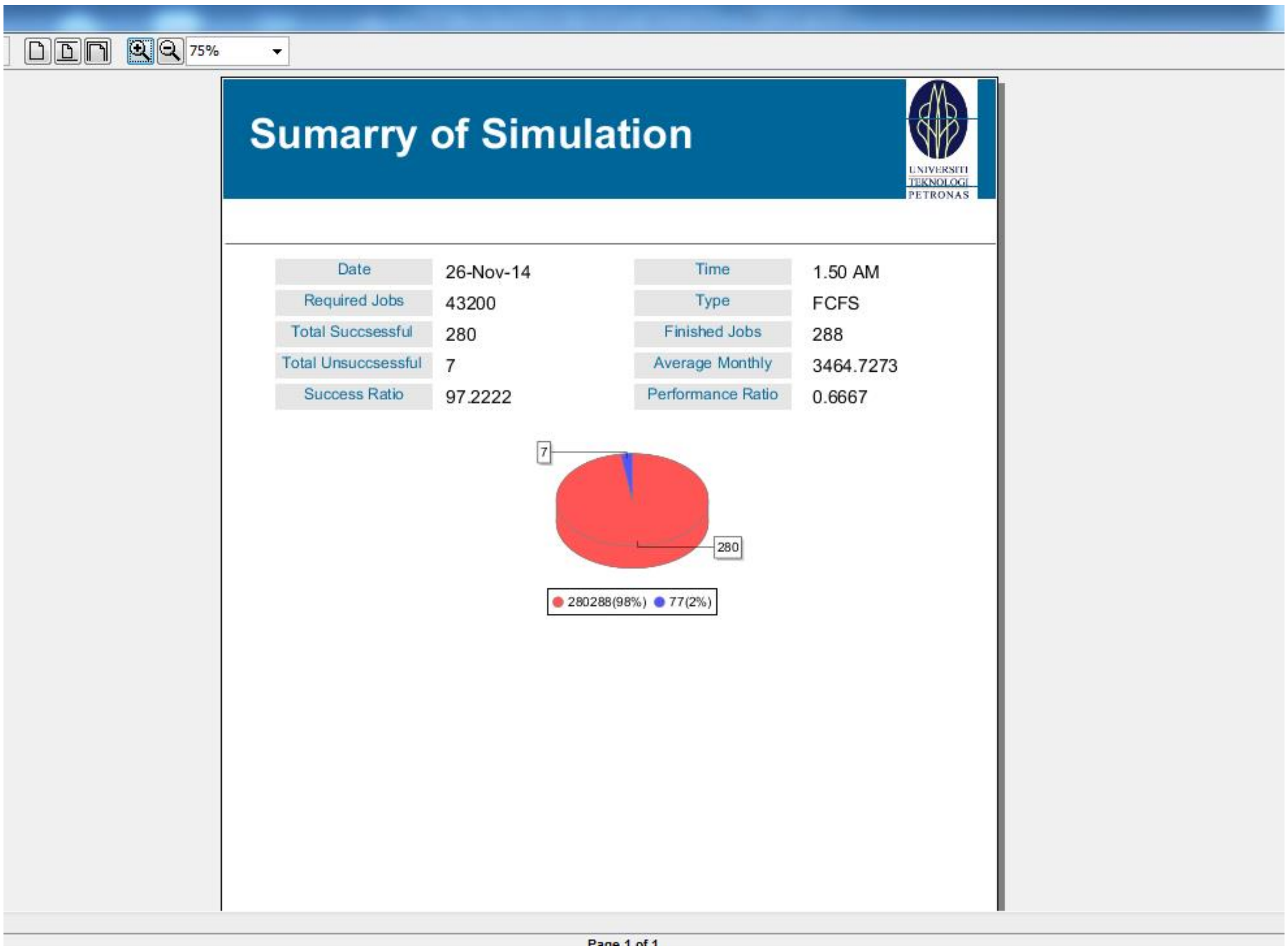
[55] J. Zukowski, *Java AWT reference* vol. 3: O'Reilly, 1997.

[56] J. Cowell, "The Layout Managers," in *Essential Java 2 fast*, ed: Springer, 1999, pp. 128-139.

Appendix A: Entity Relationship Diagram of the Implemented MySQL Database



Appendix B: Generated PDF Report by the Application



75%

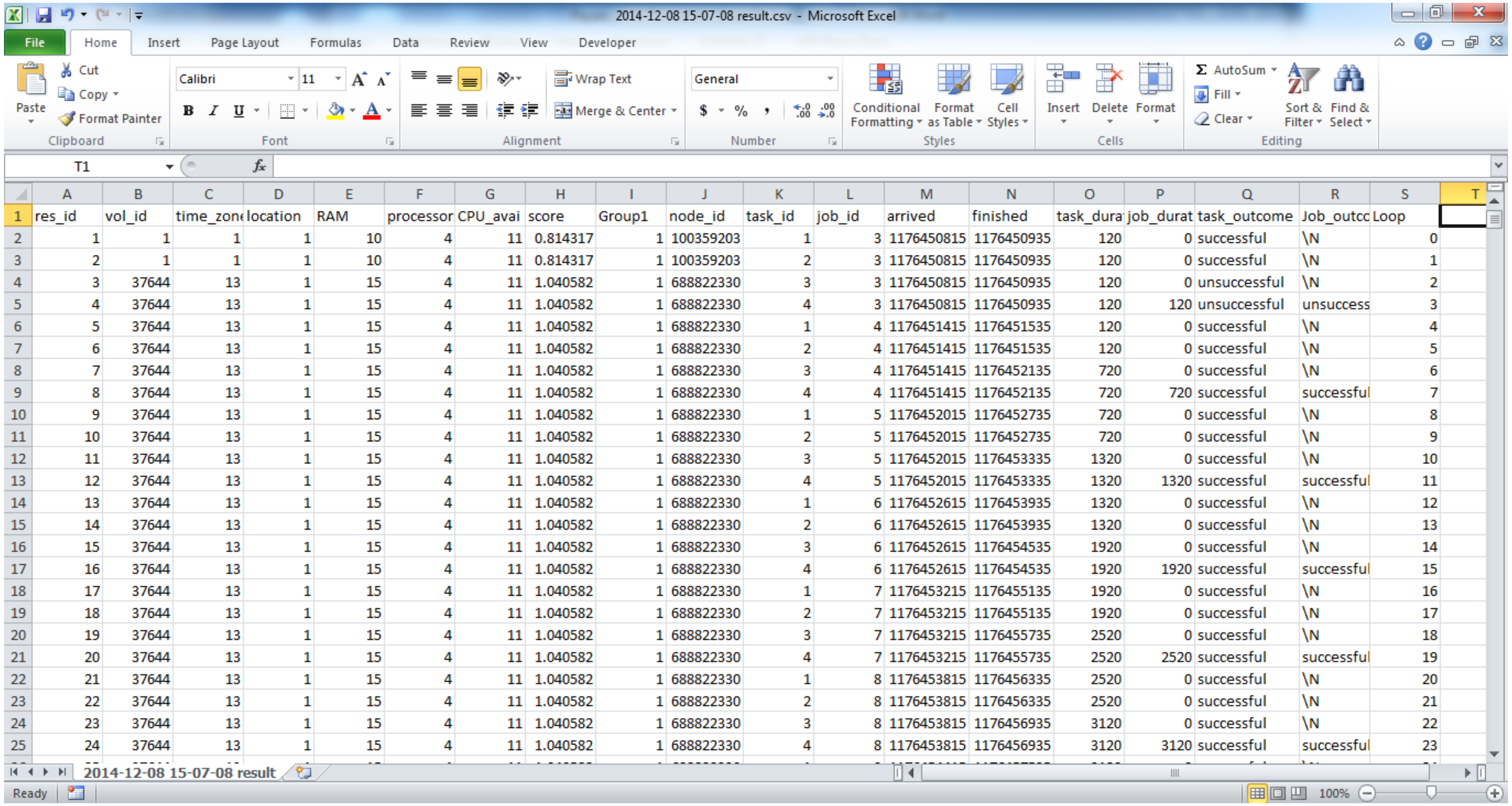
Sumarry of Simulation

UNIVERSITI TEKNOLOGI PETRONAS

Date	26-Nov-14	Time	1.50 AM
Required Jobs	43200	Type	FCFS
Total Succsessful	280	Finished Jobs	288
Total Unsuccsessful	7	Average Monthly	3464.7273
Success Ratio	97.2222	Performance Ratio	0.6667

Page 1 of 1

Appendix C: Exported Result in A CSV File.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1	res_id	vol_id	time_zon	location	RAM	processor	CPU_avai	score	Group1	node_id	task_id	job_id	arrived	finished	task_dura	job_durat	task_outcome	Job_outc	Loop	
2	1	1	1	1	10	4	11	0.814317	1	100359203	1	3	1176450815	1176450935	120	0	successful	\N	0	
3	2	1	1	1	10	4	11	0.814317	1	100359203	2	3	1176450815	1176450935	120	0	successful	\N	1	
4	3	37644	13	1	15	4	11	1.040582	1	688822330	3	3	1176450815	1176450935	120	0	unsuccessful	\N	2	
5	4	37644	13	1	15	4	11	1.040582	1	688822330	4	3	1176450815	1176450935	120	120	unsuccessful	unsuccess	3	
6	5	37644	13	1	15	4	11	1.040582	1	688822330	1	4	1176451415	1176451535	120	0	successful	\N	4	
7	6	37644	13	1	15	4	11	1.040582	1	688822330	2	4	1176451415	1176451535	120	0	successful	\N	5	
8	7	37644	13	1	15	4	11	1.040582	1	688822330	3	4	1176451415	1176452135	720	0	successful	\N	6	
9	8	37644	13	1	15	4	11	1.040582	1	688822330	4	4	1176451415	1176452135	720	720	successful	successful	7	
10	9	37644	13	1	15	4	11	1.040582	1	688822330	1	5	1176452015	1176452735	720	0	successful	\N	8	
11	10	37644	13	1	15	4	11	1.040582	1	688822330	2	5	1176452015	1176452735	720	0	successful	\N	9	
12	11	37644	13	1	15	4	11	1.040582	1	688822330	3	5	1176452015	1176453335	1320	0	successful	\N	10	
13	12	37644	13	1	15	4	11	1.040582	1	688822330	4	5	1176452015	1176453335	1320	1320	successful	successful	11	
14	13	37644	13	1	15	4	11	1.040582	1	688822330	1	6	1176452615	1176453935	1320	0	successful	\N	12	
15	14	37644	13	1	15	4	11	1.040582	1	688822330	2	6	1176452615	1176453935	1320	0	successful	\N	13	
16	15	37644	13	1	15	4	11	1.040582	1	688822330	3	6	1176452615	1176454535	1920	0	successful	\N	14	
17	16	37644	13	1	15	4	11	1.040582	1	688822330	4	6	1176452615	1176454535	1920	1920	successful	successful	15	
18	17	37644	13	1	15	4	11	1.040582	1	688822330	1	7	1176453215	1176455135	1920	0	successful	\N	16	
19	18	37644	13	1	15	4	11	1.040582	1	688822330	2	7	1176453215	1176455135	1920	0	successful	\N	17	
20	19	37644	13	1	15	4	11	1.040582	1	688822330	3	7	1176453215	1176455735	2520	0	successful	\N	18	
21	20	37644	13	1	15	4	11	1.040582	1	688822330	4	7	1176453215	1176455735	2520	2520	successful	successful	19	
22	21	37644	13	1	15	4	11	1.040582	1	688822330	1	8	1176453815	1176456335	2520	0	successful	\N	20	
23	22	37644	13	1	15	4	11	1.040582	1	688822330	2	8	1176453815	1176456335	2520	0	successful	\N	21	
24	23	37644	13	1	15	4	11	1.040582	1	688822330	3	8	1176453815	1176456935	3120	0	successful	\N	22	
25	24	37644	13	1	15	4	11	1.040582	1	688822330	4	8	1176453815	1176456935	3120	3120	successful	successful	23	