

**Deanship of Graduate Studies
Al-Quds University**



**PC Grid Computing Environment
In
Higher Education Institutions**

Bader Ahmed Bader Ajrab

M.Sc. Thesis

Jerusalem-Palestine

1434/2013

**PC Grid Computing Environment
In
Higher Education Institutions**

Prepared By:
Bader Ahmed Bader Ajrab

B.Sc. Electrical Engineering, 1998, Birzeit University,
Ramallah, Palestine

Supervisor: Dr. Labib Arafeh

A thesis submitted in Partial fulfillment of requirements for
the degree of Master of Electronics and Computer
Engineering/ Department of Electronics and Computer
Engineering/ Faculty of Engineering/ Graduate Studies -
Al-Quds University

1434/2013

Al-Quds University
Deanship of Graduate Studies
Electronics and Computer Engineering Department

Thesis Approval

PC Grid Computing Environment In Higher Education Institutions

Prepared By: Bader Ahmed Bader Ajrab
Registration No.: 20811425

Supervisor: Dr. Labib Arafeh

Master thesis submitted and accepted, Date: 15/06/2013

The names and signatures of the examining committee members are as follows:

1. Head of Committee: Dr. Labib Arafeh
2. Internal Examiner: Dr. Ahmad Qutob
3. External Examiner: Dr. Mohammed Aldasht

Signature:

Signature:

Signature:

Jerusalem – Palestine

1434/2013

Dedication

To my beloved parents, brothers, sisters, wife, and my children: Ahmad, Zina and Juman.

Bader Ajrab

Declaration

I certify that this thesis submitted for the degree of Master, is the result of my own research, except where otherwise acknowledged, and that this study (or any part of the same) has not been submitted for a higher degree to any other university or institution.

Signed:.....

Bader Ahmed Bader Ajrab

Date: 15/06/2013

Acknowledgement

I would like to thank my advisor Dr. Labib Arafah, for his wise guidance and continuous encouragement.

I am so grateful to Al-Quds Open University's Administration and Information and Communication Technology Center (ICTC) staff for facilitating this research.

I owe my wife for her endless support, understanding, patience, and also for her great help and encouragement.

Jerusalem, June 2013

Bader Ajrab

Abstract

Last decade witnessed a comprehensive revolution in technology; a revolution that had a great effect on people's lifestyle. It made great resources (CPU power, storage, memory, communication speed, and graphic processing units) available on regular PCs; such resources were not even available on servers in the past.

These ample resources have exceeded the actual needs of regular PC users (such as word processing and web surfing) so that these resources are not fully utilized.

In Higher Education Institutions (HEI), there is a great need for computational resources; large amounts of data are being accumulated and manipulated, in addition, a great portion of scientific experiments and student graduation projects need complex computational power that cannot be provided by a single desktop computer. On the other hand, supercomputers and/or dedicated clusters are not affordable by most HEIs in Palestine.

The convenient solution is to build a local PC Grid Computing environment by harnessing wasted computer resources from computers available in HEI computer labs with minimum or no cost, by deploying open-source grid computing frameworks.

In this research, Jerusalem Branch and Bethany study center at Al-Quds Open University have been chosen as a testbed for exploring the building and testing of a local PC Grid.

At first, the researcher tested the actual CPU utilization in a sample of computer lab PCs for 7 working days from 08:00 to 15:30 and revealed that CPU utilization does not exceed 10% for 90% of the experiment time.

Then, the researcher tested two frameworks to build a local desktop grid: Alchemi .NET; as an example of an open-source PC Grid computing framework, and Berkeley Open Infrastructure for Network Computing (BOINC) as an example for Public Resource Computing framework.

Alchemi was easy to install and configure (plug-and-play). It has also been noticed that application execution time in Alchemi-based grid is inversely proportional to the number of Executor nodes involved; which is a great advantage.

On the other hand, Alchemi causes network communication overhead; it also absorbs all available idle CPU power from Executor nodes causing CPU heating which is not a green approach.

Furthermore Alchemi had poor performance on PCs running Windows 7 compared to similar PCs running Windows XP.

BOINC server module runs on a dedicated Linux machine (mostly Debian) which requires more system administration skills than plug-and-play Alchemi.

Deploying BOINC clients on 3 gradually up to 43 computer lab PCs (total of 71 CPU cores) produced up to 106 GFLOPS provided that CPU utilization for connected PCs was set to 50% only. Assuming same experiment conditions, a total of 3550 CPU cores

available in QOU computer labs is expected to produce around 5.27 TeraFLOPS gained from wasted CPU cycles.

Network impact was also tested; routing delay and network traffic caused about 4.55% loss in the total gained GFLOPS for a 6 Mbps line and 13.63% loss in a 2 Mbps line.

BOINC client was also tested under windows XP and windows 7. It ranked competitive performance on Windows 7 compared to Windows XP (up to 57%).

Keywords: Grid Computing, BOINC, Alchemi.NET, Public Resource Computing, Volunteer Computing, Computational Power, PC Grid, Desktop Grid.

Table of Contents

DEDICATION	iv
DECLARATION	v
ACKNOWLEDGEMENT	vi
ABSTRACT	vii
TABLE OF CONTENTS	ix
LIST OF TABLES.....	xiii
LIST OF FIGURES.....	xiv
LIST OF APPENDICES	xvii

Chapter One

INTRODUCTION	1
1.1. Introduction	1
1.2. Motivation	2
1.3. Problem Description and Approach	2
1.4. Limitations of the study	3
1.5. Challenges	3
1.6. Contributions	3
1.7. Literature Survey	4
1.7.1. Availability of computer resources in Enterprise Desktop PCs	4
1.7.2. Power of Local PC Grid	5
1.7.3. The University of Westminster case study	6
1.7.4. Literature Survey Summary	7
1.8. Thesis organization.....	7

Chapter Two

BACKGROUND.....	9
2.1. Introduction	9
2.2. Grid Computing.....	9

2.3. Definition.....	9
2.4. Classification of Grids according to scope of resource sharing	10
2.4.1. Cluster Grids.....	10
2.4.2. Enterprise Grid.....	11
2.4.3. Utility Grid	12
2.4.4. Partner/Community Grids.....	12
2.5. Grid Architecture	13
2.6. PC Grid Computing	14
2.7. Public Resource Computing	15
2.8. Grid Middleware.....	16
2.8.1. Entropia	16
2.8.2. Grid MP	17
2.8.3. XtremWeb	17
2.8.4. distributed.net	17
2.8.5. HTCondor.....	18
2.8.6. Alchemi.NET.....	18
2.8.7. BOINC.....	19
2.8.7.1. BOINC server	20
2.8.7.2. Task server components	21
2.8.7.3. BOINC client.....	22
2.8.7.4. The Database	23
2.9. Summary.....	24

Chapter Three

METHODOLOGY	25
3.1. Description of experiment environment.....	25
3.2. Description of experiments.....	27
3.2.1. Measuring CPU utilization	27

3.2.2. Examining Alchemi .NET	27
3.2.3. Examining BOINC	27
3.3. Summary.....	29
Chapter Four	
EXPERIMENTS AND RESULTS	30
4.1. Part One: Measuring CPU utilization.....	30
4.2. Part Two: Examining Alchemi .NET	32
4.2.1. Testing Performance	33
4.2.2. Operating System effect	37
4.2.3. Testing network effect	38
4.3. Part Three: Examining BOINC	38
4.3.1. Testing Performance.....	38
4.3.2. Testing Network effect	41
4.3.3. Testing Operating System effect	41
4.4. Results and discussion	42
4.4.1. Part one: Measuring CPU utilization in computer lab PCs	42
4.4.2. Part two: Results of examining Alchemi .NET	43
4.4.3. Part three: Results of examining BOINC	50
4.5. Summary.....	52
Chapter Five	
CONCLUSION AND FUTURE WORK.....	53
5.1. Conclusion.....	53
5.2. Future Work.....	54
REFERENCES	56
Appendix A: CPU power at QOU Computer labs	60
Appendix B: Current Setup at QOU	62
B.1. General information	62

B.2. Computer labs	62
B.3. Networking	63
B.4. Lab PCs	65
B.5. PC Specifications	66
B.6. Authentication and Authorization	68
Appendix C: Dr. Rajkumar Buyya's offer	69
Appendix D: Alchemi .NET installation.....	70
D.1. Common requirements.....	70
D.2. Installing the Manager	70
D.3. Installing the Executor	71
Appendix E: BOINC installation	73
E.1. Installing BOINC server:	73
E.1.1. Choosing hardware:	73
E.1.2. Installing Software:	74
E.2. Creating and running a BOINC project	75
E.3. Deploying BOINC Clients	77
Appendix F: GLOSSARY	81
ملخص	83

List of Tables

Table No.	Table Title	Page
Table 2.1	PRC, Local PC Grid, and Grid comparison	16
Table 3.1	Specifications of experiment PCs	26
Table 3.2	The value of n for CPUs used in experiments	28
Table 4.1	Average execution time for PI calculator	34
Table 4.2	Execution time (in seconds) of PI calculator for increasing work loads	35
Table 4.3	Operating system effect on Alchemi	37
Table 4.4	Recorded vs. expected performance for BOINC example project	40
Table 4.5	Network effect on BOINC performance	41
Table 4.6	OS effects on BOINC client for different CPU and RAM	41
Table 4.7	Comparison between actual and ideal execution time, speed up factor	44
Table 4.8	Execution time (in seconds) for different decimal digits of PI	47
Table 4.9	a and c and R^2 of recorded data	48
Table B.1	Communication line bandwidths between ICTC and QOU branches/SC	65
Table B.2	Total number of PCs in QOU labs.	66

List of Figures

Figure No.	Figure Title	Page
Figure 1.1	Thesis Description	2
Figure 1.2	Local Desktop Grid at the University of Westminster	7
Figure 2.1	A typical form of cluster computing	11
Figure 2.2	Example Enterprise Grid infrastructure	11
Figure 2.3	Utility Grid architecture	12
Figure 2.4	Example of a partner grid	13
Figure 2.5	Common Grid Architecture	14
Figure 2.6	PC Grid Computing classification	14
Figure 2.7	A layered architecture for Alchemi framework	18
Figure 2.8	Distributed components and their relations	19
Figure 2.9	BOINC architecture	20
Figure 2.10	BOINC task server components	21
Figure 2.11	BOINC daemons and components	21
Figure 2.12	BOINC client overview	23
Figure 3.1	Jerusalem and Bethany SC sites	25
Figure 4.1	Screenshot of CPU Usage logger tool interface	30
Figure 4.2	Snapshot of sample log file	31
Figure 4.3	Average CPU utilization	32
Figure 4.4	Alchemi Grid	33
Figure 4.5	Average execution time for PI calculator (100 digits)	34
Figure 4.6	Speed up factor vs. number of Executors	35
Figure 4.7	A plot of workload vs. execution time with varying number of Executors	36
Figure 4.8	Speed up factor for increased workload with varying number of	36

Executors

Figure 4.9	Operating system effect on Alchemi	37
Figure 4.10	BOINC testing process	38
Figure 4.11	Monitoring server status	39
Figure 4.12	Computing preferences webpage	39
Figure 4.13	Harvested GFLOPS vs. number of CPU cores involved	40
Figure 4.14	Actual vs. ideal execution time	45
Figure 4.15	Trend line for actual execution time	46
Figure 4.16	Actual vs. ideal speed up factor	47
Figure 4.17	A plot of thread size vs. execution time	48
Figure 4.18	Alchemi Console showing available vs. used CPU power	50
Figure 4.19	Gained GFLOPS with Trend line	51
Figure B.1	QOU branches and study centers	62
Figure B.2	Computer lab distribution in QOU	63
Figure B.3	QOU network topology	64
Figure B.4	Memory distribution overview.	67
Figure B.5	Storage capacity overview	67
Figure B.6	Operating system distribution overview	67
Figure C.1	Snapshot of Dr.Buyya's offer	69
Figure D.1	Alchemi Manager GUI	71
Figure D.2	Alchemi Executor GUI	72
Figure E.1	Installing BOINC server block diagram	73
Figure E.2	Software installation block diagram	74
Figure E.3	Creating a sample BOINC project	75
Figure E.4	Starting BOINC project	76
Figure E.5	Stopping the BOINC project	76

Figure E.6	Deploying BOINC client block diagram	77
Figure E.7	Attaching BOINC client to the test project	77
Figure E.8	BOINC client attached to test project	78
Figure E.9	BOINC client uploading tasks	78
Figure E.10	BOINC client uploading finished jobs	78
Figure E.11	BOINC project configuration file on the client side	80

List of Appendices

Appendix A	CPU Power in QOU Computer labs	60
Appendix B	Current setup at QOU	62
Appendix C	Dr. Rajkumar Buyya's offer	69
Appendix D	Alchemi .NET installation	70
Appendix E	BOINC installation	73
Appendix F	Glossary	81

Chapter One

Introduction

1.1. Introduction

The term *Grid* or *Grid Computing* is derived from the electric power grid. It is a general term that combines different technologies and solutions to different target groups.

A Grid Computing system slices complex tasks into small subtasks that are sent to multiple interconnected computers to complete tasks more efficiently and quickly.

Since it emerged, grid computing was deployed on dedicated, high performance nodes and clusters only, but in the last two decades the processing power of Personal Computers (PCs) increased noticeably (according to Moore's law: the complexity of the chip doubles every 18 months) (Moore,1965). Local Area Networks (LANs) bandwidths increased to 1.0 Gbps. Also the Internet increased in bandwidth and expanded to the consumer market. Moreover, studies proved (Mutka, 1992) that the processors in PCs are most likely underutilized (idle) most of the time while PCs are running, and many PCs are not even turned on much of the time.

Due to these advances in CPU power and network bandwidth, new concepts such as Local Desktop Grid, and Public Resource Computing (PRC) emerged.

Local Desktop Grid, or Local PC Grid, harvests idle resources on non-dedicated ordinary desktop PCs in companies, universities, hospitals, and other institutions. These harvested resources are then harnessed to perform specific complex tasks.

Public Resource Computing technology, or Volunteer Computing, is a form of high performance computing in which volunteers from around the world donate a portion of their computers' resources to a computationally intensive research computation (so called "projects") (Anderson, 2003). Researchers who do not have access to supercomputing facilities can use PRC to increase their ability to process compute-intensive data at little to no extra cost.

In the case of Higher Educational Institutions (HEI), large number of PCs (and thus CPU power) are idle most of the time especially in computer laboratory PCs, so it will be useful (or maybe even profitable) to make use of this fact by building a Local PC Grid from such PCs and deploying complex computations on it without the need to purchase new hardware or equipment. This will serve student software projects that consume CPU power (such as simulation and image rendering), also will serve managerial software (such as manipulating accumulative averages for all university students for several years).

The primary goal of this thesis is to build a local PC grid in Al-Quds Open University (QOU) as a testbed distributed environment using two approaches:

- First: by deploying an open source grid computing middleware (Alchemi .NET).
- Second: by deploying customized open source PRC framework (BOINC).

For each approach, the overall performance is measured, and factors such as network connection and Operating System (OS), are tested.

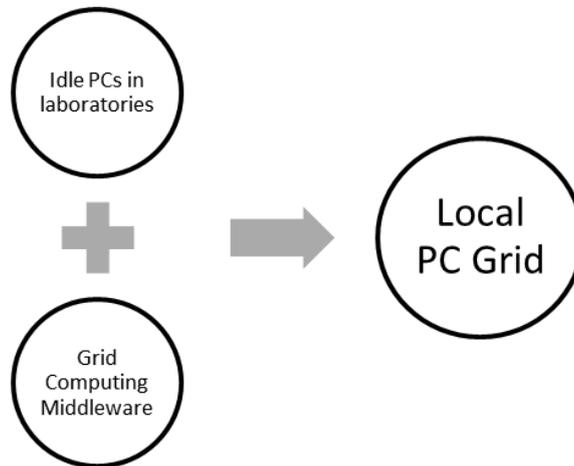


Fig. 1.1: Thesis description

Fig. (1.1) illustrates the expected goal of the research.

1.2. Motivation

This research is motivated by the advances being made in the field of Grid computing and Public Resource Computing and the advantages being derived by various disciplines through the adoption of Public Resource Computing technologies.

Palestinian HEI cannot afford supercomputers, but still a huge computational power is needed (e.g. data analysis, simulation, research, or visualization). A solution is found inside the walls of most of HEI; large number of Personal Computers (PCs) - especially computer laboratory PCs - are idle most of the time, so it will be beneficial to build a local PC grid and deploy such heavy computations on it without the need to purchase new hardware or equipment. This will not only serve student research goals, it will also serve managerial software run by the institution.

1.3. Problem Description and Approach

QOU contains 114 computer labs (Computer, Internet, Multipurpose (multimedia and eLearning), continuing education, and Information and Communication Technology laboratories (ICT)). These labs contain nearly 1850 PCs (about 3550 CPU cores) running Microsoft Windows OS (ICTC, 2013), for detailed information about QOU labs please refer to Appendices A and B.

These PCs are upgraded and maintained continuously, but yet their utilization is not full.

The idea is to build a virtual HPC out of these wasted resources in a way that sets expenses to minimum. The best approach is to build a Local PC Grid using a grid middleware.

This middleware must meet the following requirements:

1. Open source: in order to set costs to minimum.
2. Runs under Microsoft Windows OS; the standard platform in QOU computer labs.
3. Easy to deploy and maintain under Active Directory Environment.
4. Secure: data and authentication information must be encrypted
5. Scalable: number of involved PCs can increase or decrease without need to reinstall or reconfigure the grid.
6. Centrally managed and monitored by the institution.
7. Reliable: can perform requested task with least errors.
8. Supports legacy applications: little or no change in application source code
9. Green solution: power consumption must be set to minimum.

1.4. Limitations of the study

There are some limitations that need to be acknowledged and addressed regarding the present study:

1. Study concentrates on measuring computational resources available in CPUs. Measuring Graphical Processing Unit (GPU) computational resources will not be addressed since it is beyond the scope of this research.
2. Study considers computers available in computer labs in Jerusalem Branch and Bethany Study Center (SC) at QOU only. This is a pilot study, and this number of PCs does not represent a real sample, but can provide as with an exploration of the power of local PC grid computing when a more QOU branches are included.
3. Study is limited to PCs available in computer laboratories and does not include servers or computers used by academic/administrative staff for security matters.
4. By computer laboratories we mean: Internet, Multipurpose (Multimedia and eLearning), Communications (ICT), and Continued Education Labs
5. Study is performed on these PCs during work hours (08:00 – 15:30) from Saturday to Wednesday only.
6. Experiments in this study are performed on PCs with Microsoft Windows OS (XP Professional and x86 Windows 7 Professional x86) only.

1.5. Challenges

Major challenge was building the BOINC server (refer to chapter four) since it needs great knowledge in Linux administration and related services such as MySQL, Apache server, and PHP.

1.6. Contributions

To summarize, the researcher's main contributions are:

1. Revealing actual CPU utilization in computer lab PCs.
2. Providing a description of a grid-based infrastructure for aggregating idle computer laboratory PCs' resources in HEI and issues involved in building such a system.

3. Examining a local PC Grid middleware (Alchemi .NET) as a master-executer model for building PC grid computing.
4. Examining a PRC computing middleware (BOINC) to build a local PC Grid Computing environment.

1.7. Literature Survey

We have reviewed, studied and analyzed several references, journal papers and articles concerning grid computing, volunteer computing and local desktop grid. We have summed up the most important and relevant articles within our knowledge in this chapter of the thesis.

1.7.1. Availability of computer resources in Enterprise Desktop PCs

One of the earlier studies about the availability of computational resources in ordinary workstations was the one performed by Matt W. Mutka in 1992 at the University of Wisconsin. He traced the intervals of user activity and the idle intervals on 17 workstations for a 4-month period and stated that desktop PCs can be under-utilized by as much as 75% of the time (Mutka, 1992).

Also Mr. Acharya and co-researchers studied the performance of workstations available in workstation pools in Universities of Berkeley, Wisconsin, and Maryland. The number of workstations involved was 60, 300, 40, respectively in each “pool. Results showed that the average idle time for desktop machines ranged from 60% to 80% of the day (Acharya et al., 1997).

Another interesting related research was performed at University of Coimbra – Portugal. In this article, Domingues and co-researchers quantified the usage of main resources (CPU, main memory, disk space, and network bandwidth) of Windows 2000 machines from classroom laboratories. For that purpose, 169 machines of 11 classroom laboratories of an academic institution were monitored over 77 consecutive days. Samples were collected from all machines every 15 minutes for a total of 583653 samples. Results showed that resources idleness in classroom computers was very high; the average CPU idleness was 97.93%, unused memory averaging 42.06%, and unused disk space of the order of gigabytes per machine (Domingues et al., 2005). These results confirm the potentiality of computing resource for harvesting, especially for PC grid computing schemes.

Vlădoiu and co-researchers (Vlădoiu et al., 2009) represented their observations regarding the availability of computing resources to be used in a desktop grid. The results were from an undergraduate student laboratory of PCs. The researches were mostly concerned with the availability of computational resources during work hours, that is, from 08:00 till 20:00 hour during several days, when computers from labs are actually used intensively. From a rough estimate, the researchers noticed that computers are available for computations about 50-60% of the time during weekdays, between 08:00 and 20:00. The availability is close to 95-100% during the night. This amount reached to approximately 75-80% availability of computers during a 24 hours interval of a working day, growing to 90-95% during weekends. They concluded that a lot of computing power is available in university laboratories, which can easily be used for scientific experiments, provided that an appropriate resource-harvesting system is available (Vlădoiu et al., 2009).

One may notice that the researchers in the previous article were only concerned in the number of the available running PCs not in the percentage of idle CPU time for every PC in the laboratory.

1.7.2. Power of Local PC Grid

PC Grid Computing is divided into three main categories namely, open, closed and semi-open grids (Tachikawa, 2006). Tachikawa defined open grids as the grid that is comprised of PCs connected through the internet and owned by individuals who are willing to offer their PCs' idle processing power for free. Whereas, closed grids are constructed by business enterprises and other organizations, depending on their existing PCs that are connected through LANs. He also mentioned that semi-open grids are grids built by enterprises that offer their grid services for local businesses to jointly provide the local community with shared computing resources. He introduced a proposal for the construction of a semi-open grid out of the 1.5 million PCs found in Japanese elementary, junior and senior high schools (an average of about 30,000 PCs per prefecture). He estimated that the resulting grid would have a computing power of 3 Tera Floating Point Operations per Second (TeraFLOPS), which is beyond 500 GFLOPS - the minimum performance for a supercomputer at that time (2006), indicating that the proposed grid would provide CPU power comparable to a supercomputer (Tachikawa, 2006).

Tachikawa built his estimation on a fact that PCs produced in 2000 or later are expected to perform at 1 GFLOPS. Then he considered that older machines among the participating PCs may perform at about one tenth of this figure, that is, 100 MFLOPS. Assuming that these PCs are in an ideal state, the resulting grid in one prefecture would have a computing power of 100 MFLOPS \times 30,000 machines that gives 3 TFLOPS. This estimation is rough and neglects important factors such as network effect.

There are several middleware that can be deployed to build a local PC Grid computing environment such as Alchemi (Buyya et al., 2005), HTCondor (HTCondor, 2013), and Berkeley Open Infrastructure for Network Computing (BOINC) (Anderson, 2005). These software are briefly presented in chapter three. Several articles have described the construction and the benefits of local PC grid to perform heavy computations. These include:

- Agus Setiawan and his colleagues from the computer science Dept. at the University of Melbourne in Australia proposed and developed an application for symmetric key cryptography (called GridCrypt) using Alchemi framework (Buyya et al., 2005). They performed a runtime comparison for the GridCrypt application using 4 Executor nodes each with the same specification of Pentium 4/ 2400 MHz processor and 512 MB of memory and running Windows 2000 Professional operating system and interconnected over a shared student laboratory LAN network of 100 Mbps. The Alchemi manager was installed on a separate computer together with SQL Server 2000. A separate user machine was used to initiate the execution of the GridCrypt application. (Setiawa et al., 2004). They reported a reasonable performance improvement when 2 and 4 Executors were used roughly 120% and 130% respectively. (Setiawa et al., 2004)
- In their research, Trifa and colleagues (Trifa et al., 2011) employed volunteer computing technology (BOINC) to speed up the Arabic OCR process (Optical Character Recognition) based on the DTW (Dynamic Time Warping) algorithm.

During these experiments, they used 16 dedicated homogeneous workstations: 3.0 GHz CPU, 512 MB RAM and running Windows XP professional. They reported that as the number of workstations involved increases, the algorithm execution time decreases and the speedup factor increases. When all the 16 workstations were used, the execution time reached the value 1450 seconds and the speedup factor reached the value of 15. This produced a factor of more than 830 recognized characters per second competing with commercial Arabic OCR tools at that time. (Trifa et al., 2011).

The speedup factor that Trifa used was found by calculating the ratio of execution time for every level of the experiment with the execution time for one workstation.

- González and his co-authors (González et al., 2008) discussed the possibility of cancelling the need for user accounts to connect BOINC clients to projects by focusing on available resources. This allowed an institution to remotely manage all its BOINC enabled resources from a centralized point of view. They performed an experiment on 60 out of 3000 GNU/Linux PCs at three computer laboratories at distant campuses of the University of Extremadura - Spain for a period of 40 days. They observed the available resources during experiment period and built their estimation of computing power to be nearly 15.69 GFLOPS of CPU time. (González et al., 2008).
- Another example is the work of Zhou and colleagues (Zhou et al., 2009). They implemented an internal computing Grid inside the Geneva University Hospital (HUG); their objective was to provide computing power to researchers inside the HUG with a minimum cost and disturbance for the hospital system administrators. The testbed was performed on 20 standard desktop PCs with a 2.8GHz CPU and 768MB RAM. Linux was installed in a virtual machine on each PC, this virtual machine served as a Grid computing node, the back-end middleware in use for each computation node is Condor. Usability was evaluated by using an application that submitted 50 jobs. In each job 1000 images are treated and results are automatically collected. This application was executed on a dedicated server, a remote Grid resource (located in Finland) and the local Grid inside HUG. An overall time comparison is: 807min., 537min., and 240min. respectively. This result exhibits the improvement of the application efficiency using Grid technology.

1.7.3. The University of Westminster case study

The Centre for Parallel Computing at University of Westminster in UK built a local desktop grid system that consists of 1500 Windows-based laboratory PCs serving 20,000 students, and distributed over six campuses (UWM, 2012). Fig. (1.2) presents the system showing the various campuses.

The unused resources of laboratory PCs were employed to run computation-intensive tasks for university researchers such as powerful simulation programs used by bio-scientists, linking results together in a sophisticated workflow, to understand how molecules potentially bind together. Researchers using the Westminster Local Desktop Grid have found that they can shorten a typical execution time from weeks to hours. (Westminster, 2012). Another advantage of this approach is that the University no longer has to pay £500,000 every four years just to upgrade their supercomputer; the upgrade process of the local desktop grid is less expensive and happens automatically when upgrading the laboratory PCs themselves to pace with technology trends (Westminster, 2012).

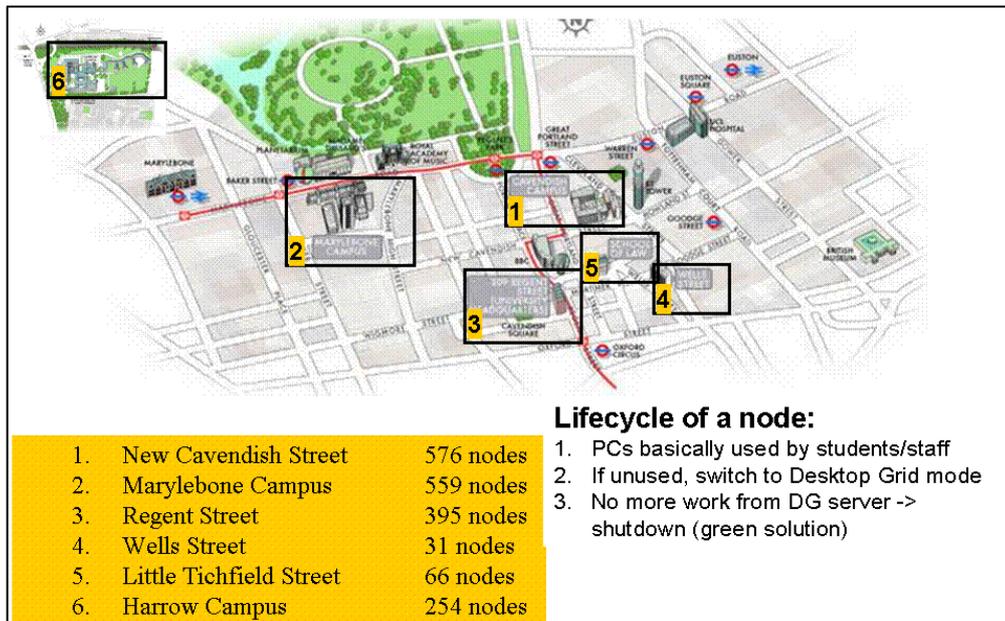


Fig. 1.2: Local PC Grid at the University of Westminster (UWM, 2012)

1.7.4. Literature Survey Summary

From the above survey one can notice the ample computing resources found within the boundaries of hospitals, universities, companies, or even secondary schools.

The researches performed on local PC grid were positive in the way that encourages building a grid computing environment in a Palestinian HEIs since supercomputers are not affordable.

Our approach is different from previous works by focusing on computer lab PCs available at Al-Quds Open University as an example of a Higher Education Institution. The research studies factors that affect the overall performance of the grid such as network traffic and operating system.

The performance metrics used include the execution time, speed up factor, the total GFLOPS, and the total credit (a credit is a description of the overall contribution of a certain PC in the grid).

1.8. Thesis organization

This thesis is presented in five chapters. The *first chapter* is an introductory chapter; motivation, problem description, limitations, challenges, and main contributions are presented, then reviews of previous research in the field of PC grid computing and PRC are surveyed.

Chapter two presents background information about concepts of grid computing, PC grid computing, PRC, and presents most common PC grid enabling frameworks.

Research methodology is represented in *chapter three*, QOU sample environment is described and computer lab statistics are provided.

While *chapter four* presents the experimental work and discusses results, it describes the implementation of Alchemi .NET and BOINC frameworks to build local PC grid, deploying sample application, discusses major results comparing them with other researches in field, provides a summary of the research.

Chapter five highlights how the aim and the objectives of this research have been met with suggestions for future work in the field.

Chapter Two

Background

2.1. Introduction

In this chapter an introduction to grid computing is introduced, grid computing is defined, and types and benefits of Grid computing are then discussed in general.

PC grid computing and PRC are explained, most common grid enabling frameworks are presented, and finally two examples are discussed in detail: Alchemi as an example for an Enterprise grid middleware and BOINC as an example of PRC framework.

2.2. Grid Computing

The term *Grid* or *Grid computing* takes its name from an analogy with the electrical "power grid". It was developed in the mid 1990's with the growth of high speed networks and the Internet that allowed distributed computer systems to be readily interconnected. Grid computing has become one of the most important techniques in high performance computing (HPC) by providing resource sharing in science, technology, engineering, and business. By taking advantage of the Internet and high speed networks, geographically distributed computers can be used collectively for collaborative problem solving.

Grid Computing is a general term that combines different technologies and solutions to different target groups. Concepts associated with the term range from cluster computing, High Performance Computing (HPC), utility computing, peer-to-peer (P2P) computing, Public Resource Computing (PRC) to specific new types of infrastructure.

2.3. Definition

In the Information technology literature, various definitions of Grid Computing are noted; these definitions encounter modifications and/or broadening over the years according to new emerging technologies. The first and most cited definition of Grid Computing was suggested by Foster and Kesselman (1998):

"A computational grid is a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities".

The above definition has been modified twice by the grid experts; once by Foster, Kesselman, and Tuecke in their paper entitled "Anatomy of the Grid" (Foster et al., 2001), and again by Foster and Kesselman in the second edition of their book "*The Grid2: The Blueprint for a New Computing Infrastructure*" (Foster and Kesselman, 2004).

The definition was refined in order to address social and policy issues, stating that Grid computing is concerned with “*coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations.*” (Foster et al., 2001)

According to Foster (2002), a Grid system is therefore a system that:

- Coordinates resources that are not subject to centralized control
- Uses standard, open, general-purpose protocols and interfaces
- Delivers nontrivial qualities of service (QoS).

From the above argument, we can say that grid computing is the collaboration of distributed interconnected computers and resources to achieve common higher performance computing and resource sharing.

The term *resources* that can be shared on the grid means:

- Processing power or computing cycles: A network of distributed high performance computers working like a single huge computer.
- Data storage: making a grid of disk devices and file systems that is remotely accessible through the network and works like a large external storage device.
- Communications
- Application software
- Scientific instruments: group of distributed and networked sensors from which data can be collected for specific purposes, such as global environmental monitoring system.

2.4. Classification of Grids according to scope of resource sharing

Depending on the scope of resource sharing involved, the following Grid Computing approaches can be distinguished (Magoulès et al., 2009):

1. Cluster Grids
2. Enterprise Grids
3. Utility Grid Services
4. Partner/Community Grids

These four different types of Grids are explained in more detail below.

2.4.1. Cluster Grids

Cluster Grids, or clusters, are a collection of co-located computers connected by a high-speed local area network and designed to be used as an integrated computing or data processing resource as shown in Fig. (2.1).

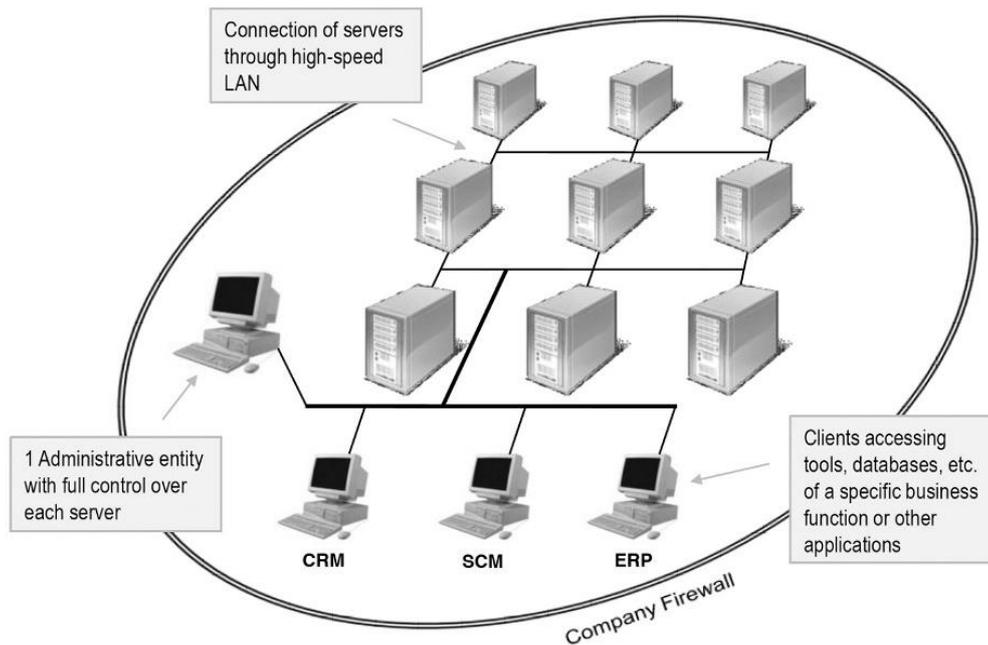


Fig. 2.1: A typical form of cluster computing (Stanoevska et al., 2010)

A cluster is a homogeneous entity. Its components differ primarily in configuration, not basic architecture. Cluster Grids are local resources that operate inside the firewall and are controlled by a single administrative entity that has complete control over each component (Foster and Kesselman, 1998).

2.4.2. Enterprise Grid

The term Enterprise Grid is used to refer to application of Grid Computing for sharing resources within the bounds of a single company (Goyal and Lawande, 2005). All components of an Enterprise Grid operate inside the firewall of a company, but may be heterogeneous and physically distributed across multiple company locations or sites and may belong to different administrative domains as illustrated in Fig. (2.2).

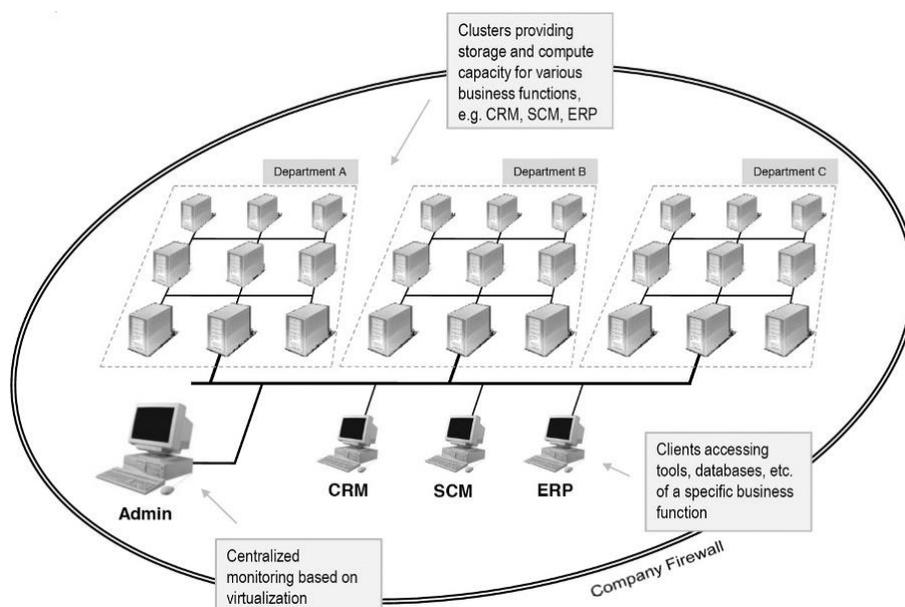


Fig. 2.2: Example Enterprise Grid infrastructure (Stanoevska et al., 2010)

With specific Enterprise Grid middleware, the available IT resources are virtualized and can be managed in a unified and central way. They can also be allocated to processes according to demand.

2.4.3. Utility Grid

A Grid that is owned and deployed by a third party service provider is called a Utility Grid. The service being offered via a Utility Grid is utility computing, i.e. compute capacity and/or storage in a pay-per-use manner. A Utility Grid operates outside the firewall of the user as depicted in Fig. (2.3).

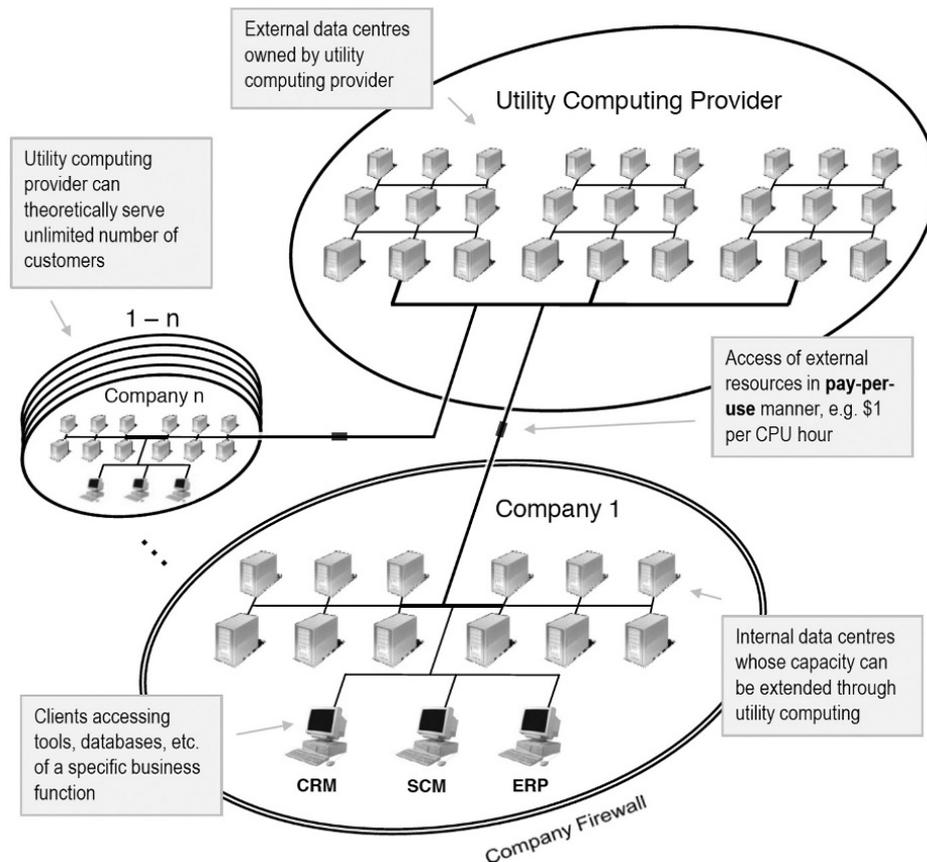


Fig. 2.3: Utility Grid architecture (Stanoevska et al, 2010)

The user does not own the Utility Grid and does not have control over its operation. Utility Computing furthermore provides scalability and flexibility of IT resources on demand.

2.4.4. Partner/Community Grids

Partner or Community Grids are a specific type of Grids that can provide support for the establishment of a VO based on IT resource sharing among collaborating entities.

In a Partner/Community Grid each participating partner provides a certain part of its infrastructure for sharing and either defines the rules under which the resources can be used by other partners or accepts the community rules for resource donation.

The architecture of a Partner/Community Grid can be viewed as a collection of independent resources (for example Cluster Grids or other resources) interconnected

through a global Grid middleware, and accessible, optionally, through a portal interface as illustrated in Fig. (2.4).

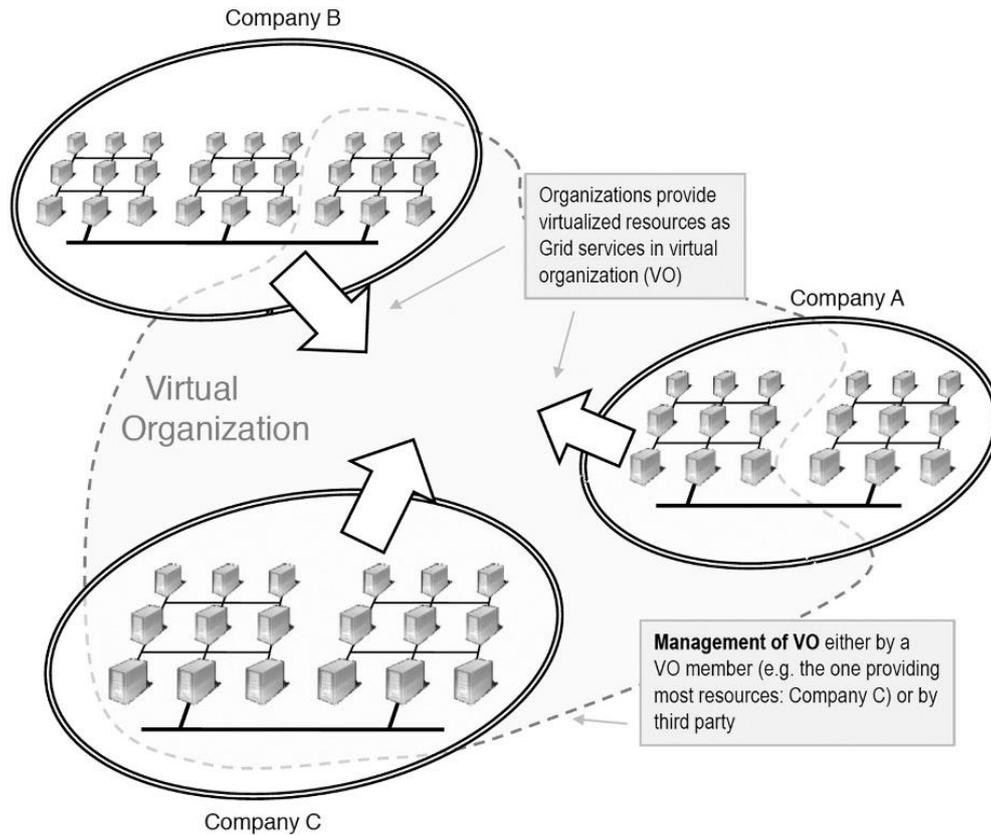


Fig. 2.4: Example of a partner grid (Stanoevska et al, 2010)

2.5. Grid Architecture

Grid architecture refers to those aspects of a grid system that are taken into consideration when a grid is designed and implemented. Grid architecture is a layered architecture, these layers are:

- Application layer: consists of grid applications and the Application Programming interfaces (APIs) from a user's perspective.
- The middleware layer: which includes the software and packages used for grid implementation, for example Alchemi, BOINC.
- Resource layer: covers the resources available to the grid such as storage, processing capabilities and other application-specific hardware.
- The network layer: which deals with the network components like routers, switches, and the protocols used for communication between any two systems in the grid.

Fig. (2.5) below illustrates the common Grid architecture.

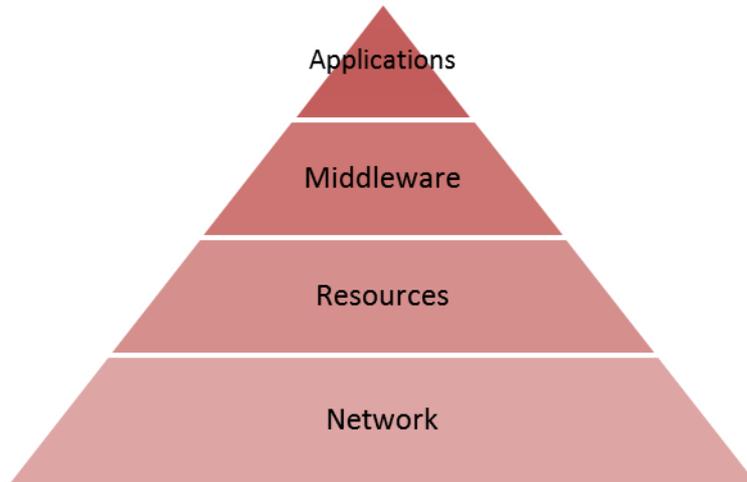


Fig. 2.5: Common Grid Architecture (adapted from Stanoevska et al, 2010)

2.6. PC Grid Computing

Grid computing meets the needs of virtual organizations by sharing resources of centrally-managed, supercomputers, HPCs, or clusters. Whereas; PC grid computing (or Desktop Grid computing) addresses the potential of harvesting idle computing resources of distributed, ordinary desktop PCs (in a company, a university, at home, etc.). These resources can be part of the same local area network (LAN) in the case of enterprises, or can be geographically spread and connected via a wide area network (WAN) such as the Internet for public resource computing (or volunteer computing).

Given the number of desktop computers across the world, this represents an enormous computing resource. The immediate implication of this is that software applications can potentially run substantially faster. In enterprises, this also means that the potential of enterprise computing resources can also be potentially increased.

PC Grid Computing can be categorized according to scale, platform, organization, and resource provider (Choi S. et al., 2007) as shown in Fig. (2.6).

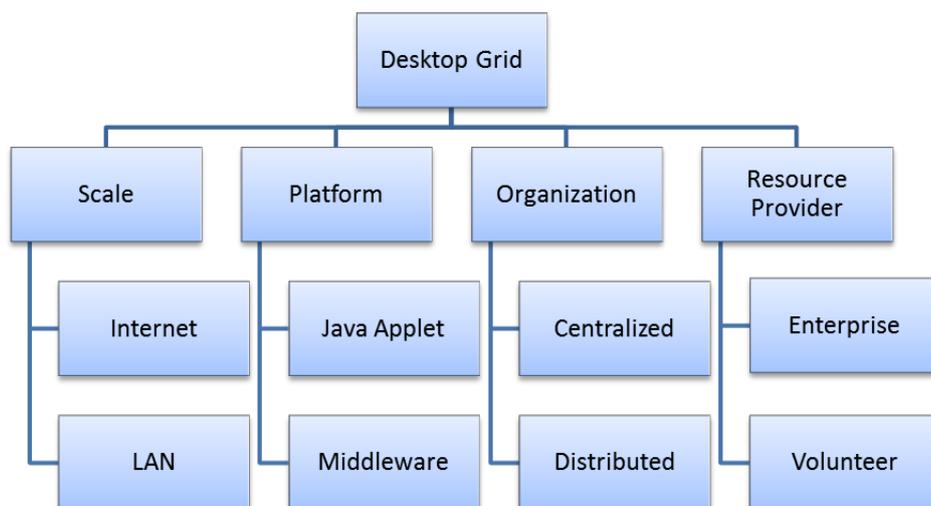


Fig. 2.6: PC Grid Computing classification (Choi S. et al., 2007)

Fig. (2.6) shows a classification of PC grid computing systems; in this research we are concerned with PC Grid computing in HEI namely, Al-Quds Open University-Jerusalem Branch, meeting the requirements presented in Chapter one, so from the above categorization we are interested in:

- Enterprise: since we are building a grid for the HEI not for public volunteers.
- Centralized: the designed grid must be monitored and administered centrally by the HEI.
- Middleware: a Java applet is a small application that is written in Java and launched from a webpage in a browser. Java applet grid is not recommended in this research since we need to restrict the students' access to the grid middleware, and this cannot be achieved with java applets. But a middleware that runs as a service at computer startup with suitable permissions cannot be controlled by unprivileged student.
- LAN: in order to reduce information security penetration, the PC grid must run inside the HEI firewall not on the Internet.

2.7. Public Resource Computing

Public Resource Computing (PRC) (also referred to as “Volunteer Computing” or “Internet computing”) is a form of high performance computing (HPC) in which volunteers from around the world donate a portion of their computers' resources to a computationally intensive research project(s) (Baldassari, 2006). Researchers who do not have access to supercomputing facilities can use PRC to increase their ability to process compute-intensive data at little to no extra cost.

There are several PRC frameworks that facilitate the creation of PRC projects by managing many of the responsibilities of the client and server. One of the most widely used PRC frameworks is the *Berkeley Open Infrastructure for Network Computing (BOINC)* (BOINC, 2012). The advantage of using BOINC is that it abstracts much of the complexity involved in creating and maintaining a large PRC project.

BOINC manages all network communications between the clients and the server, so there is no need for a PRC project developer to write any network code. BOINC manages the project database and connections to it, so developers do not need to write any database connection code or SQL queries.

BOINC handles the distribution of work units and collection of results; it can recover from situations in which clients receive a work unit, but never return a result or return an erroneous result. The BOINC is designed to be highly scalable and the system can currently support on the order of tens of millions of client requests per day (Anderson et al., 2005). BOINC structure will be discussed in the following sections.

There are several differences between Grid computing and PC Grid Computing (PRC and Local PC Grid), in several aspects, these differences are summarized the Table (2.1)

Table 2.1: PRC, Local PC Grid , and Grid comparison (Choi S. et al., 2007)

Item	PC Grid		Grid
	PRC	Local PC Grid	
Resource	Desktop <ul style="list-style-type: none"> Anonymous resource provider 	Desktop <ul style="list-style-type: none"> within a corporation, university, etc. 	Supercomputer, cluster, scientific instruments, database, storage,
Connection	<ul style="list-style-type: none"> Non dedicated poor bandwidth Immediate presence (connectivity) Consider firewall, NAT, Dynamic address 	<ul style="list-style-type: none"> Non dedicated intermediate bandwidth More constant connectivity than PRC 	Dedicated high speed, bandwidth
Heterogeneity	High heterogeneous <ul style="list-style-type: none"> Need resource grouping 	<ul style="list-style-type: none"> Intermediate heterogeneous Less heterogeneous than PRC 	Low heterogeneous
Dedication	<ul style="list-style-type: none"> Non-dedicated High volatile <ul style="list-style-type: none"> Need an incentive mechanism 	<ul style="list-style-type: none"> Non-dedicated Low volatile (non-business hours) <ul style="list-style-type: none"> Need an incentive mechanism 	Dedicated <ul style="list-style-type: none"> Be able to use reservation
Trust	Malicious volunteer <ul style="list-style-type: none"> Need result certification 	Low trustworthy resource provider	High trustworthy resource provider
Failure	Unreliable (faulty)	Unreliable <ul style="list-style-type: none"> More reliable than PRC 	More reliable than desktop grid
Manageability	Individual-based administration <ul style="list-style-type: none"> Totally distributed to individual Difficult to manage 	Individual-based administration <ul style="list-style-type: none"> More controllable than PRC 	Domain-based administration <ul style="list-style-type: none"> Professional administrator
Application (job)	<ul style="list-style-type: none"> Independent (mainly) Computation-intensive (mainly) High-throughput (mainly) 	<ul style="list-style-type: none"> Independent (mainly) Computation-intensive (mainly) <ul style="list-style-type: none"> Data-intensive (possible) High throughput (mainly) 	<ul style="list-style-type: none"> Independent or dependent Computation or data-intensive High performance (mainly)

2.8. Grid Middleware

The most common PC grid middleware packages include: Entropia (Chien et al., 2003), distributed.net (distributed.net, 2013), Grid MP (UNIVA, 2013), XtremWeb (Germain et al., 2000), HTCondor (HTCondor, 2013), Alchemi .NET (Buyya et al., 2005), and BOINC (BOINC, 2012).

These middlewares are briefly presented in the following sections.

2.8.1. Entropia

Entropia is a commercial product that is sold as part of Entropia's PC Grid enterprise solution developed in 2001 (Chien et al., 2003). Entropia facilitates a Windows desktop grid system by aggregating desktop resources into a single logical resource. It is based on a centralized architecture in which a central job manager administers various desktop clients.

The node manager provides a centralized interface to manage all of the clients on the Entropia grid, which is accessible from anywhere on the enterprise network.

In 2004, Entropia Company ceased commercial operations, and its original website (<http://www.entropia.com/>) is currently unavailable (Wikipedia, 2013).

2.8.2. Grid MP

The Grid MP is a commercial distributed computing software package that is developed and sold by UNIVA (formerly known as United Devices) (UNIVA, 2013). It also supports harnessing and aggregation compute resources available on corporate networks. It basically has a centralized architecture, where a Grid MP Service acting as a manager accepts jobs from the user, schedules them on the resources having pre-deployed Grid MP agents. The Grid MP agents can be deployed on clusters, workstations or desktop computers. Grid MP agents receive jobs and execute them on resources, advertise their resource capabilities on Grid MP services and return results to the Grid MP services for subsequent collection by the user.

2.8.3. XtremWeb

XtremWeb is a Peer-to-Peer, JAVA-based volunteer computing system developed at the University of Paris-Sud, France (Germain et al., 2000). It implements three distinct entities, the coordinator, the workers and the clients, to create a so-called XtremWeb network. *Clients* are software instances available for any user to submit tasks to the XtremWeb network. They submit tasks to the *coordinator*, providing binaries and optional parameter files and permit the end user to retrieve results. *Workers* are software parts spread among volunteer hosts to compute tasks.

Popular XtremWeb deployments are XWHEP: XtremWeb for High Energy Physics (Xtremweb-HEP, 2013), and XtremWeb-CH project - which can be considered as the updated version of the original XtremWeb, aims at building an effective Peer-To-Peer system for high performance computing, based on a completely symmetric model where nodes are both providers and consumers. (Xtremweb-CH, 2013) (Bellavista et al., 2010)

A peer-to-peer solution is not a suitable solution in the case of HEI; project submission and administration must be through the HEI only.

2.8.4. distributed.net

A very similar framework is the distributed.net project. It is a volunteer computing middleware that attracts participants to help in two main projects:

- RC5: that aims to decipher encrypted messages submitted from RSA Company as a puzzle with cash prizes for those who solve them. The company's aim is to test the security of their own products and to demonstrate the vulnerability of encryption schemes they consider inadequate (distributed.net, 2013).
- OGR (Optimal Golomb Ruler). A Golomb Ruler is a mathematical term given to a set of whole numbers where no two pairs of numbers have the same difference. An OGR is a regular ruler, except that the marks are placed so that no two pairs of marks measure the same distance. The search for OGRs becomes exponentially

more difficult as the number of marks increases, and this required the help of volunteers from the globe. (distributed.net, 2013)

The focus of the distributed.net project is on these two specialized computing challenges; the server code cannot be obtained and thus used for any other projects or applications.

2.8.5. HTCCondor

HTCondor (formerly known as Condor till 2012) is a High Throughput Computing (HTC) environment developed at the department of Computer Science, University of Wisconsin, Madison, USA (HTCondor, 2013).

HTCondor provides a job queuing mechanism, scheduling policy, priority scheme, resource monitoring, and resource management. Users submit their serial or parallel jobs to HTCondor, HTCondor places them into a queue, chooses when and where to run the jobs based upon a policy, carefully monitors their progress, and ultimately informs the user upon completion (HTCondor, 2013)

2.8.6. Alchemi.NET

Alchemi .NET is designed basically to build an Enterprise PC Grid, but it also supports PCR computing. Alchemi is built as layered architecture as shown in Fig. (2.7). Alchemi follows the master-worker parallel programming paradigm (Kruskal and Weis, 1948) in which a central component dispatches independent units of parallel execution to workers and manages them. In Alchemi, this unit of parallel execution is termed ‘grid thread’ and contains the instructions to be executed on a grid node, while the central component is termed ‘Manager’ (Buyya et al., 2005).

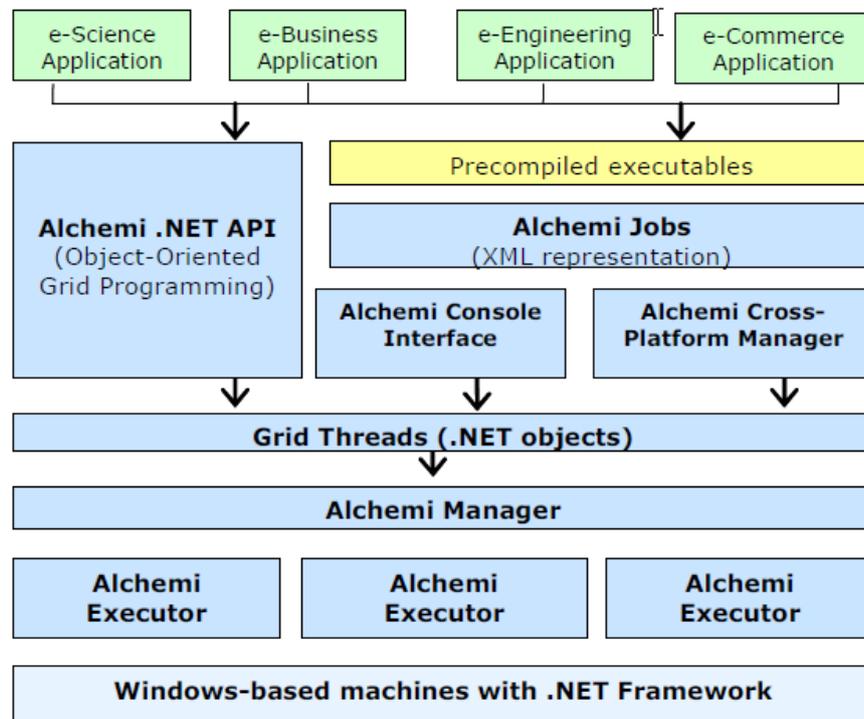


Fig. 2.7 A layered architecture for Alchemi framework (Buyya et al., 2005).

Fig. (2.7) represents the different layers of Alchemi framework, a ‘grid application’ consists of a number of related grid threads. Grid applications and grid threads are exposed to the application developer as .NET classes/objects via the Alchemi .NET API. When an application written using this API is executed, grid thread objects are submitted to the Alchemi Manager for execution by the grid. Alternatively, file-based jobs (with related jobs comprising a task) can be created using an XML representation to grid-enable legacy applications for which precompiled executables exist. Jobs can be submitted via Alchemi Console Interface or Cross-Platform Manager web service interface, which in turn convert them into the grid threads before submitting them to the Manager for execution by the grid.

Programming environment of Alchemi is object oriented .NET API as the name implies, this enables wiring grid applications in any .Net – supported programming language.

The atomic unit of independent parallel execution is grid thread with many grid threads comprising a grid application (Buyya et al., 2005).

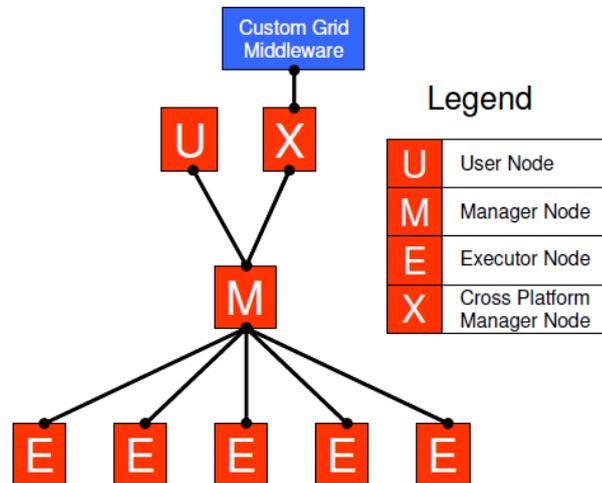


Fig. 2.8: Distributed components and their relations (Buyya et al., 2005)

Fig (2.8) shows the distributed components of Alchemi: manager, executor, cross platform, and user nodes.

Alchemi became licensed after 2006. The new version of Alchemi is called Aneka which is a commercial package for Private cloud/grid enabling (ManjraSoft, 2012).

We e-mailed Dr. Rajkumar Buyya, the developer of Alchemi and Aneka, and CEO of Manjrasoft Ltd., requesting pricing of Aneka licensing in order to get a full feasibility study for the purpose of this research, Mr. Buyya’s offer is shown in Appendix B.

Despite this fact, Alchemi .NET is still functional and well documented; it can still be deployed to construct a local desktop grid.

2.8.7. BOINC

Berkeley Open Interface for Network Computing (BOINC) is an open source middleware platform for PRC. It was developed in February 2002 by U.C. Berkeley Spaces Sciences Laboratory by the group that developed and continues to operate SETI@home lead by Dr.

David Anderson (BOINC, 2012). SETI@HOME tries to find extraterrestrial life by analyzing radio signals (SETI, 2013).

BOINC is designed to be a free structure for anyone wishing to start a volunteer computing project. It is a set of software modules that enable the use of idle CPU cycles on a personal computer to do scientific computing.

BOINC is the most popular DG system today with the aggregated computational power of more than 2,576,332 participants is about 8,361.840 TeraFLOPS, thus providing one of the most powerful “supercomputers” in the world (BOINCstats, 2013) (TOP500, 2013).

The structure of BOINC is simple. BOINC follows the client-server architecture; the server generates work units (WU), distributes them to clients and collects their results. Each PC, acting as a client, communicates with the server to get WUs which include executables and input files and return results of computation, BOINC is not peer-to-peer; the clients do not communicate with each other (Anderson, 2005).

These components are shown in Fig. (2.9) and are introduced in detail below.

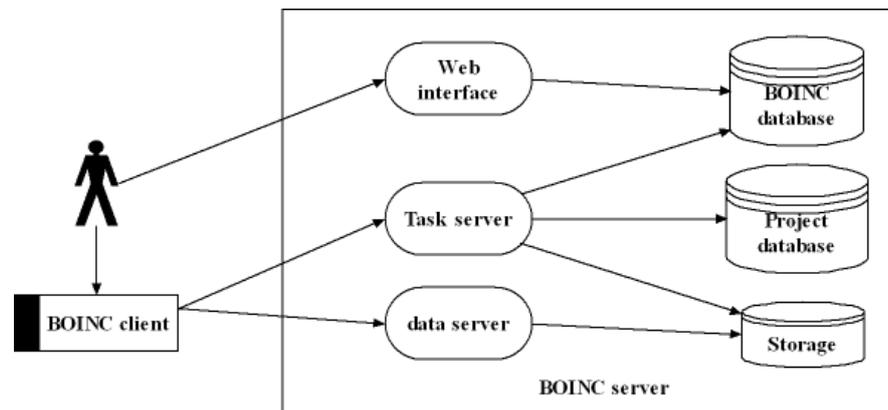


Fig. 2.9 : BOINC architecture (adapted from Anderson et al., 2005)

Fig. (2.9) describes BOINC architecture showing server and client components

2.8.7.1. BOINC server

BOINC server runs on Linux and uses Apache, PHP, and MySQL as a basis for its web and database systems, which easily scales to projects of any size.

BOINC server consists of the following components:

- **Web interfaces:** uses Apache web server and PHP, for user account and team management, message boards, current server status, and other features.
- **The task server:** creates tasks, posts them to clients, and processes returned tasks.
- **The data server:** downloads input files and executables, and uploads output file, uses HTTP protocol.

These components share various data stored on disk, including relational databases and upload/download files.

The most important server is the task server. BOINC daemons periodically check the state of the database and perform any needed tasks within their area of responsibility

Below we go into details of these daemons and a few other components that construct the BOINC server.

2.8.7.2. Task server components

Referring to the article entitled: “High-Performance Task Distribution for Volunteer Computing” (Anderson et al., 2005). Task server components are described in Fig. (2.10).

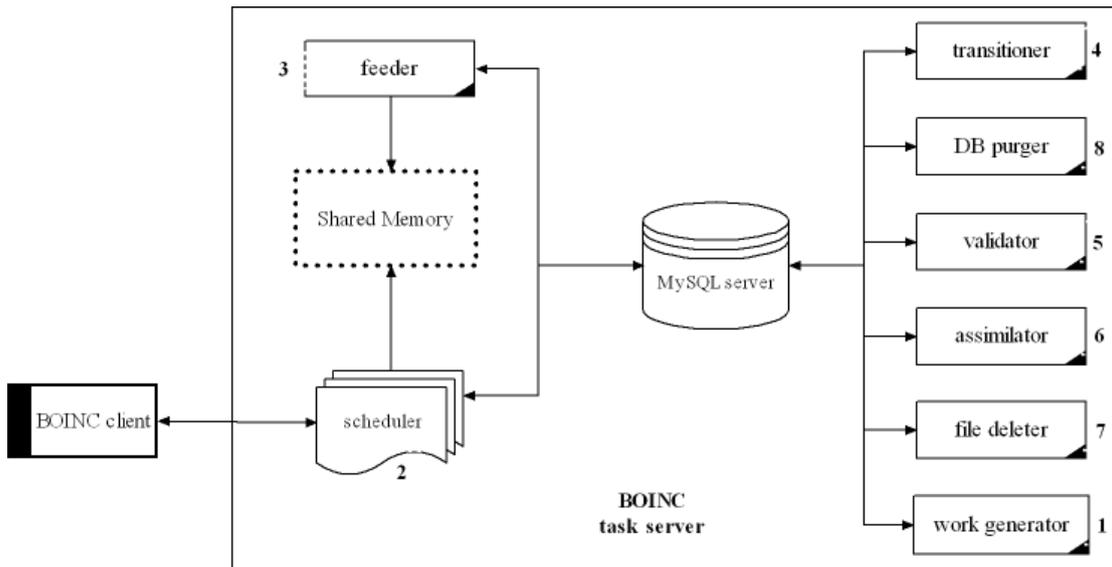


Fig 2.10: BOINC task server components (adapted from Anderson et al., 2005)

BOINC daemons are described in Fig. (2.11) and discussed below.

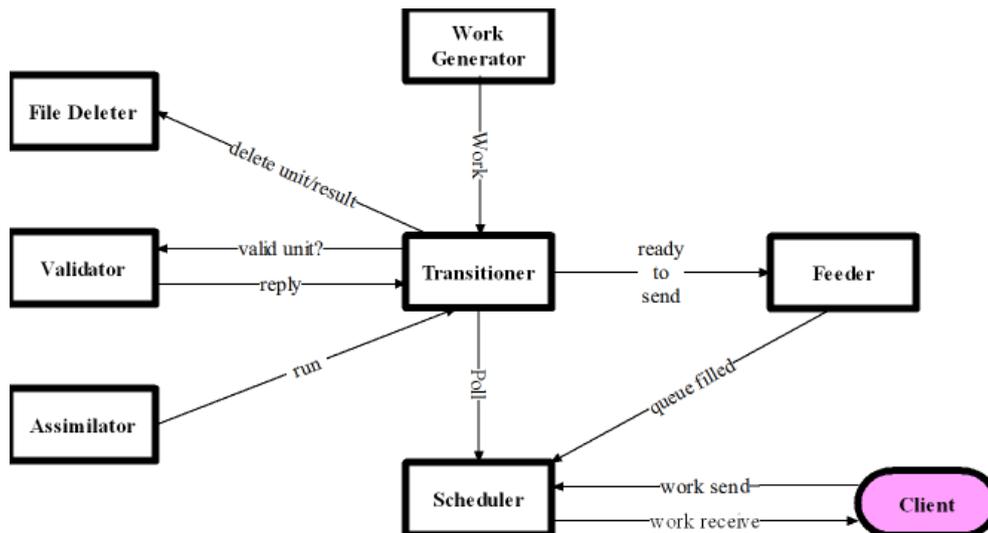


Fig. 2.11: BOINC daemons and components (adapted from Anderson et al., 2005)

1. The **work generator** creates new jobs and their input files. For example, the SETI@home work generator reads digital tapes containing data from a radio telescope, divides this data into files, and creates jobs in the BOINC database.
2. The **scheduler** handles requests from BOINC clients. Each request includes a description of the host, a list of completed instances, and a request for additional work, expressed in terms of the time the work should take to complete. The reply includes a list of instances and their corresponding jobs.
3. The **feeder** restructures the scheduler's database access. It maintains a shared-memory segment containing static database tables such as applications, platforms, and application versions, and a fixed-size cache of unsent instance/job pairs. The scheduler finds instances that can be sent to a particular client by scanning this memory segment. A semaphore synchronizes access to the shared-memory segment.
4. The **transitioner** examines jobs for which a state change has occurred (e.g., a completed instance has been reported). Depending on the situation, it may generate new instances; flag the job as having a permanent error, or trigger validation or assimilation of the job.
5. The **validator** compares the instances of a job and selects a canonical instance representing the correct output. It determines the credit granted to users and hosts that return the correct output, and updates those database records.
6. The **assimilator** handles job that are "completed": i.e., that have a canonical instance or for which a permanent error has occurred. Handling a successfully completed job might involve writing outputs to an application database or archiving the output files.
7. The **file deleter** deletes input and output files that are no longer needed.
8. The **database purger** removes jobs and instance database entries that are no longer needed, first writing them to XML log files. This bounds the size of these tables, so that they act as a working set rather than an archive. This allows database management operations (such as backups) to be done quickly.

2.8.7.3. BOINC client

Four components construct the BOINC client: core client, manager, screensaver, and command line tool.

These components are shown in Fig. (2.12) and presented below.

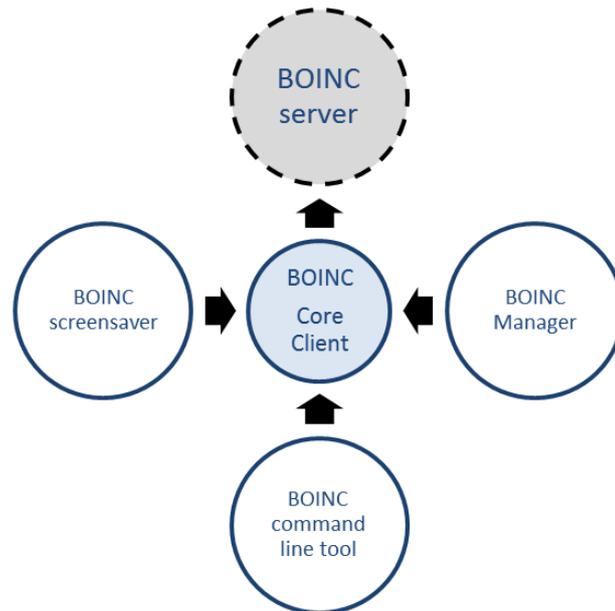


Fig. 2.12: BOINC client overview

1. The core client takes care of scheduling among jobs from different projects, possibly preempting jobs, downloading and uploading results to the different projects it is attached to (BOINC, 2012).
2. BOINC manager enables control over the core client via user preferences. Some of these preferences are project specific and some are not. As a project specific preference the user can set a minimum and a maximum amount of work the client should keep on the computer for the given project. (BOINC, 2012)
3. The screensaver displays application specific graphics as a screensaver just to attract and entertain volunteer users in the first place. (BOINC, 2012)
4. BOINC command tools provide non-GUI version of BOINC manager. (BOINC,2012)

The client is available for Windows and Linux on Intel x86 architectures, for Mac OS X on PowerPC, and Solaris on Sparc architectures, but since it is open source it should not be too difficult to get it running on other platforms as well.

BOINC client empowers users with plenty of configuration options to choose from, including the network bandwidth the client is allowed to use, number of CPUs, time schedule, and percentage of CPU time available to BOINC tasks. To further reward users for their perceived disadvantage, the project team is encouraged to publish daily statistics with top contributing users, countries, and teams. But mainly, participants donate their computing power for they are genuinely interested in scientific research and feel good about being able to take part and make contribution to humanity.

2.8.7.4. The Database

A MySQL database stores all information relevant to the BOINC server complex. This includes information about registered users and their associated hosts, about applications

and application versions, about BOINC core clients and the versions involved and of course about WUs and their associated results. Basically the entire state of the server is stored in this database and queried by among others the above mentioned daemons (BOINC, 2012).

2.9. Summary

In this Chapter, we introduced the Grid computing technology in general, and discussed the PC grid computing and Public Resource Computing.

Common PC Grid enabling middleware are then presented focusing on design and features of Alchemi .NET and BOINC.

Chapter three presents the research methodology.

Chapter Three

Methodology

3.1. Description of experiment environment

All experiments in this research were performed in Jerusalem Branch and Bethany Study Center (SC) at QOU. These two sites were selected as pilot study sample of QOU branches.

Although this pilot study sample does not represent a real sample, which must be at least 20% of study community, but it can be used for exploring the power of a local PC grid in QOU. More accurate results can be gathered when scaling up the testbed to include most QOU branches and study centers.

Jerusalem Branch includes three labs: Computer, Internet and Multipurpose (multimedia + eLearning) labs. Bethany SC includes one lab which is used for a dual purpose; as a computer and an Internet lab. All PCs in both sites are interconnected with local area network (LAN) of 100Mbps speed.

Jerusalem Branch site is not connected directly to Bethany site; it is connected to Information & Communication Technology Center (ICTC) building in Ramallah (IPVPN 2 Mbps line), Bethany Study Center site is connected to ICTC (6 Mbps IPVPN line) as illustrated in Fig (3.1). More information about QOU current setup is presented in Appendices A and B.

The two locations fall into different subnets; subnet in Jerusalem branch is 10.12.x.x and in Bethany SC 10.13.x.x

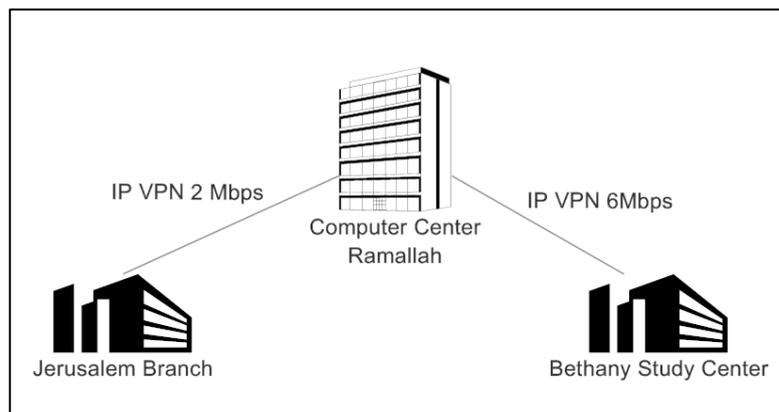


Fig. 3.1: Jerusalem and Bethany SC sites

Total number of PCs = 32 (in Jerusalem Branch) + 11 (in Bethany SC) = 43 PCs.

More details about specification of these PCs are provided in Table (3.1).

Table 3.1: Specifications of experiment PCs

	Lab	Sum	PCs	RAM (GB)	CPU description	CPU cores	OS
Jerusalem Branch	Computer	16	9	4.0	Intel Core i3-3.3 GHz	18	Microsoft Windows 7 Professional x86
			3	2.0	Intel Core 2 Due - 3.3 GHz	6	Microsoft Windows 7 Professional x86
			2	2.0	Intel Core 2 Due - 2.33 GHz	4	Microsoft Windows 7 Professional x86
			1	2.0	Intel Core 2 Due - 2.13 GHz	2	Microsoft Windows 7 Professional x86
			1	0.75	Intel Pentium 4- 2.4 GHz	1	Microsoft Windows 7 Professional x86
	Multipurpose (multimedia + eLearning)	8	8	1.0	Intel Core 2 Due 2.33 GHz	16	Microsoft Windows XP Professional x86 SP3
	Internet	8	7	3.0	Intel Pentium 4- 3.0 GHz	7	Microsoft Windows XP Professional x86 SP3
1			0.5	Intel Pentium 4- 3.2 GHz	1	Microsoft Windows XP Professional x86 SP3	
Bethany Study Center	Computer/ Internet	11	5	0.5	Intel Pentium 4- 3.2 GHz	5	Microsoft Windows XP Professional x86 SP2
			1	1.0	Intel Pentium 4- 3.2 GHz	1	Microsoft Windows XP Professional x86 SP3
			1	1.0	Intel Core 2 Due - 3.3 GHz	2	Microsoft Windows XP Professional x86 SP2
			1	1.0	Intel Dual-Core - 2.5 GHz	2	Microsoft Windows XP Professional x86 SP3
			3	4.0	Intel Core i3- 3.3 GHz	6	Microsoft Windows 7 Professional x86 SP3
Total number of PCs		43	Total Cores			71	

The number of CPU cores described in Table (3.1) above was obtained from CPU datasheets provided by CPU manufacturer (Intel) website. (Intel, 2013)

The experiments were performed during work hours (08:00 – 15:30), from Saturday to Wednesday only. Lab PCs are turned on at 08:00, students can freely log in to any vacant PC and start working, but they are prohibited from shutting down the used PC; they have to leave it on so that the next student doesn't have to wait for PC boot-up process.

During the experiments, we made sure that grid middleware (Alchemi or BOINC) runs in the background while the student is using the PC.

3.2. Description of experiments

Three major experiments are performed:

1. Measuring CPU utilization
2. Examining Alchemi .NET
3. Examining BOINC

3.2.1. Measuring CPU utilization

Although previous researches showed that CPUs are underutilized; we need to reveal the actual CPU utilization in computer labs and present this fact in figures. For this purpose, we used a tool called CPU Usage Logger (CPU logger, 2012), this tool was set to run on computer startup. It then automatically begins recording average CPU utilization each 5 seconds in a plain text log file saved on the local hard drive.

The log files are then imported and analyzed using Microsoft Excel 2007.

3.2.2. Examining Alchemi .NET

Testing Alchemi .NET is performed by installing Alchemi Manager on one PC, and then installing Alchemi Executer on 8 PCs. Executors are to be attached manually to the Manager by entering correct settings. Finally a grid application is to be run in the manager and the overall performance is observed.

Alchemi Manager and Executors are to be installed on identical PCs; they have the same hardware components: Intel Core 2 Due, 2.33 GHz processor /1.0GB RAM / Microsoft Windows XP Professional.

The test application is the computation of the value of PI to n decimal digits. The algorithm used allows the computation of the p^{th} digit without knowing the previous digits. (Bellard, 2013)

There are several example applications that come with the Alchemi Software Development Kit (SDK) that can be used to test the grid performance such as: Prime Number Generator, Alchemi Renderer, and Mandelbrot (Alchemi, 2012), but the PI calculator is mostly used in Alchemi testing (Buyya, 2005), so it was used here in order to compare obtained results with those of other researches.

Execution time is adopted as the performance metric here; the PI calculator example calculates and shows the execution time after each trail.

3.2.3. Examining BOINC

In this part of the experiment, we adopted **Giga Floating Point Operations Per Second (GFLOPS)** as the performance metric of the BOINC grid. GFLOPS value is the amount of floating point operations a CPU can handle per second.

Theoretical expected GFLOPS can be calculated from the following equation:

$$\text{GFLOPS} = F \times C \times n \dots \dots \dots (4.1)$$

- F : CPU frequency (GHz)
- C : number of CPU cores
- n : number of floating-point operations per CPU cycle

The value of n is CPU-specific; we can get this value from the microprocessor compliance metrics of the CPU provided by the CPU manufacturer.

Table (3.2) lists frequency, number of cores, and the value of n for each CPU used in experiments. (Intel, 2013)

Table 3.2: The value of n for CPUs used in experiments

CPU type	Frequency (GHz)	Number of Cores	n	GFLOPS
Intel Pentium 4 CPU	2.40	1	2	4.80
Intel Pentium 4 CPU	3.00	1	2	6.00
Intel Pentium 4 CPU	3.20	1	2	6.40
Intel Core 2 Duo CPU E6400	2.13	2	4	17.04
Intel Core 2 Duo CPU E6550	2.33	2	4	18.64
Intel Core 2 Duo CPU E8600	3.33	2	4	26.64
Intel Core 2 Duo CPU T6570	2.10	2	4	16.80
Intel Core2 Duo CPU T7300	2.00	2	4	16.00
Intel Pentium Dual-Core CPU E5200	2.50	2	4	20.00
Intel Core i3-2120	3.30	2	8	52.80

In order to reduce CPU heating and guarantee a green solution we have to reduce the used CPU time to 50% only, this value is recommended by the creators of BOINC (BOINC, 2012).

Number of connected PCs will be increased step by step as follows:

1. First Experiment: 3 PCs (selective PCs from computer and Internet labs)
2. Second Experiment: 8 PCs (Internet lab in Jerusalem branch)
3. Third Experiment: 16 PCs (Computer lab in Jerusalem branch)
4. Fourth Experiment: 24 PCs (both Internet and Computer labs)
5. Fifth Experiment: 33 PCs (Computer + Internet + Multipurpose labs)
6. Sixth Experiment: 43 PCs (Computer + Internet + Multipurpose labs in Jerusalem and the computer lab in Bethany)

For each step, the gained GFLOPS are to be recorded.

Credit performance metric is used instead of GFLOPS when testing the OS effect.

Credit is a measure of how much work a computer has done, A BOINC project gives credit for the computations a computer performs for it. BOINC's unit of credit, the Cobblestone, is 1/100 day of CPU time on a reference Computer (BOINC, 2012).

BOINC has a built-in crediting system to encourage volunteers to contribute and compete in BOINC projects, the more a volunteer contributes to a project -or several projects- the more credit he earns (BOINC, 2012), this credit is a description of the number of verified work units performed by a certain client and reflects the computational power of that PC.

We chose to record the credit since it is more significant for small number of PCs, GFLOPS are hard to compare since it provides integer readings only.

3.3.Summary

In this chapter, the research environment was described, and the intended experiments were described in general.

Next chapter represents the performed experiments of Alchemi and BOINC in detail, and provides discussion of obtained results.

Chapter Four

Experiments and Results

4.1. Part One: Measuring CPU utilization

The purpose of this part is to measure CPU utilization in the Computer laboratory PCs in order to reveal the actual idle computational resources in these PCs.

For this purpose, the CPU Usage Logger (CPU logger, 2012). This small, simple and reliable freeware utility offers a handy way for developers and software testers to easily monitor and log CPU usage (utilization) for any period of time. The optionally produced log file is tab delimited and easily importable into Microsoft Excel for further trending and analysis (CPU logger, 2012)

The tool calculates a 5-second moving average of CPU usage by averaging 10 samples taken at 500ms intervals as shown in Fig. (4.1).

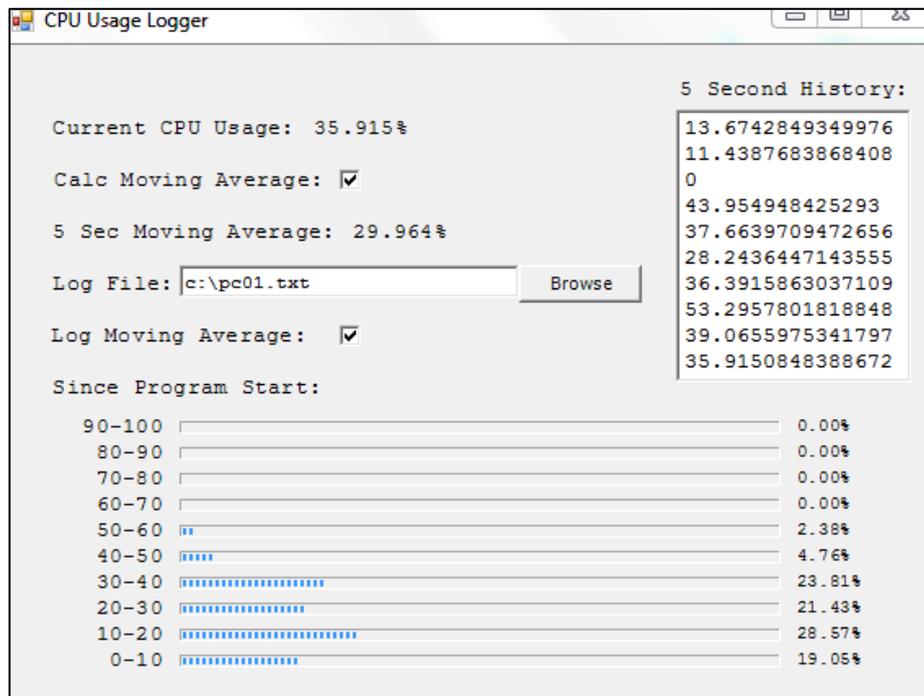


Fig. 4.1: Screenshot of CPU Usage logger tool interface

The following data is optionally logged to an outside tab-delimited log file if so desired:

- Date of reading
- Time of reading
- The 5-second moving average (last 10 samples averaged)

- Maximum CPU usage that occurred during the 5 second period (the max of the numbers that were averaged)
- Percentage of utilization falling in the 90-100% range since starting logging.
- Percentage of utilization falling in the 80-90% range since starting logging
- Percentage of utilization falling in the 70-80% range since starting logging
- Percentage of utilization falling in the 60-70% range since starting logging
- Percentage of utilization falling in the 50-60% range since starting logging
- Percentage of utilization falling in the 40-50% range since starting logging
- Percentage of utilization falling in the 30-40% range since starting logging
- Percentage of utilization falling in the 20-30% range since starting logging
- Percentage of utilization falling in the 10-20% range since starting logging
- Percentage of utilization falling in the 0-10% range since starting logging

This tool was deployed on 14 PCs in the computer laboratory in Jerusalem Branch. The tool was run during working hours (08:30 to 15:30) for 7 working days, CPU utilization readings were saved in log files in plain text format.

Although the duration of the measurement is short, but during this period, the computer lab was used intensively for semester registration, assignment solving, research editing, and practical examinations training, thus, measurements reflect good description of CPU utilization. More accurate results can be obtained by running the tool for a longer period of time (i.e. for one semester).

Fig. (4.2) illustrates a snapshot of a sample log with description of each field.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	CPU utilization percentage intervals													
2	Date	Time	Moving Average	Max CPU usage	90-100	80-90	70-80	60-70	50-60	40-50	30-40	20-30	10-20	0-10
3	10/06/2013	12:17:37	13.125	34.375	0	0	0	0	0	3	3	10	48	35
4	10/06/2013	12:17:42	26.349	87.5	0	2	0	0	0	2	2	20	41	32
5	10/06/2013	12:17:47	10.469	18.75	0	2	0	0	0	2	2	16	39	39
6	10/06/2013	12:17:52	35.388	75	0	2	2	2	0	5	3	16	34	36
7	10/06/2013	12:17:57	8.125	26.563	0	1	1	1	0	4	3	17	31	41
8	10/06/2013	12:18:02	4.957	12.5	0	1	1	1	0	4	2	15	30	46
9	10/06/2013	12:18:07	5.781	18.75	0	1	1	1	0	3	2	13	29	49
10	10/06/2013	12:18:12	5.133	21.875	0	1	1	1	0	3	2	13	26	53
11	10/06/2013	12:18:17	16.875	39.063	0	1	1	1	0	3	5	12	25	53
12	10/06/2013	12:18:22	3.75	10.938	0	1	1	1	0	2	4	11	24	56
13	10/06/2013	12:18:27	10.469	23.438	0	1	1	1	0	2	4	11	26	55
14	10/06/2013	12:18:32	18.125	45.313	0	1	1	1	0	3	6	10	26	54
15	10/06/2013	12:18:37	11.719	43.75	0	1	1	1	0	4	5	9	25	55
16	10/06/2013	12:18:42	6.094	20.313	0	1	1	1	0	4	5	10	24	56
17	10/06/2013	12:18:47	3.75	21.875	0	1	1	1	0	4	5	10	23	57
18	10/06/2013	12:18:52	6.875	18.75	0	1	1	1	0	3	4	9	23	58
19	10/06/2013	12:18:57	7.813	25	0	1	1	1	0	3	4	10	23	59
20	10/06/2013	12:19:02	5.938	17.188	0	0	0	0	0	3	4	9	22	60
21	10/06/2013	12:19:07	1.875	6.25	0	0	0	0	0	3	4	9	21	62

Fig. 4.2: Snapshot of sample log file

Using Microsoft Excel 2007 data tools, every log file was imported into a separate excel spreadsheet, the average was calculated for every range of every PC using the AVERAGE function, then these averages are averaged for the 14 PCs and are plotted as described in Fig. (4.3).

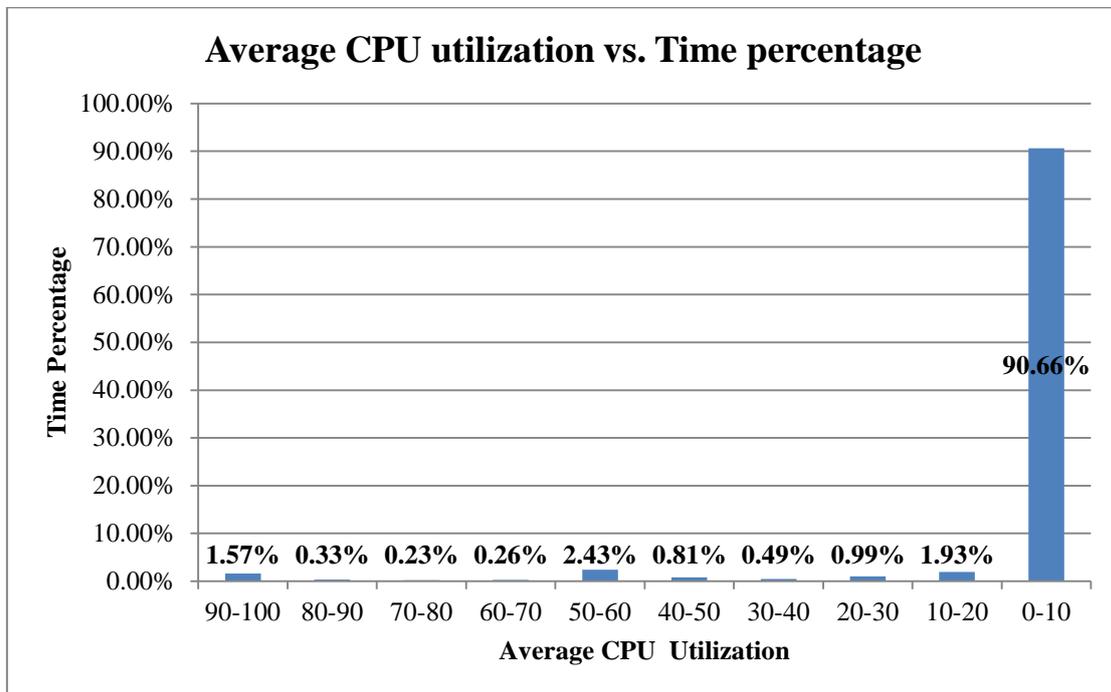


Fig. 4.3: Average CPU utilization

It can be noticed from Fig. (4.3) above that average CPU utilization falls in the 10-0% region for nearly 90% of the time.

4.2. Part Two: Examining Alchemi .NET

Testing Alchemi .NET ver. 1.0.6 (Sourceforge, 2012) is performed by installing Alchemi Manager on one PC, then installing Alchemi Executer on several PCs, and finally deploying grid application and observing the overall performance.

A grid is created by installing Executors on each machine that is to be part of the grid and linking them to a central Manager component (Alchemi, 2012).

In order to test Alchemi .NET, the Alchemi Manager was installed on one PC and Alchemi Executors were deployed on eight PCs located in Jerusalem branch multipurpose lab.

Steps for installing Alchemi grid are illustrated in Appendix D.

Alchemi Manager and Executors were installed on identical PCs; they have the same hardware components: Intel Core 2 Due, 2.33 GHz processor /1.0GB RAM / Microsoft Windows XP Professional.

Screenshot for the resulting grid is shown in Fig. (4.4):

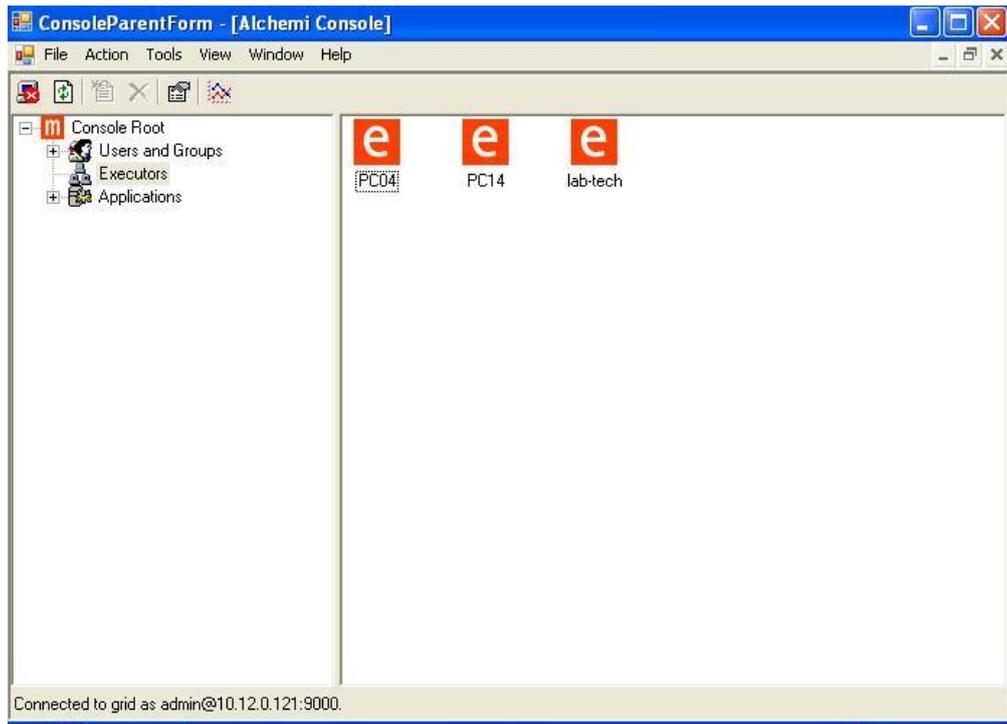


Fig 4.4: Alchemi Grid

Fig. (4.4) presents the Alchemi grid; one Manager node with three Executor nodes.

The next step after building the grid was to test it by running an example grid application.

The Alchemi .NET distribution package contains a source code for example applications (alchemi-1.0.6-sdk-net-2.0.zip) from which we chose the PI Calculator example developed in C# .NET. PI Calculator calculates the value PI for n decimal digits (default is 100 digits)

4.2.1. Testing Performance

The experiment was performed in Jerusalem branch multipurpose lab. One Executer was attached to one Manager and the example grid application (PI calculator for 100 digits) was run, and execution time is recorded.

Then we increased the number of grid PCs to 8, one PC at a time, and recorded execution time at each step, every step is repeated for 5 times and the average execution time is calculated.

The Execution time can be recorded from the PI calculator program console automatically.

Speed up factor for a number of Executors is the ratio of the average execution time of one Executor to the average elapsed execution time.

Table (4.1) shows recorded results and calculated speed-up factor.

Table 4.1: Average execution time for PI calculator

Executers	Average Execution Time(sec)	Speed up Factor
1	61.89	1.00
2	33.77	1.83
3	27.26	2.27
4	20.77	2.98
5	16.02	3.86
6	14.39	4.30
7	14.20	4.36
8	13.78	4.49

It can be noticed from Table (4.1) that execution time decreases when increasing number of grid PCs involved. Executors increased from 1 to 8 but average time elapsed from 61.89 seconds to 13.78 seconds only. The above results are illustrated in Fig. (4.5).

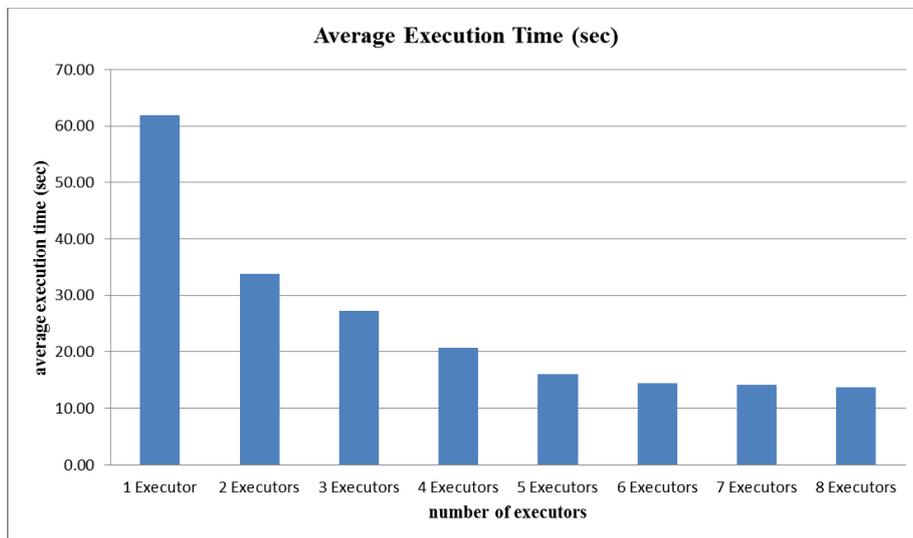


Fig. 4.5: Average execution time for PI calculator (100 digits)

Fig. (4.5) presents the average execution time in seconds vs. the number of Executor nodes involved.

Speed up factor is plotted against the number of Executors involved, and presented in Fig.(4.6).

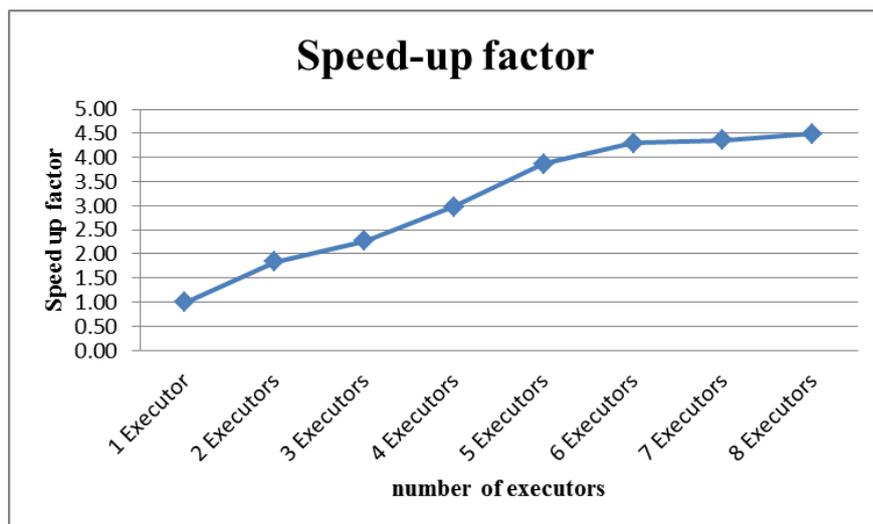


Fig. 4.6: Speed up factor vs. number of Executors

Fig.(4.6) illustrates how the speed up factor tends to be constant as the number of Executors increases.

Alchemi was then tested for a more complex calculation; the test was performed for a range of workloads (calculating 1000, 1200, 1400, 1600, 1800, 2000 and 2200 decimal digits of PI), each with one to eight Executors enabled. The workload was sliced into 10 threads; each thread is to calculate 10 decimal digits of PI. Execution time was recorded as the elapsed clock time for the test program to complete. Results are listed in Table (4.2).

Table 4.2: Execution time of PI calculator (in seconds) for increasing work loads

Executors	Decimal Digits of PI						
	1000	1200	1400	1600	1800	2000	2200
1 Executors	751.26	927.91	1214.96	1370.16	1652.72	1898.35	2205.97
2 Executors	358.63	442.06	450.95	645.85	767.24	901.88	1046.24
3 Executors	238.97	301.42	371.84	440.43	518.64	612.01	712.50
4 Executors	185.91	229.67	276.44	334.95	378.77	465.35	541.05
5 Executors	149.95	186.66	228.00	272.52	322.48	376.98	445.68
6 Executors	127.33	156.84	191.74	231.32	270.28	314.25	368.09
7 Executors	107.16	133.51	163.93	197.65	237.65	268.03	311.57
8 Executors	98.15	121.47	147.85	181.72	208.15	240.56	278.63

Table (4.2) presents the execution time for increasing workload with increasing number of Executors; these results are plotted and depicted in Fig. (4.7).

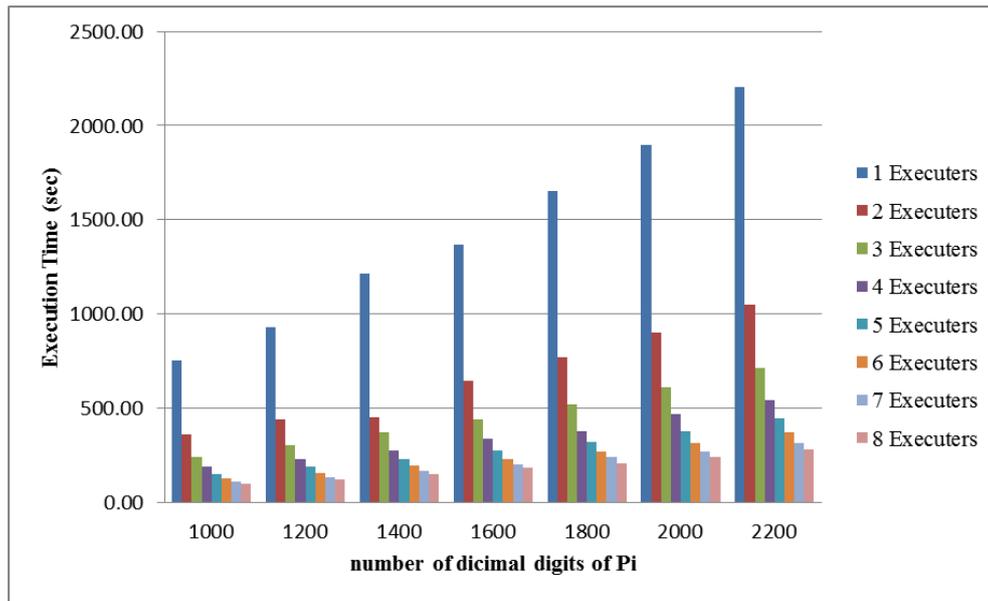


Fig 4.7: A plot of workload vs. execution time with varying number of Executors

Fig. (4.7) shows that as the workload increases, the efficiency of the grid increases in terms of execution time.

Speed up factor is plotted in Fig.(4.8).

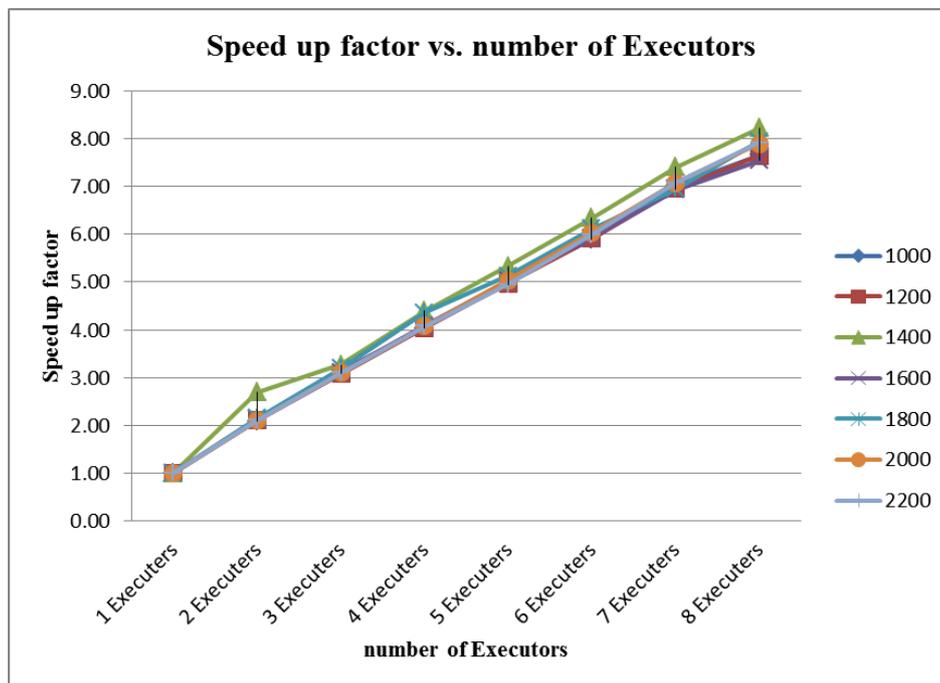


Fig. 4.8: Speed up factor for increased workload with varying number of Executors

Fig. (4.8) depicts the speed up factor for increased workload with varying number of Executors, It is noticed that the speed up factor is directly proportional to the number of Executors in this range.

4.2.2. Operating System effect

It is important to test the OS effect on the local PC grid since we have two different OSs in the labs: windows XP and windows 7. Windows 7 is expected to be dominant in the near future, so we need to know the effect of this upgrade on the local PC grid.

In this section we compare the performance of Alchemi grid under two different operating systems.

The Alchemi Grid is tested twice: once with Executor node (Intel Pentium 4/3.0 GHz CPU/1.0GB RAM) running windows XP, and the other is with the same Executor node running windows 7. The PI calculator program was set to calculate 100 decimal digits and Execution time is recorded, 5 trials each time. Table (4.3) shows experiment results.

Table 4.3: Operating system effect on Alchemi

OS	Average execution time(sec)
Windows XP	32.68
Windows 7	72.16

Figure (4.9) presents a chart of data in Table (4.3).

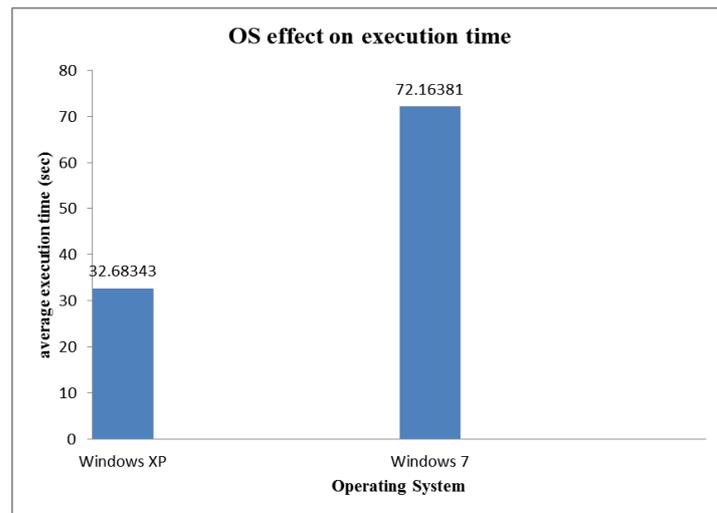


Fig. 4.9: Operating system effect on Alchemi

From Table (4.3) and Fig. (4.9) it is obvious that execution under Windows XP is $32.68/72.17 = 45.29\%$ faster than that of Windows 7.

A possible explanation for this result is that Windows 7 was released on 2007 (Microsoft, 2012), and Aneka (the new licensed version of Alchemi) was released on 2009 (ManjraSoft, 2012). The tested version of Alchemi (released 2006) was not developed to take advantage of new features of Windows 7 environment.

4.2.3. Testing network effect

In order to examine network effect on Alchemi Grid, the Manager node was located in Jerusalem branch computer lab and the Executor nodes were located in Bethany SC computer lab and again the PI calculator sample application is run.

It has been noticed that calculations were not performed at all; job threads were not sent from the Manager to any of the Executors, although these Executors are connected to the Manager and show in the manager console.

According to Alchemi support mailing list (Alchemi, 2013), this is a Domain Name Server problem that can be solved by editing the `C:\WINDOWS\system32\drivers\etc\hosts` file located on the hard drive of the Manager PC and adding an entry of each executor computer name and IP address.

This solution is not appropriate for large scale local PC grid environment, and this reveals a disadvantage of Alchemi.

4.3. Part Three: Examining BOINC

In this section, BOINC performance is tested. The BOINC server is installed and a project is created and run, and then BOINC clients are deployed on lab PCs, and they are attached to the created project, Fig. (4.10) illustrates the overall process.

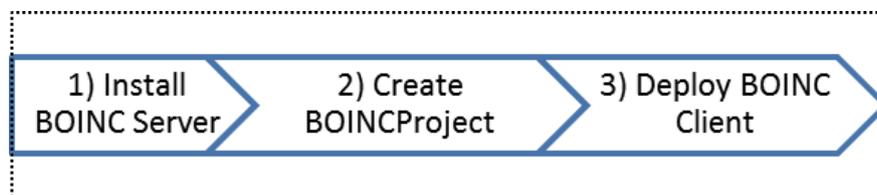


Fig 4.10: BOINC testing process

Detailed steps for installing BOINC grid are illustrated in Appendix E.

4.3.1. Testing Performance

To test the performance of the grid, we attached a small number of PCs then increased them gradually monitoring the output performance measured in GFLOPS, the number of generated GFLOPS can be recorded from the test project portal: <http://10.12.0.100/test>.

A screenshot of the Test project portal is illustrated in Fig. (4.11).

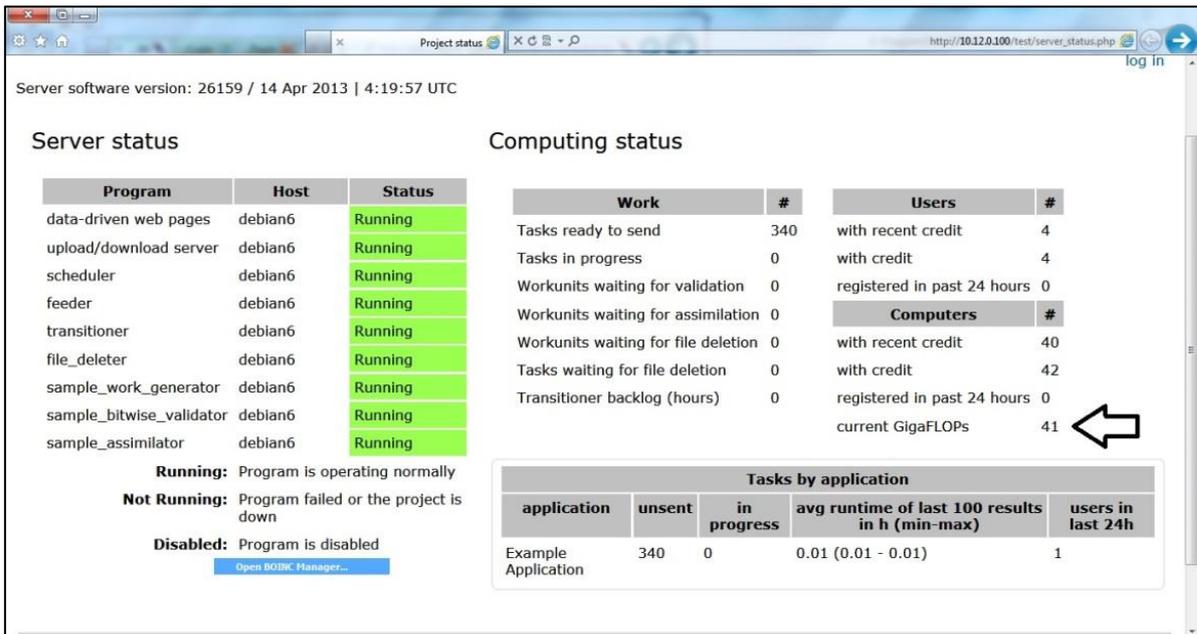


Fig 4.11: Monitoring server status

Fig. (4.12) shows a screenshot of the project status portal page, current GFLOPs can be read directly from the server portal as indicated.

Maximum CPU utilization was set to 50% only; this was performed by customizing the general computing preferences for the created grid user (*testgrid*) from the following link http://10.12.0.100/test_ops on the project's portal. Fig. (4.13) presents the screenshot of computing preferences webpage.

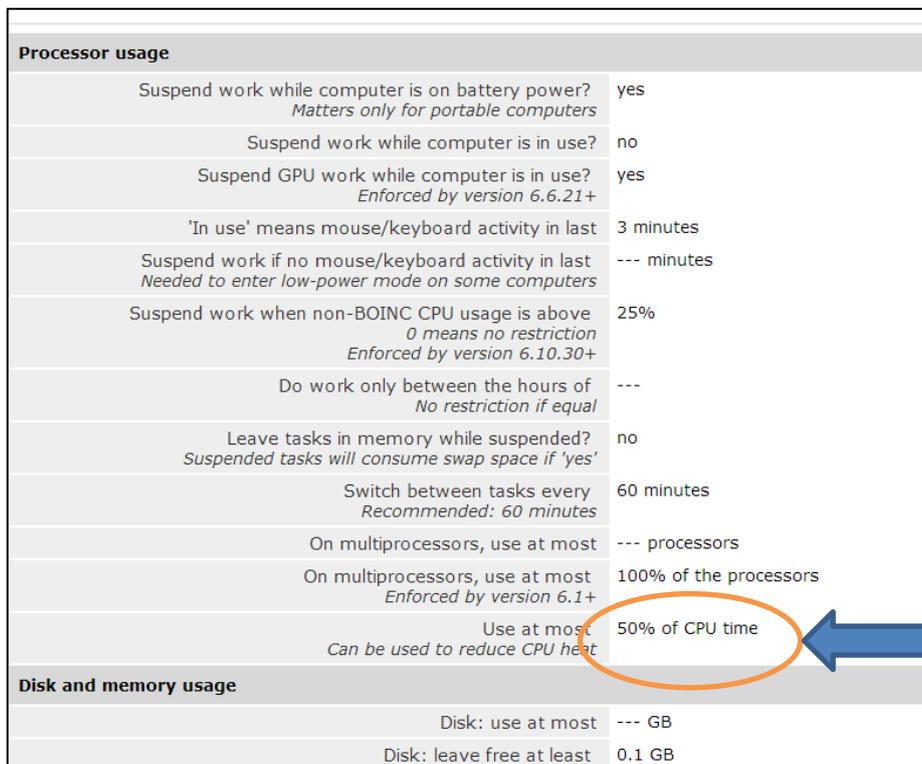


Fig. 4.12: computing preferences webpage

Table (4.4) shows the recorded compared to expected performance and utilization percentage of the BOINC grid.

Table 4.4: Recorded vs. expected performance for BOINC example project

Number of connected PCs	number of cores	harvested GFLOPS	expected GFLOPS	50% of expected GFLOPS	utilization
3	5	4	28.8	14.4	27.78%
8	8	11	48.4	24.2	45.45%
16	31	35	606.24	303.12	11.55%
24	39	51	654.64	327.32	15.58%
33	55	63	1129.84	564.92	11.15%
43	71	104	1320.48	660.24	15.75%

Table (4.4) lists the harvested GFLOPS vs. the number of PCs and CPU cores involved.

Expected GFLOPS were calculated by matching each CPU used in the experiment with GFLOP value provided by CPU manufacturer and listed in table (3.2), this value is then multiplied by 50% since this is the maximum set value for CPU utilization in this experiment.

Utilization percentage is the ratio of gained GFLOPS to the 50% of expected GFLOPS.

Data in Table (4.4) is plotted and presented in Fig.(4.13).

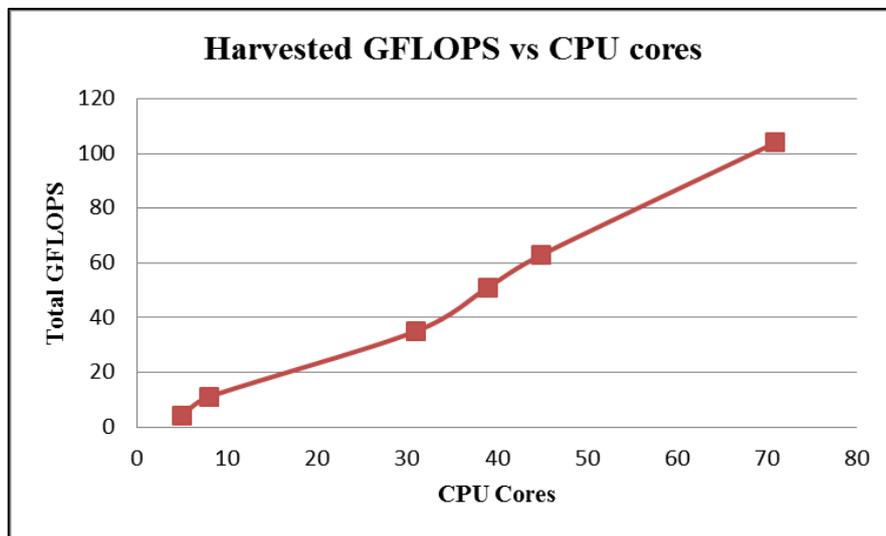


Fig. 4.13: Harvested GFLOPS vs. number of CPU cores involved

Fig.(4.13) describes the direct relation between harvested GFLOPS and number of CPU cores involved.

4.3.2. Testing Network effect

In order to experience the network effect, two readings of GFLOPS were taken; one with BOINC server and 8 clients located in Jerusalem branch and the other with the BOINC server located in Bethany SC.

Readings are listed in Table (4.5).

Table 4.5: Network effect on BOINC performance

Description	Gained GFLOPS
Server and clients in Jerusalem	11
Server in Bethany and clients in Jerusalem	9
percentage loss	18.18%

Table (4.5) presents the total gained GFLOPS in both situations, note that loss due to network effect is $11-9 = 2$ GFLOPS, which is a percentage of 18.18%.

The communication line between Jerusalem branch and ICTC is 2 Mbps, between Bethany SC and ICTC is 6 Mbps, thus the 18.18% loss is caused essentially by the slower link.

For the BOINC server to be located in ICTC, this loss is reduced to minimum due to reduction of routing delay.

4.3.3. Testing Operating System effect

Last test in this section is the effect of OS (for different CPU and RAM) on the overall contribution of a certain PC in the BOINC based grid.

To perform the test, we have chosen a PC running Windows XP and attached it to the BOINC test project. Two different CPUs are used each time, and for each CPU a different value of RAM is used, credit is recorded for each case.

After that, Windows 7 is installed and same procedure is applied. Readings are recorded at the end of each working day (total of 7 hours each).

Table (4.6) lists the recorded credit for each situation.

Table 4.6: OS effects on BOINC client for different CPU and RAM

	Windows XP				Windows 7			
CPU	3000 (Dual Core)	3000 (Dual Core)	3000	3000	3000 (Dual Core)	3000 (Dual Core)	3000	3000
RAM (GB)	1.0	1.5	1.0	1.5	1.0	1.5	1.0	1.5
PC Credit	2958.56	3604.16	1263.89	1775.51	3722.22	4287.04	1983.80	2341.13
Performance improvement percentage					26%	19%	57%	32%

Table (4.6) presents credit values recorded for each setup and performance improvement percentage. Performance improvement percentage is calculated by dividing the difference between adjacent Windows 7 credit and windows XP credits by the windows XP credit. One can note that Windows 7 performance is better than Windows XP performance at all cases. One can also note that the CPU is the major factor that contributes in the total credit.

The Performance improvement percentage was calculated by dividing the host credit difference by the credit of windows XP.

Example:

- Windows XP credit for a 3.0 GHz Dual- Core CPU / 1.5 GB RAM = 3604.16,
- Windows 7 credit for a 3.0 GHz Dual- Core CPU / 1.5 GB RAM = 4287.04
- Difference = 4287.04 - 3604.16 = 682.88
- Performance improvement = $682.88/3604.16 = 18.95\%$

4.4. Results and discussion

The three main experiment parts that were performed are:

1. Measuring CPU utilization in computer lab PCs
2. Building and testing Alchemi –based grid
3. Building and testing BOINC –based grid

These experiments are and described in sections 4.1, 4.2 and 4.3 respectively, results of each part are discussed here.

4.4.1. Part one: Measuring CPU utilization in computer lab PCs

The test has been performed on 14 PCs in the computer lab. Readings were taken for 7 work days during work hours only (08:00 to 15:30).

It has been revealed that average CPU utilization does not exceed 10% CPU utilization for nearly 90% of the test time.

This result comes in favor with other results presented by other researchers:

- Mutka stated that desktop PCs can be under-utilized by as much as 75% of the time (Mutka, 1992)
- Acharya and co-researchers stated that idle time for desktop machines ranged from 60% to 80% (Acharya et al., 1997)
- Domingues and co-researchers noted that average CPU idleness is 97.93% (Domingues et al., 2005)

- Vlădoiu and co-researchers the computational availability approximately 75-80% during work hours (Vlădoiu et al., 2009).

It is worth noting that in (Vlădoiu et al., 2009) the researchers were only concerned in the number of the available running PCs not in the percentage of idle CPU time for every PC in the laboratory which is more accurate; consider a case of a running PC that performs heavy tasks, such PC cannot be considered as “available”.

The process of collecting and analyzing CPU utilization data was performed manually; i.e. log files were collected from each machine and imported to Microsoft Excel. An automatic mechanism was introduced by Han and Gnawali (Han et al., 2012), in their article they built a C++ application to measure CPU usage and installed as a Windows Service on all computers in the lab.

Every second, the process monitor calculated each process CPU usage by using Windows Management Instrumentation API. The process monitor then transmits the list of processes and their CPU utilization to a database server over wired Ethernet.

The result of their measurement was that the fraction of CPU utilization while user is not logged in stayed below 30% on most machines, due to real-time back up and viruses scans and other background processes (Han et al., 2012).

Our results emphasize the fact that idle CPU power is ample in PCs available in computer labs, and can be harnessed to build a local PC Grid environment.

4.4.2. Part two: Results of examining Alchemi .NET

After testing the Alchemi Framework, several points can be highlighted:

- Installation: Alchemi was easy to install and deploy (plug-and-play), with no need to worry about scripts and configuration files.

System requirements are simple: .NET framework for both Executor and Manager Nodes, and SQL server for Manager Node only.

- Overall Performance: two experiments were performed to test the grid, for both experiments, 8 identical Executors were involved, and the PI calculator was set to calculate 10 decimal digits for each thread:
 1. First: Alchemi was set to calculate 100 decimal digits of PI
 2. Second: Alchemi was set to calculate decimal digits of PI from 1000 up to 2200, adding 200 decimal digits each time just to increase workload.

1. Results for the first experiment:

Table (4.7) summarizes the results obtained from Table (4.1)

Table 4.7: Comparison between actual and ideal execution time and speed up factor

Executors	average execution time (sec)	ideal execution time (sec)	actual speed up factor	ideal speed up factor
1 Executor	61.89	61.89	1.00	1
2 Executor	33.77	30.95	1.83	2
3 Executor	27.26	20.63	2.27	3
4 Executor	20.77	15.47	2.98	4
5 Executor	16.02	12.38	3.86	5
6 Executor	14.39	10.32	4.3	6
7 Executor	14.2	8.84	4.36	7
8 Executor	13.78	7.74	4.49	8

Table (4.7) compares between actual execution time and speed up factor values along with expected values.

Ideal execution time is based upon an assumption that execution time reduces in a fraction according to the involved Executors, for example: in this experiment the execution time when we used one Executor was 61.89 and when using 2 executors, the work load is divided equally between them, so the expected execution time is $61.89/2$ and so on.

Referring to (Trifa et al., 2011) the *actual speedup factor* that Trifa used was found by calculating the ratio of actual execution time for every level of the experiment with the recorded execution time for one Executor.

For example: the actual speed up factor when 8 Executors are used is $61.89/13.78 = 4.49$, while the ideal speed up factor is 8 since we used 8 Executors.

The theoretical performance of Alchemi is can be expected according to the following equation:

$$T(n) = T(1)\left(\frac{1}{n}\right) \dots \dots \dots (4.2)$$

Where:

- T(n) = expected execution time for *n* Executors, *n* is a positive integer \neq zero
- T(1) = recorded execution time for one Executor

Fig.(4.14) presents the difference between actual vs. ideal execution time.

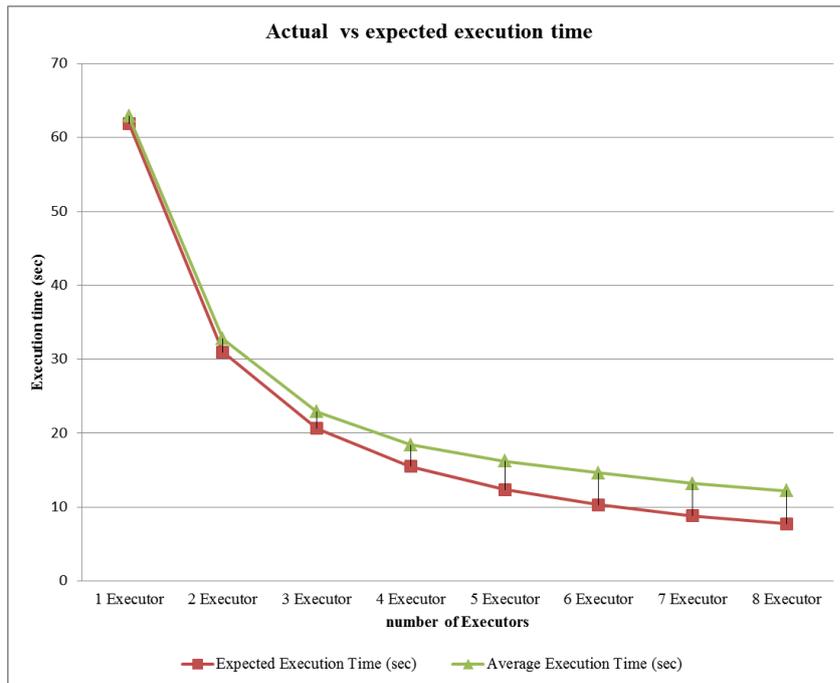


Fig. 4.14: Actual vs. ideal execution time

Based on Table (4.8) and Fig. (4.14), equation (4.2) can be written as follows taking into consideration slow down factors like network traffic and CPU occupancy:

$$T(n) = a * T(1)/n^c \dots\dots\dots (4.3)$$

Where a and c are constant coefficients, these coefficients represent the error produced by network communication and volatile CPU power availability.

To be able to a measure how well recorded outcomes are replicated by the model we will use the coefficient of determination, denoted R^2 and pronounced R squared. The model is said to closely represent the recorded outcomes if R^2 is closer to one (Coefficient of determination, 2013).

Using Microsoft Excel Trend line utility, it has been found that the function that best fits the recorded values is $y = 59.453 x^{-0.753}$, and R^2 value equals 0.9837 which is close to one, this means that the trend line nearly fits the actual data.

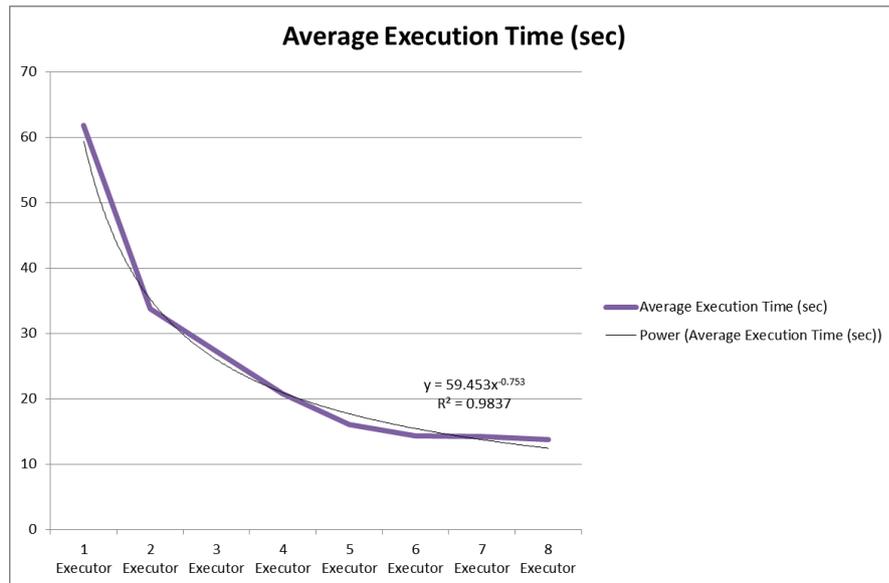


Fig. 4.15: Trend line for actual execution time

Fig. (4.15) describes the trend line of the actual recorded execution time.

Accordingly, a in equation (4.3) equals $59.453/61.89$ which yields 0.961 , and c equals 0.753 .

Speed up factor:

Fig (4.16) represents the comparison between actual and ideal speed up factor, one can notice that actual speed up factor in this experiments tends to be constant after the 5th Executor; this means that adding more Executors will not significantly improve the overall performance of the Alchemi grid.

This result comes in favor with the results of Luther and co-researchers (Luther et al., 2005), the reason behind this is that increasing the number of Executors creates a network overhead that delays the total execution time (Luther et al., 2005)

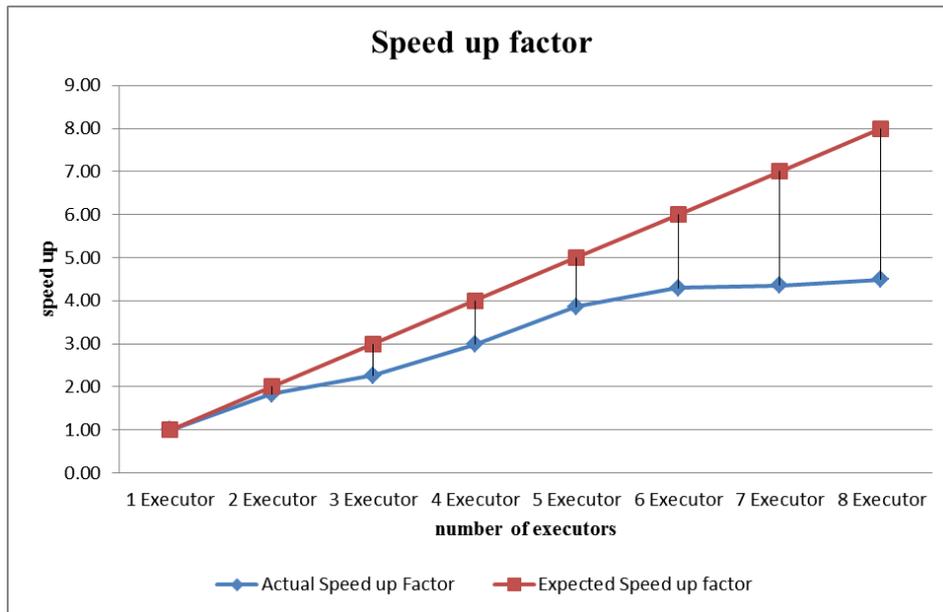


Fig.4.16: Actual vs. ideal speed up factor

Fig. (4.16) presents the difference between actual and ideal speed up factor. Note that after the 5th Executor the speed up factor tends to be constant.

2. Results for the second experiment:

The workload was increased in this experiment, the grid performance was observed and execution time was recorded.

Table (4.8) presents the recorded execution time for each value of calculated decimal digits of PI.

Table 4.8: Execution time (in seconds) for different decimal digits of PI

Executors	Decimal Digits of PI						
	1000	1200	1400	1600	1800	2000	2200
1 Executors	751.26	927.91	1214.96	1370.16	1652.72	1898.35	2205.97
2 Executors	358.63	442.06	450.95	645.85	767.24	901.88	1046.24
3 Executors	238.97	301.42	371.84	440.43	518.64	612.01	712.50
4 Executors	185.91	229.67	276.44	334.95	378.77	465.35	541.05
5 Executors	149.95	186.66	228.00	272.52	322.48	376.98	445.68
6 Executors	127.33	156.84	191.74	231.32	270.28	314.25	368.09
7 Executors	107.16	133.51	163.93	197.65	237.65	268.03	311.57
8 Executors	98.15	121.47	147.85	181.72	208.15	240.56	278.63

Applying the same approach from the previous experiment, we calculated the a and c coefficients, these coefficients along with R-square values are presented in Table (4.9).

Table 4.9: a and c and R^2 of recorded data

Decimal digits	a	c	R^2
1000	0.965	0.977	0.999
1200	0.967	0.976	0.999
1400	0.883	0.970	0.983
1600	0.958	0.971	0.998
1800	0.951	0.986	0.997
2000	0.972	0.989	0.999
2200	0.972	0.987	0.999
Average	0.979	0.953	0.996

Comparing results in Table (4.9) with results from previous experiments, one can notice that the overall performance of the grid is very close to ideal theoretical model described in equation (4.2). We can conclude that for heavy workloads, the efficiency of Alchemi is better.

Fig. (4.8) also describes that the speed up factor is directly proportional to the number of Executors in hand.

On the other hand, in this experiment, we set each thread in the PI calculator to calculate only 10 decimal digits. Comparing with (Luther et al., 2005), the researchers set the thread to calculate 50 decimal digits. It is noticed that the execution time in their experiment is less than execution time for our experiment, for example: at 2200 decimal digits on 6 Executors, our reading is 368.09 seconds and their reading is about 80 seconds although they performed the experiments on old Pentium III 1.7 GHz/512 MB desktops. Fig.(4.17) illustrates the results of Luther and co-researchers.

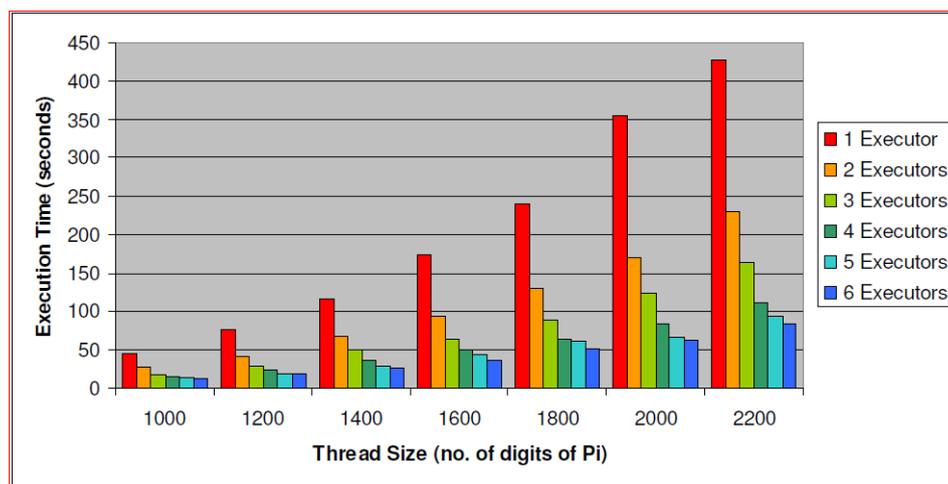


Fig. 4.17: A plot of thread size vs. execution time (Luther et al., 2005)

The above comparison implies that Alchemi threads transfer have great overhead on the network, this overhead causes delay in overall execution time.

- Operating system effect:

When testing the Alchemi performance under windows XP and Windows 7 for same hardware. Average Execution time under Windows XP was recorded to be 32.68 seconds compared to 72.16 seconds for Windows 7; thus execution under Windows XP is $(72.17 - 32.68)/32.68 = 120.8\%$ faster than that of Windows 7.

We did not find any article that supports this since all surveyed articles concerning Alchemi described its deployment under Windows 2000 (Setiawa et al., 2004) or windows XP (Vecchiola et al., 2007).

This shortage in Alchemi prevents us from deploying it in QOU, since QOU updates hardware and software regularly, and windows 7 will be the default OS in the near future.

- Network effect:

When connecting Executors in a subnet to a Manager in another subnet, it has been noticed that job threads were not sent from the Manager to any of the Executors, although these Executors can “see” the Manager and appear in the Alchemi Manager Console.

This fact records a shortage in the Alchemi framework probably in "hopping" jobs from one subnet to the other. An approach that needs testing is to use that Alchemi hierarchy model, where there is a Manager node in each subnet that acts as an Executor for a central Manager, it receives job threads and distributes them to underlying Executors in the subnet then collects and sends results to the “General Manager” (Alchemi, 2012).

- CPU utilization:

Alchemi was observed to consume all available idle CPU power, this is illustrated in Fig. (4.19). There is no available option in Alchemi Manager to limit the CPU allowable usage to a certain percentage.

This is apparently an advantage, but the fact is the more the CPU is busy the more it is heated and the more the PC consumes electricity for cooling, this increases cost and violates green solution requirement of the PC Grid.

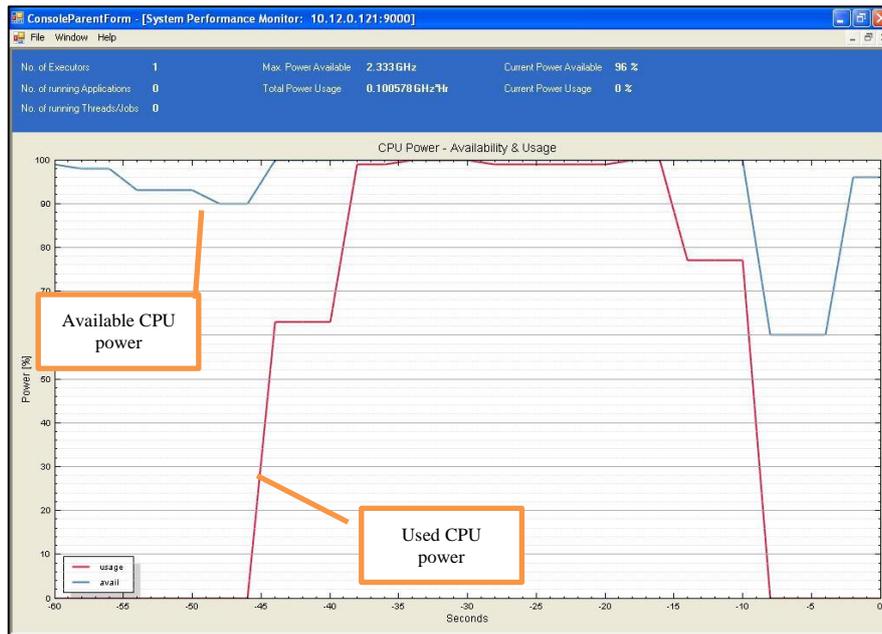


Fig. 4.18: Alchemi Console showing available vs. used CPU power

Fig. (4.18) illustrates how Alchemi consumes all available CPU power to perform thread calculations.

4.4.3. Part three: Results of examining BOINC

- Installation:

One of the disadvantages of BOINC is the installation complexity. BOINC server runs in Linux machines and requires Linux administration skills to install and run software prerequisites.

- Performance:

As described in section 4.4, PCs were attached to the sample project on the BOINC server, and the total performance was observed and recorded in terms of GFLOPS. Results are presented in Table (4.5).

Expected GFLOPS are calculated using equation (4.1), keep in mind that CPU utilization for client PCs was set to 50% in order to minimize CPU heating.

Utilization was calculated by dividing the gained GFLOPS by the ideal GFLOPS.

Data presented in Table (4.5) is plotted in Fig.(4.20), a trend line is also plotted.

The trend line is linear, its equation is:

$$Y = 1.49 X - 4.62 \dots\dots\dots (4.4)$$

Where: Y represents gained GFLOPS and X represents number of cores involved. The constant (- 4.62) can be neglected for large number of CPU cores.

As noticed in Fig. (4.19), R^2 equals 0.9893 which is close to 1, this means that equation (4.4) strongly describes the actual performance of the BOINC grid.

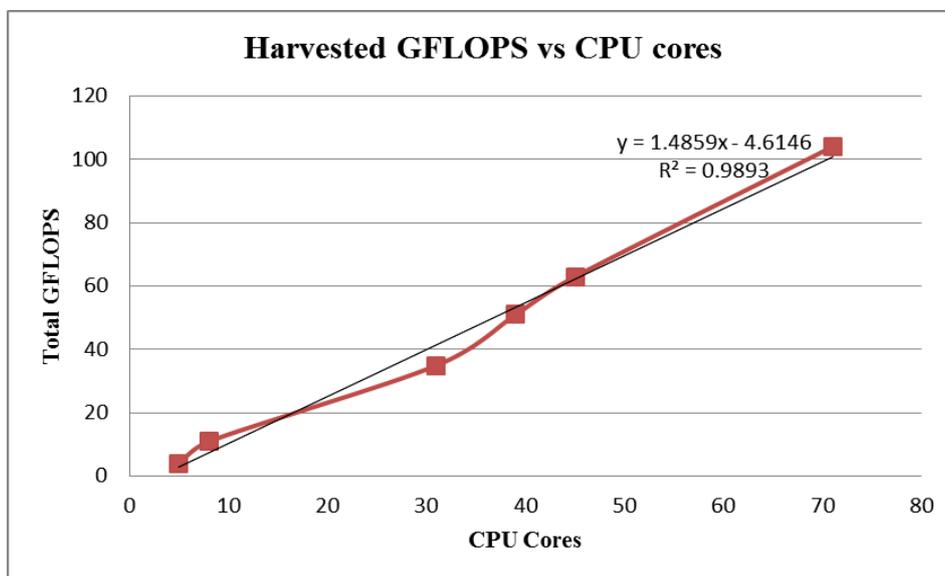


Fig. 4.19: Gained GFLOPS with Trend line

Although, the explored 71 cores cannot be used as a scientific sample, we can show that for a total of 3550 CPU cores that are available at QOU computer labs (Appendix A) holding the same limitations and assumptions of the same experiment conditions, we expect to gain $1.49(3550) - 4.62 = 5270$ GFLOPS = 5.27 TeraFlops.

Although, we have obtained these promising and potential results, we need to have a scientific sample before coming to a formal declaration.

- Operating system effect:

BOINC client performance was tested under Windows XP and Windows 7 with changes made to CPU, RAM each time. Results showed that BOINC has ranked great performance under windows 7; range of performance improvement percentage is from 19% to 57%.

- Network effect

Table (4.6) presented the total loss in GFLOPS due to communication traffic to be 18.18%.

The communication line between Jerusalem branch and ICTC is 2 Mbps, between Bethany SC and ICTC is 6 Mbps, thus the 18.18% loss is caused essentially by the slower link. The 2 Mbps causes 6/8 of delay which equals 13.64% and the other fraction of the delay is caused by the 6Mbps line (around 4.56%). Network delay can be decreased by locating the BOINC server in ICTC.

4.5. Summary

In this Chapter, several experiments have been performed: First, the CPU utilization was **measured in order to reveal the actual utilization of computer lab PCs**, then the Alchemi framework was installed and tested, and finally the BOINC framework was installed and tested. Major results and findings were also highlighted and discussed.

Chapter five represents the conclusion and suggestions for future work.

Chapter Five

Conclusion and Future Work

5.1. Conclusion

The general objective of this work is to introduce a local PC grid computing environment in Higher Educational Institutions presenting QOU as an example case (Jerusalem Branch), by harvesting idle CPU power available in regular, LAN connected, Computer lab PCs running Windows Operating System in these institutions.

This objective was successfully reached by two approaches: one is by deploying a PC grid computing middleware (Alchemi .NET), and the other is by customizing a Public Resource Computing (BOINC) for local PC grid usage.

The first finding in this research is that CPU utilization in computer labs is very low. The average CPU utilization for a 7 working days (0800: 15:30) is less than 10% for 90% of the time, this comes in agreement with all surveyed researches in the field.

Our results emphasize the fact that idle CPU power is ample in PCs available in computer labs, and can be harnessed to build a local PC Grid environment.

When testing Alchemi as the first approach, it has been found that Alchemi was very easy to install and run. In addition, Alchemi was also powerful and reliable when deploying heavy computation. The execution time was used as the performance evaluation parameter and was found to meet the ideal case.

The expected execution time has been modeled as

Expected Execution Time (EET) is $ET(n) = T(1)/n$,

where: $T(1)$ is the experimental execution time for one Executor and

n is an integer representing the number of Executors.

Alchemi was deployed on eight Executors, and a curve fitting to the resulting readings was performed, the resulting model for Experimental Execution Time ($T(n)$) was:

$$T(n) = 0.979 T(1) / n^{-0.953}$$

The relation between the two models was measured by coefficient of determination, R^2 which is found to be 0.996; this means that the model is very close to ideal case

Despite of Alchemi's advantages, it was revealed that it has some shortages that include:

1. Alchemi couldn't send jobs between different subnets, thus it can only be deployed in one branch in QOU environment, an approach that needs testing is to use that Alchemi hierarchy model (Alchemi, 2012).
2. Not green! : Alchemi was observed to consume all available idle CPU power; there is no available option in Alchemi Manager to limit the CPU allowable usage to a certain percentage.
3. Alchemi proved poor performance when running under Windows 7. Execution time under Windows XP was recorded to be 120.8% faster than that of Windows 7. This shortage in Alchemi prevents us from deploying it; QOU updates hardware and software regularly, and windows 7 will be the default OS in the near future.
4. Alchemi causes network traffic overhead that may delay transferring other important data especially when given small shrinks of jobs for each Executor.

When testing BOINC as the second approach, we were able to gain 108 GFLOPS out of 71 CPU cores with nearly no cost.

Curve fitting of observed GFLOPS produced the following relation between involved cores (X) and gained GFLOPS (Y)

$$Y = 1.49 X - 4.62$$

Which is very close to the actual curve ($R^2 = 0.9893$).

We were able to estimate a total of nearly costless 5.27 TeraFLOPS for 3550 CPU cores available in computer lab PCs. Although, we have obtained these promising and potential results from small exploration sample, we need to have a scientific sample before coming to a formal declaration.

When comparing BOINC client performance under windows XP then Windows 7, BOINC client running under Windows 7 produced more work compared with windows XP, improvement percentage is from 19% up to 57%. This result is encouraging since it is expected for Windows 7 to replace windows XP in Computer labs in the near future.

Some decrease in computation was observed due to communication between clients and server, the delay caused by the 6Mbps line was estimated to be around 4.56%. This delay can be dramatically reduced when locating the BOINC server in ICTC, the central hub of QOU.

Despite the above advantages of BOINC, the main disadvantage is the complication of the setup process; BOINC server runs on Linux machines and requires Linux administration skills to install and run software prerequisites that are mostly performed from the command prompt not from GUI.

5.2. Future Work

Although, we have obtained promising initial results, but still the following recommendations may help to further contributions in PC Grid computing:

1. Our research was limited to harnessing idle CPU power available in Computer Lab PCs, without any addressing to the Graphical Processing Unit (GPU) power. GPU possess huge computational power that needs to be explored and utilized in a grid computing environment.
2. Distributed file storage needs also to be addressed in HEI. Large hard drive capacities are available but a small portion is used; for instance: a 500 GB hard drive is the minimum standard these days, but for a computer lab PC uses a maximum of 100 GB only and the rest is not used.
3. A study of the peak times of data traffic on the HEI LAN during working hours needs also to be addressed, so as to stop running grid applications in such periods of time to avoid network traffic delay.
4. Finally, BOINC client performance was tested under Windows XP and Windows 7, but now Windows 8 has hit the market; and sooner or later it will be available in HEI computer labs. A new stable version of BOINC client is available (version 7.0.64) since 17/April/2013; testing it under Window 8 is a great idea for future work.

References

1. Acharya A., Edjlali G., and Saltz J., "The utility of exploiting idle workstations for parallel computation". Proceedings of the ACM SIGMETRICS '97 international conference on Measurement and modeling of computer systems, p.p. 225-234, 1997.
2. Alchemi support mailing list, <http://www.mail-archive.com/alchemi-users@lists.sourceforge.net/msg00484.html>, (accessed on 08/04/2013)
3. Alchemi User Guide, <http://sourceforge.net/projects/alchemi/files/alchemi-.NET.2.0/>, (accessed on 04/12/2012).
4. Anderson, D., Korpela, E. and Walton, R., "High-Performance Task Distribution for Volunteer Computing", Proceedings of the First IEEE International Conference on e-Science and Grid, . Melbourne, Australia, 2005.
5. Anderson, D., "Public Computing: Reconnecting People to Science", Conference on Shared Knowledge and the Web, Residencia de Estudiantes, Madrid, Spain, Nov. 17-19, 2003.
6. Baldassari J., "Design and Evaluation of a Public Resource Computing Framework", Worcester Polytechnic Institute, 2006.
7. Bellard F., Computation of the n^{th} digit of PI in any base in $O(n^2)$, (accessed on 07/06/2013)
8. Bellavista P., Chang R., Chao H, Lin S, Sloot P., "Advances in Grid and Pervasive Computing", 5th International Conference Proceedings", Hualien, Taiwan, Springer, 2010.
9. BOINC wiki, <http://www.boinc-wiki.info/>, (accessed on 04/01/2013).
10. BOINC: Berkeley Open Infrastructure for Network Computing. <http://boinc.berkeley.edu/> (accessed on 12/12/2012).
11. BOINCstats, <http://www.boincstats.com/>, (accessed 10/04/2013)
12. Buyya R., Luther A., Ranjan R. , Venugopal S.. "Alchemi: A .NET-based Enterprise Grid Computing System", 6th International Conference on Internet Computing (ICOMP'05), Las Vegas, 2005.
13. C. Kruskal , A. Weiss, "Allocating independent subtasks on parallel processors", IEEE Transactions on Software Engineering, 11:1001-1016, 1984.
14. Caphyon, <http://www.advancedinstaller.com>, (accessed on 03/01/2013).
15. Chien A., Calder B., Elbert S., and Bhatia K., "Entropia: Architecture and Performance of an Enterprise Desktop Grid System", Journal of Parallel and Distributed Computing, Volume 63, Issue 5, Academic Press, USA, May 2003.
16. Choi S., Kim H., Byun E., Baik M., Kim S., Park C., Hwang C., "Characterizing and Classifying Desktop Grid", Seventh IEEE International Symposium on Cluster Computing and the Grid (CCGrid'07), IEEE 2007.
17. Cloudbus, <http://www.cloudbus.org/~alchemi/>, (accessed on 10/05/2011).

18. Coefficient of determination, <http://www.coefficientofdetermination.com/>, (accessed 12/05/2013).
19. CPU logger, <http://sourceforge.net/projects/cpu-usage-log/>, (accessed on 02/03/2012).
20. Debian, <http://www.debian.org>, (accessed on 10/06/2012).
21. distributed.net, <http://www.distributed.net>, (accessed 10/01/2013)
22. Domingues P., Marques P., Luis Silva L., “Resources Usage of Windows Computer Laboratories”, Proceedings of the International Conference on Parallel Processing Workshops (ICPPW’05), 2005.
23. Foster, I., “What is the grid? A three-point checklist”, Grid Today, 20/07/2002. Available online <http://dlib.cs.odu.edu/WhatIsTheGrid.pdf>, (accessed on 18/03/2013).
24. Foster, I. and Kesselman, C.,”The grid: Blueprint for a new computing infrastructure”, Morgan Kaufmann, San Francisco, CA,1998.
25. Foster, I. and Kesselman, C., “The Grid 2: Blueprint for a New Computing Infrastructure”, Morgan Kaufmann, 2nd ed., ch.4. San Francisco, CA, 2004.
26. Foster, I., Kesselman, C. and Tuecke, S., “The Anatomy of the grid: Enabling scalable virtual organizations”, International Journal of High Performance Computing Applications, 15(3): 200-222, 2001.
27. Foster, I., Kesselman, C. and Tuecke, S., “What is the Grid? A Three Point Checklist”, s.l. : Grid Today, 2002.
28. Germain C., Neri V., Fedak G., Cappello F., “XtremWeb: building an experimental platform for Global Computing”, Proceedings of the 1st IEEE/ACM International Workshop on Grid Computing (Grid 2000), Bangalore, India, 12/2000.
29. González, D., Gil, G., De Vega, F., Segal, B.,”Centralized BOINC Resources Manager for Institutional Networks”, IEEE, 2008.
30. Goyal B, Lawande S.,”Grid Revolution: An Introduction to Enterprise Grid Computing”. McGraw-Hill, Emeryville, CA, 2005.
31. Han D., Gnawali O., “Understanding Desktop Energy Footprint in an Academic Computer Lab”, Green Computing and Communications (GreenCom) Conference, IEEE, 2012.
32. HTCondor, <http://research.cs.wisc.edu/htcondor/index.html>, (accessed 11/01/2013)
33. ICTC, “Official statistics provided by Information & Communication Technology Center”, QOU, 2013.
34. Intel microprocessor export compliance metrics, <http://www.intel.com/support/processors/sb/cs-017346.htm>, (accessed on 09/06/2013).
35. Luther A., Buyya R., Ranjan R., Venugopal S., “Peer-to-Peer Grid Computing and a .NET-based Alchemi Framework”, High Performance Computing: Paradigm and Infrastructure, Wiley Press, 2005.
36. Magoulès F., Pan J., Tan K, Kumar A., “Introduction to Grid Computing”, CRC Press, 2009.

37. ManjraSoft, <http://www.manjrasoft.com/>, (accessed on 02/04/2012).
38. Microsoft Incorporation, <http://www.microsoft.com/>, (accessed on 04/04/2012).
39. Microsoft Technet, [http://technet.microsoft.com/en-us/library/cc780036\(WS.10\).aspx](http://technet.microsoft.com/en-us/library/cc780036(WS.10).aspx), (accessed on 13/02/2013).
40. Moore, G., "Cramming more components onto integrated circuits", Electronics Magazine, Volume 38, Number 8, April 19, 1965.
41. Mustafee N., "A grid computing framework for commercial simulation packages", School of Information Systems, Computing and Mathematics Brunel University, UK, May, 2007.
42. Mutka M., "Estimating capacity for sharing in a privately owned workstation environment". IEEE Transactions on Software Engineering 1992, 18(4):319-328.
43. Oracle, <https://www.virtualbox.org/>, (accessed on 11/06/2012).
44. QOU, Al-Quds Open University website, <http://www.qou.edu/> , (accessed on 03/02/2013)
45. Semeria C., "Multiprotocol Label Switching: Enhancing Routing in the New Public Network", Juniper Networks, 2001.
46. SETI, <http://setiathome.berkeley.edu/>, (accessed on 04/01/2013)
47. Setiawa A., Adiutama D., Liman J., Luther A., Buyya R., "GridCrypt: High Performance Symmetric Key Cryptography Using Enterprise Grids", Proceedings of the 5th International Conference on Parallel and Distributed computing, Applications and Technologies, 2004.
48. SourceForge, [http://sourceforge.net/projects/alchemy/files/alchemy-.NET 2.0/Alchemy v.1.0.6/](http://sourceforge.net/projects/alchemy/files/alchemy-.NET%202.0/Alchemy.v.1.0.6/), (accessed on 04/12/2012).
49. Stanoevska K., Wozniak T., Ristol S., "Grid and Cloud Computing: A Business Perspective on Technology and Applications", Springer, 2010.
50. SZTAKI, Computer and Automation Research Institute of the Hungarian Academy of Sciences (MTA), <http://www.desktopgrid.hu>, (accessed on 10/06/2012)
51. Tachikawa, M., "PC Grid Computing: Using Increasingly Common and Powerful PCs to Supply Society with Ample Computing Resources", Science and Technology. Quarterly Review No.18/January, 2006.
52. TOP500 list, <http://www.top500.org/list/2012/11/>, (accessed 10/04/2013).
53. Trifa, Z., Labadi M., Khemakhem M., "Arabic Cursive Characters Distributed Recognition using the DTW Algorithm on BOINC: Performance Analysis", International Journal of Advanced Computer Science and Applications (IJACSA), Vol. 2, No.3, March, 2011.
54. Univa, <http://www.univa.com/>, (accessed 09/01/2013).
55. UWM, The University of Westminster website, [http://wgrass.wmin.ac.uk/index.php/Desktop_Grid:Westminster Local DG](http://wgrass.wmin.ac.uk/index.php/Desktop_Grid:Westminster_Local_DG), (accessed 10/08/2012).

56. Vecchiola C., Nadiminti K., Buyya R., “Image Filtering on .NET-based Desktop Grids”, GCC '07 Proceedings of the Sixth International Conference on Grid and Cooperative Computing, pp 582-592, IEEE, 2007.
57. Vlădoiu M., Constantinescu Z., Negoiană C., “Availability of Computational Resources for Desktop Grid Computing”, Seria Matematică - Informatică – Fizică, Seria Matematică - Informatică - Fizică Vol. LXI, No. 1/2009.
58. Westminster news and events, <http://www.westminster.ac.uk/news-and-events/news/2011/university-of-westminster-launches-new-diy-supercomputer-saving-hundreds-of-thousands-of-pounds>, (accessed 20/08/2012).
59. Wikipedia, [http://en.wikipedia.org/wiki/Entropia,_Inc._\(company\)](http://en.wikipedia.org/wiki/Entropia,_Inc._(company)), (accessed 02/01/2013).
60. Wilkinson, B., “Grid Computing: Techniques and Applications”, 1st ed., Chapman & Hall/CRC Press LLC, Florida, USA, 2008.
61. Xtremeweb-CH, <http://www.xtremwebch.net/>, (accessed 08/01/2013)
62. Xtremeweb-HEP, <http://www.xtremweb-hep.org/>, (accessed 08/01/2013)
63. Zhou X, Depeursinge A., Niinimaki M., Geissbuhler A., and Müller H., “Grid Computing Inside Hospitals Using Virtualization Technology: A Secure Solution for Heavy Computing Tasks”, HUG Research Daym Geneva, Switzerland, 2009.

Appendix A: CPU power at QOU Computer labs

	CPU Type	Pentium 4							Dual Core	Core2 Duo						D processor	Core i3				Core i5		Celeron D	
		1	1	1	1	1	1	1		2	2	2	2	2	2		2	2	2	2	2	4		
Branch/SC	Number of Cores/CPU	1.7	2.0	2.4	2.8	3.0	3.2	3.6	1.8	2	2.2	2.33	2.4	3.0	3.6	3.06	3.1	3.2	3.3	3.1	3.2	3.06		
Jenin	Computer									48							24	24				96		
	Internet	15				12																27		
	ICT									4												4		
	Multipurpose	9								6												15		
	Visually impaired CE		4																			4		
Tubas	Computer									21								25				46		
	Internet			2		4	4				2		5									17		
	Multipurpose									1							7					8		
Tulkarem	Computer					23	2	1			7	2	2				9	23				69		
	Internet			18		10	4															32		
	ICT					1				3												4		
	Multipurpose			5						3							6					14		
Nablus	Computer	2				2					1				20		1	50				76		
	Internet	18													1		4					23		
	ICT					1				1	1							1				4		
	Multipurpose						5											7				12		
Jericho	Computer					2					1							17				20		
	Internet					10																10		
	Multipurpose					2				5												7		
Jerusalem	Computer			1							1	2		3				9				16		
	Internet					7	1															8		
	Multipurpose										8											8		
Beit- Sahour	Computer					23												18				41		
	Internet			16																		16		
	Multipurpose					5																5		
Bethlehem	Computer			2		4	4			18		15						29				72		
	Internet		19				1															20		
	ICT				1					3												4		
	Multipurpose										10						6					16		
Dura	Computer					5					19			17		2						43		
	Internet																	18				18		
	Multipurpose					2					4			1		1						8		

	CPU Type	Pentium 4						Dual Core	Core2 Duo						D processor	Core i3				Core i5		Celeron D	
Hebron	Computer					24					21							46		25		116	
	Internet																					23	
	ICT																					3	
	Multipurpose																					22	
North Gaza	Visually impaired					3																3	
	Computer										25									25		50	
Gaza	Internet							6		7										7		20	
	Computer								24											48	24	96	
	Internet							12													12	24	
Middle Gaza	Multipurpose														10							10	
	Computer							20		5										15	10	50	
	Internet							6		6											6	18	
Khan Younis	Multipurpose					10																10	
	Computer							15											23		10	48	
	Internet							15		10												25	
Rafah	Multipurpose							8														8	
	Computer																			45		45	
	Internet																				15	15	
Jenin SC	Multipurpose																				11	11	
	Computer									15								10				25	
Bedia SC	Internet	8								4							8					20	
	Computer													21								21	
Bethany SC	Internet								32													32	
	Computer						6		1				1					3				11	
Yatta	Computer									28								19				47	
	Internet	14							11													25	
	Multipurpose				1																	1	
PCs		66	23	89	1	133	117	1	94	117	231	118	7	14	71	10	3	64	409	23	165	88	1819
Cores		66	23	89	1	133	117	1	188	234	462	236	14	28	142	20	6	128	818	92	660	88	3553

Appendix B: Current Setup at QOU

B.1. General information

Since its establishment in 1991, QOU grew rapidly to become the largest academic community in Palestine, providing open education to over than 61764 students in various specializations (QOU, 2013).

QOU's campus is distributed over every Palestinian province in West bank and Gaza strip, a total of 19 branches and 3 study centers. Fig (B.1) illustrates the geographic distribution of these branches and Study Centers (SC).

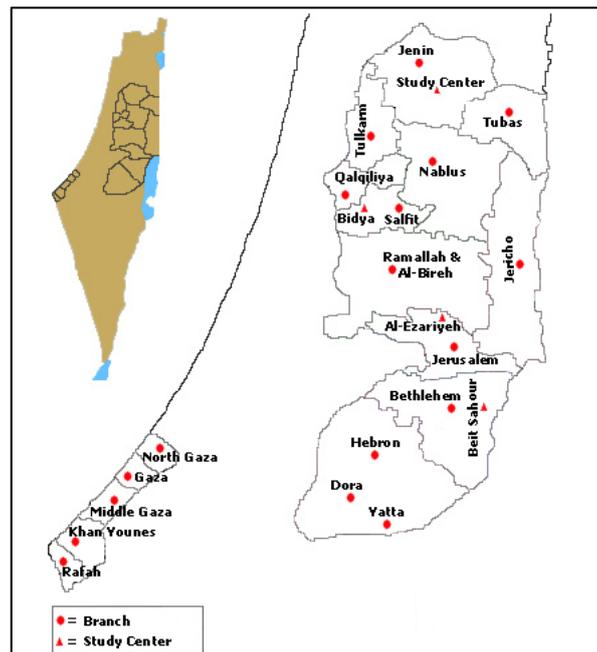


Fig. B.1: QOU branches and study centers (QOU, 2013)

B.2. Computer labs

Computer labs in QOU are classified into the following categories:

1. Computer labs: used for educational purposes, students' practices, training and assignments solving, besides some practical lectures in IT subjects.
2. Internet labs: for Internet usage for academic purposes, and for using portal services of QOU website.
3. Information and Communication Technology (ICT) labs: these labs contain electronic and communications training kits, components, and measurement instruments. These labs are used in practical training of Information and Communication Technology courses in Technology and Applied Science Faculty.
4. Multipurpose labs: these labs are used for two purposes: eLearning and browsing multimedia-aided courses.

5. Continuing Education (CE) labs: these labs are used IT training courses, especially in Branches where there are many training courses organized for the local community.
6. Computer labs for visually impaired students: these labs are equipped with special devices and computer applications to help visually impaired students.

The percentage of each lab category to the total number of labs is presented in Fig. (B.2).

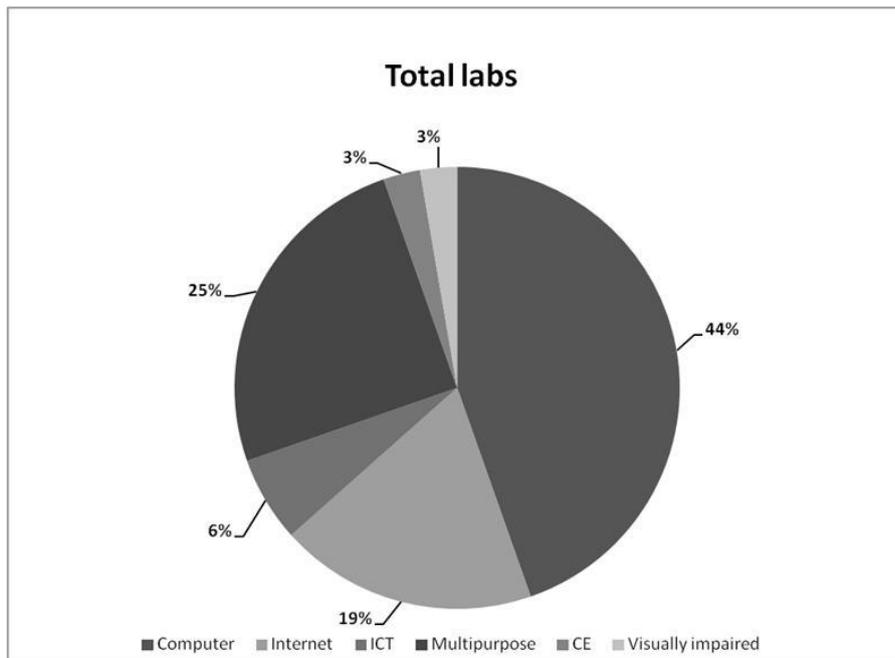


Fig. B.2: computer lab distribution in QOU

B.3. Networking

The main computer center at QOU is called "Information & Communication Technology Center (ICTC)"; it is located in Ramallah city. All data centers are located there; all branches and study centers are connected directly to ICTC via Internet Protocol Virtual Private Network (IPVPN) lines (ICTC, 2013).

Fig (B.3) presents the network topology in QOU.

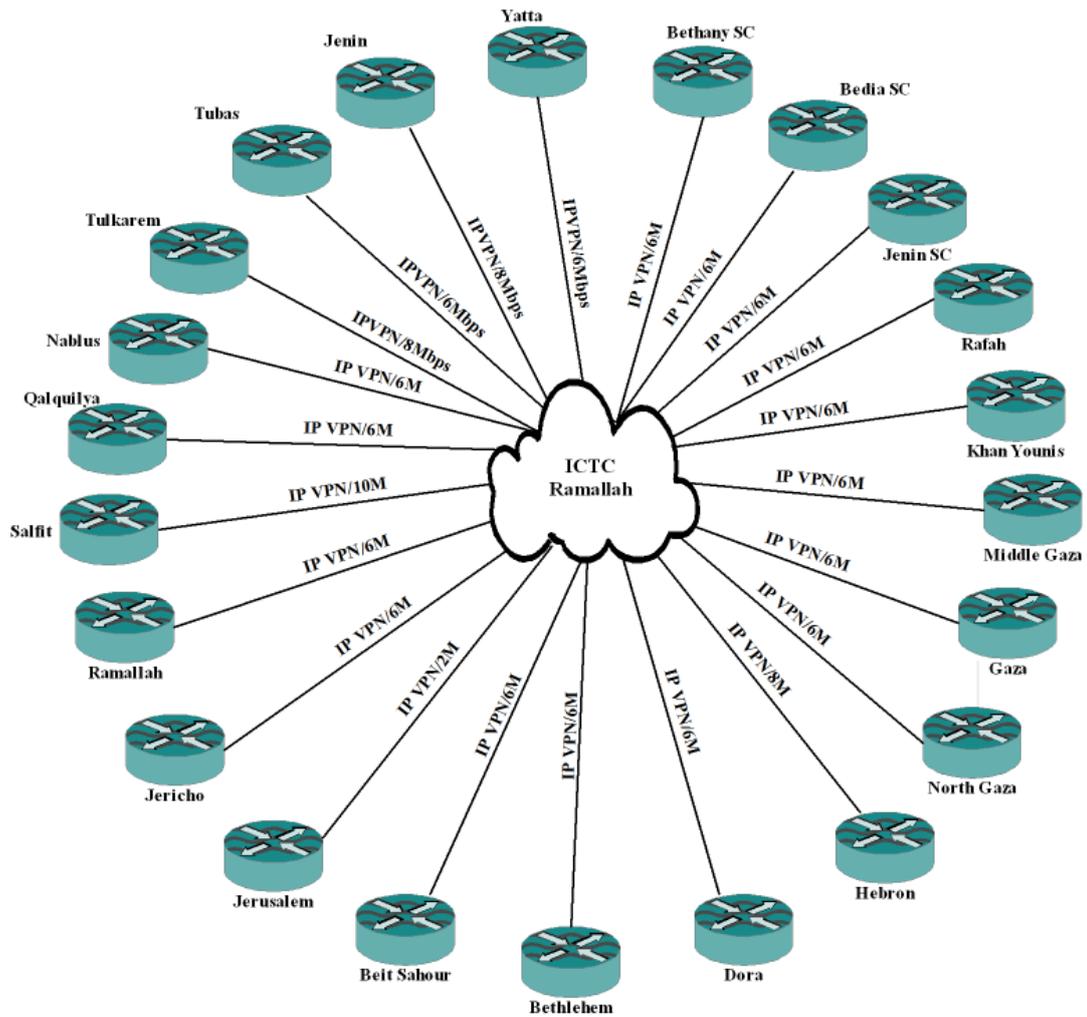


Fig. B.3: QOU network Topology

Fig (B.3) illustrates how branches and study centers are connected to ICTC; it is shown that the connection type is IPVPN with bandwidth varying from 2.0 Mbps to 10.0 Mbps. Multiprotocol Label Switching (MPLS) technology is used. MPLS offers simpler mechanisms for packet-oriented traffic engineering and multiservice functionality with greater scalability by separating routing information from packet data using labels (Semeria, 2001).

Bandwidth of each line connecting branches and SC is presented in Table (B.1).

Table B.1: communication line bandwidths between ICTC and QOU branches/SC

Branch/SC	Bandwidth (Mbps)
Jenin	8
Tubas	6
Tulkarem	8
Nablus	6
Qalqilya	6
Salfit	10
Ramallah	6
Jericho	6
Jerusalem	2
Beit-Sahour	6
Bethlehem	6
Dora	6
Hebron	8
North Gaza	6
Gaza	6
Middle Gaza	6
Khan Yunis	6
Rafah	6
Jenin SC	6
Bedia SC	6
Bethany SC	6
Yatta Branch	6

B.4. Lab PCs

According to a recent statistics provided from ICTC (ICTC, 2013), the total number of PCs located in 112 labs is 1819 PC, from which there are 1147 PCs located in computer labs and 411 in internet labs.

Table (B.2) lists every branch and SC in QOU with available PCs in each lab.

Table B.2: Total number of PCs in QOU labs.

Branch/SC	Computer	Internet	ICT	Multipurpose	CE	Visually impaired	Total PCs
Jenin	98	27	5	16	16	3	165
Tubas	45	7	0	7	0	0	59
Tulkarem	76	32	3	17	0	0	128
Nablus	78	20	4	12	0	0	114
Qalqilya	44	16	0	12	13	0	85
Salfit	25	21	0	4	15	0	65
Ramallah	80	51	3	10	0	3	147
Jericho	17	8	0	8	0	0	33
Jerusalem	16	8	0	8	0	0	32
Beit-Sahour	38	12	0	5	0	0	55
Bethlehem	64	20	4	15	0	0	103
Dora	43	18	0	12	0	0	73
Hebron	127	23	3	12	0	3	168
North Gaza	42	15	0	0	1	0	58
Gaza	98	15	0	7	0	0	120
Middle Gaza	48	20	0	14	1	0	83
Khan Yunis	52	26	0	9	1	0	88
Rafah	47	15	0	5	0	0	67
Jenin SC	25	14	0	8	0	0	47
Bedia SC	21	31	0	0	0	0	52
Bethany SC	11	0	0	0	0	0	11
Yatta	50	12	0	2	0	0	64
Total PCs	1147	411	22	183	47	9	1819

B.5. PC Specifications

QOU paces up with latest hardware specifications in computer labs and upgrades/scales up lab PCs on a regularly basis, these specifications are presented in detail in Appendix A, digest is represented below:

- **CPU frequencies:** CPU frequencies range from single-core Intel Pentium 4/1.7 GHz up to dual-core Intel Core i5/3.2 GHz.
- **Total number of CPU cores:** 3553
- **Physical Memory (RAM):** memory ranges from 128MB up to 4GB. Memory percentage is distribution is shown in Fig. (B.4):

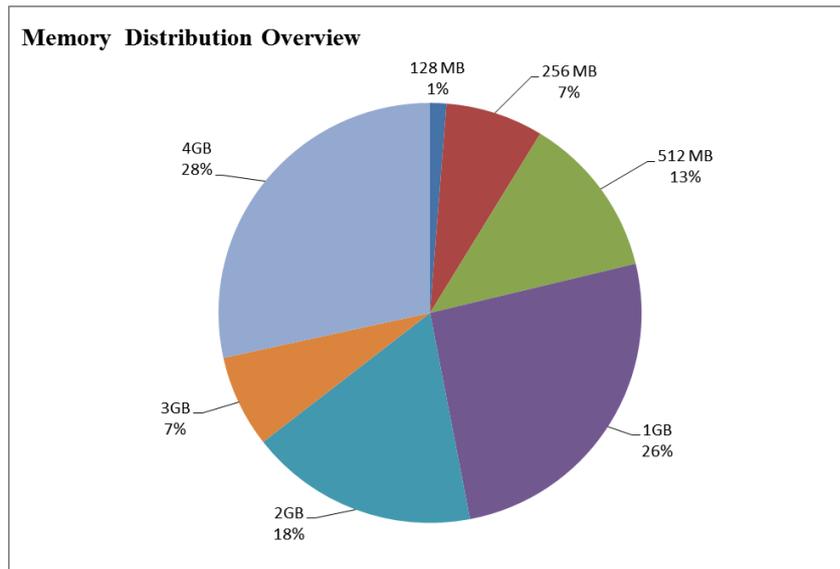


Fig. B.4: Memory distribution overview.

- **Storage:** hard drives capacities range from 40 GB up to 500GB. Storage capacity distribution percentage is shown in Fig. (B.5):

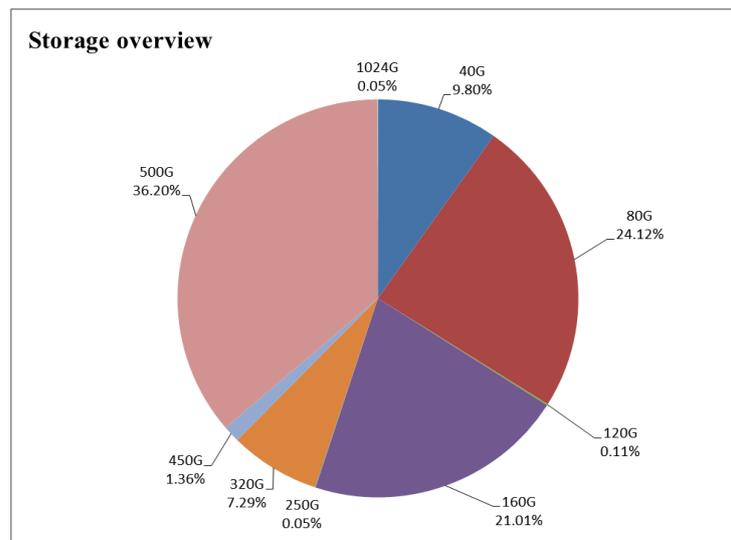


Fig. B.5: Storage capacity overview

- **Operating System:** All lab PCs have Microsoft Windows installed, two releases of Microsoft windows are deployed: Microsoft Windows XP professional, and Microsoft Windows 7 professional, Fig. (B.6) represents the operating system distribution.

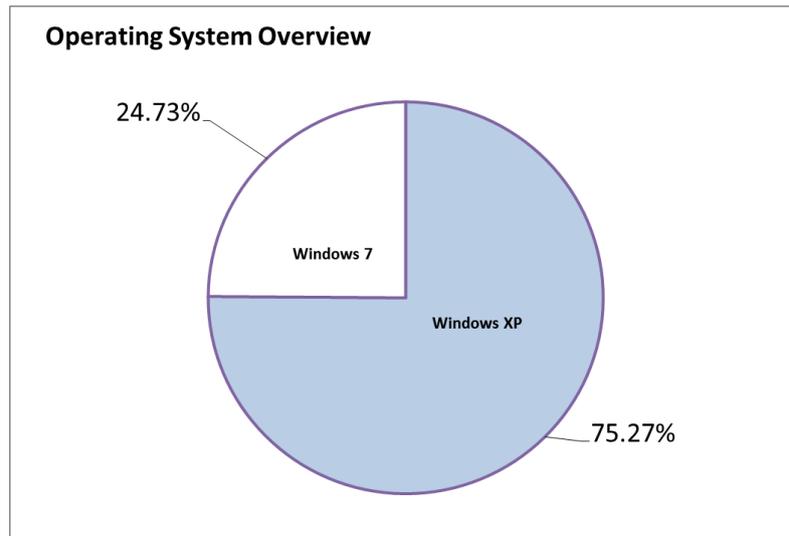


Fig. B.6: Operating system distribution overview

B.6. Authentication and Authorization

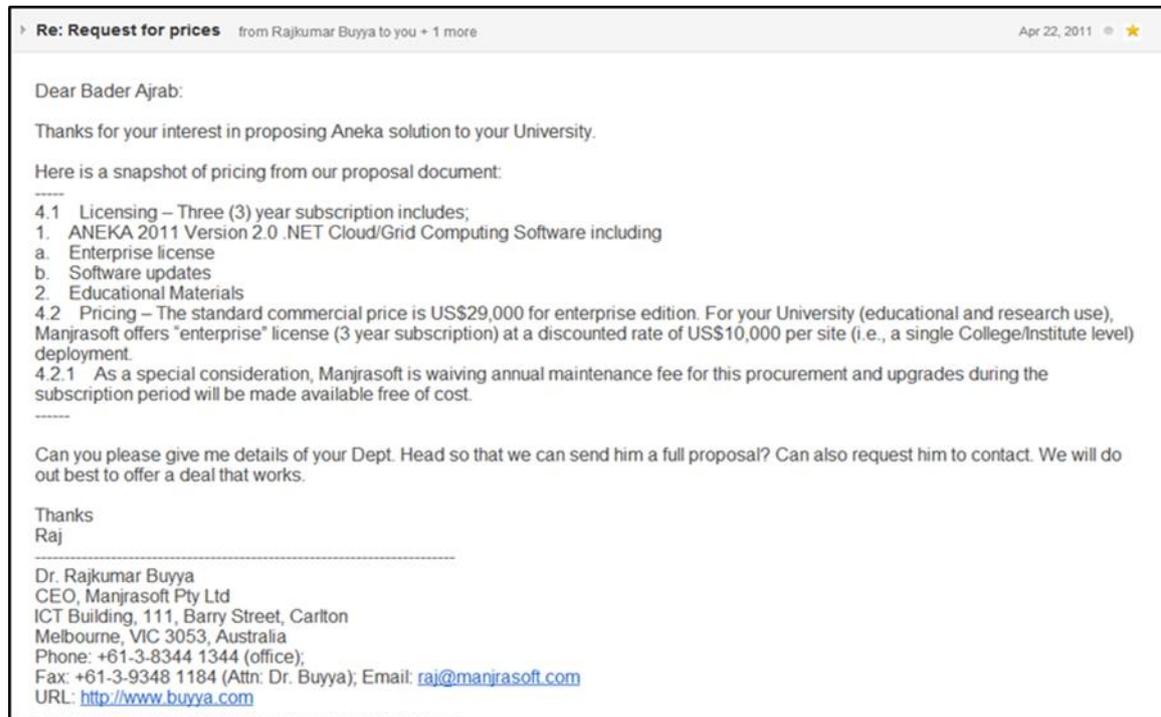
Authentication and authorization in QOU labs is handled by Domain controllers running Active Directory (AD) (Microsoft TechNet, 2013) installed on Windows Server (2003 or 2008 releases).

With AD Domain Controller, students willing to use lab PCs log on with limited privileges; they cannot install software packages and change system configuration.

Appendix C: Dr. Rajkumar Buyya's offer

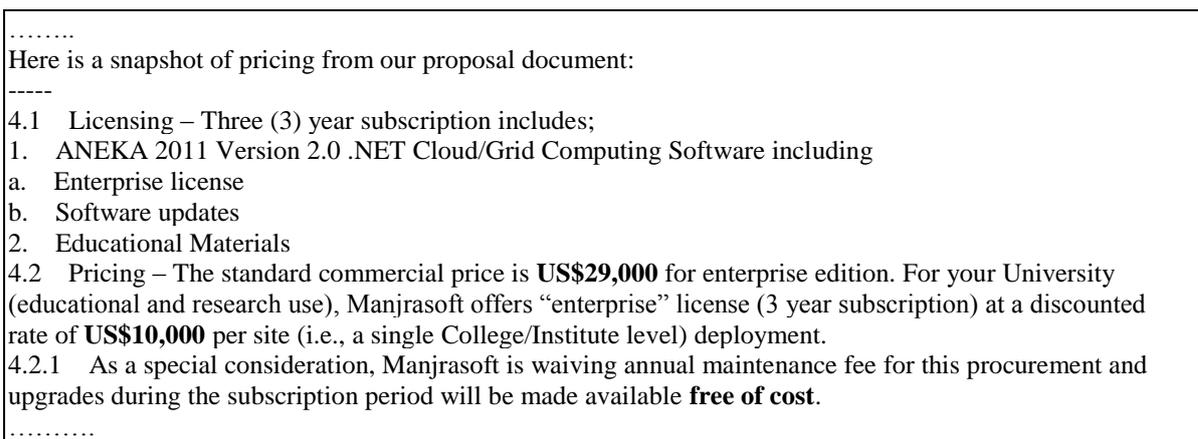
We e-mailed Dr. Rajkumar Buyya CEO of ManjraSoft Ltd., the developer of Alchemi.NET and Aneka, requesting pricing of Aneka Licensing so as to get a full feasibility study for the purpose of this research.

The figure shows the email sent to Mr. Buyya and his reply.



C.1: Snapshot of Dr.Buyya's offer

A copy of Dr. Buyya's offer is shown below:



Appendix D: Alchemi .NET installation

This appendix documents the installation, configuration, and operation of the various parts of the Alchemi framework for setting up Alchemi grids.

The latest version (version 1.0.6) of Alchemi .NET package was downloaded from Sourceforge (Sourceforge, 2012).

The downloaded Alchemi package consists of the following elements:

1. Alchemi-1.0.6-sdk-net-2.0.zip: Test examples source codes.
2. Alchemi.ExecutorExecSetup.msi: The Alchemi executor, if we want to install it as a normal Windows desktop application.
3. Alchemi.ExecutorService.msi: The Alchemi executor, if we want to install it as a normal Windows service.
4. Alchemi.ManagerExecSetup.msi: The Alchemi manager, if we want to install it as a normal Windows desktop application.
5. Alchemi.ManagerServiceSetup.msi: The Alchemi manager, if we want to install it as a windows service.
6. XPManagerSetup.msi: The Cross-Platform Manager, which is a web services interface that exposes a portion of the functionality of the Manager in order to enable Alchemi to manage the execution of grid jobs.

Both Manager and Executor can be installed and run as normal applications, or as windows services that can be controlled from "services" in control panel. We have chosen to install the Manager and the Executor in the normal mode (application mode) not windows service mode in order to gain easier control on the start/stop of experiment procedure.

D.1. Common requirements

- Microsoft .NET framework (version 2.0) (Microsoft, 2012) was installed on both the Manager and the Executor PCs.

D.2. Installing the Manager

Before installing the Manager, the following software must be installed:

- Microsoft SQL server 2005 (Microsoft, 2012).
- Microsoft Visual Studio 2005 was installed on the manager PC in order to compile example grid applications (Microsoft, 2012).
- The Manager Setup installer (Alchemi.ManagerExecSetup.msi) was used to install Alchemi Manager as a windows application. Default system administrator password (sa) was used to install the database during the installation process. Default users and group permissions were set during installation: these are three

accounts; Executor (password: executor), user (password: user) and admin (password: admin) belonging to the 'Executors', 'Users' and 'Administrators' groups respectively.

Fig. (D.1) presents the GUI of the Alchemi Manager. The GUI shown is used to start /stop the Manager Service.

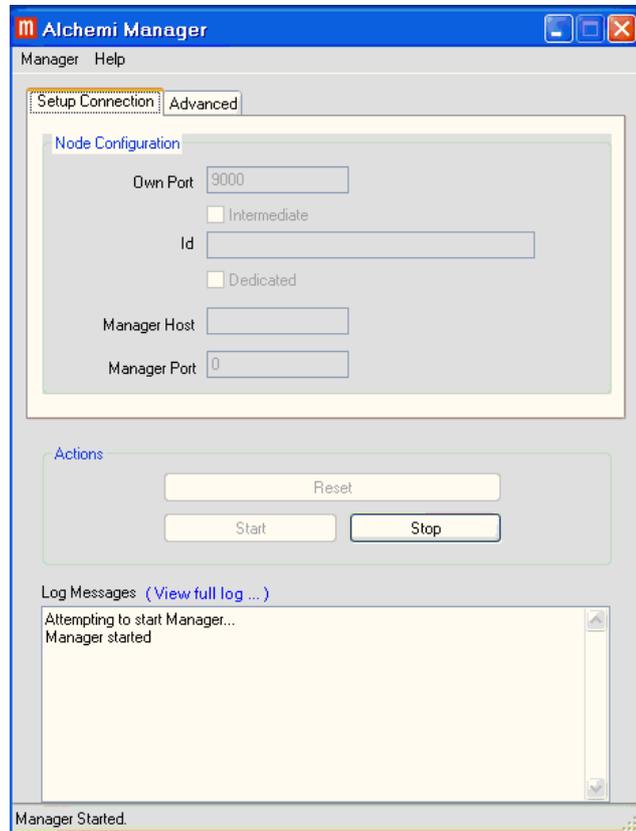


Fig. D.1: Alchemi Manager GUI

- The Manager can be run from:

Start → Programs → Alchemi → Manager → Alchemi Manager.

The database configuration settings used during installation automatically appear when the Manager is first started.

- Clicking the "Start" button starts the Manager.
- When closed, the Manager is minimized to the system tray.

Note: Alchemi .NET console was also installed on the Manager PC.

D.3. Installing the Executor

We used the Executor setup installer (Alchemi.ExecutorExecSetup.msi) to install the Executor as a windows application.

Executors were installed in the dedicated mode (computational tasks are sent from the manager to the executor automatically) in order to automate the calculation process without student intervention.

- The Executor can be run from:

Start → Programs → Alchemi → Executor → Alchemi Executor.

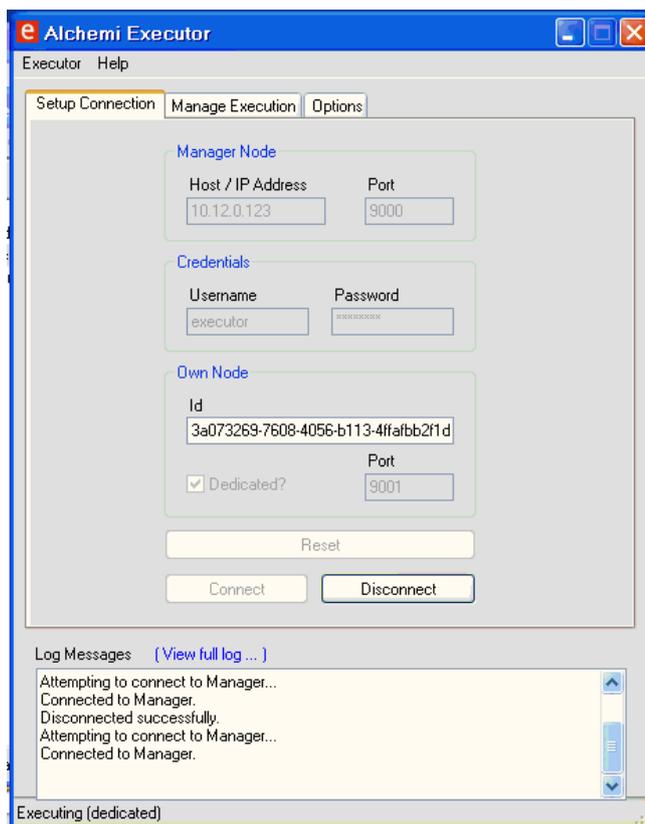


Fig. D.2: Alchemi Executor GUI

Fig. (D.2) presents the GUI of the running Alchemi Executor in the dedicated mode.

- The host and port of the Manager to connect to are entered.
- Clicking the "Connect" button connects the Executor to the Manager.
- When closed, the Executor sits in the system tray

Appendix E: BOINC installation

This appendix documents the installation process of BOINC server, creating and running a BOINC project, then the installation of BOINC clients.

E.1. Installing BOINC server:

This step consists of two stages: choosing hardware then installing software (operating system, BOINC server and its dependencies), see Fig. (E.1).



Fig.E.1: Installing BOINC server block diagram

E.1.1. Choosing hardware:

Although a dedicated server is needed to run a BOINC-based local desktop grid, the BOINC server requires low cost server hardware. The recommended hardware configuration for more than 1000 connected PCs (SZTAKI, 2012):

- Processor: 2000 MHz Intel Pentium
- Hard disk: 100GB
- Internet: 100Mbit/sec

And the recommended hardware requirements up to 100 connected PCs are (SZTAKI, 2012):

- Processor: Intel Pentium 1000 MHz
- Hard disk: 60 GB
- LAN: 100 Mbps

In this experiment, the total number of PCs involved is 43, and the above requirements are suitable.

To insure portability of BOINC server, we chose to install the server on a 64-bit virtual machine using "Oracle VM Virtualbox 4.2.8" package (Oracle, 2012) having the following hardware configuration:

- Processor: Intel Core2 Duo T7300/2.0 GHz
- Hard disk: 60 GB
- Memory: 2.0 GB
- LAN: 100 Mbps

E.1.2. Installing Software:

The software requirements installation processes are shown in Fig. (D.2)

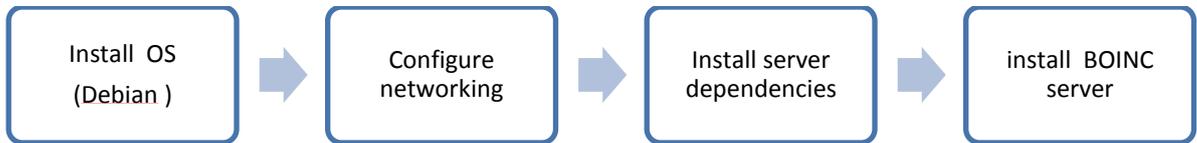


Fig.E.2: Software installation block diagram

1. Operating System installation: BOINC server runs almost on any up-to-date Unix or Linux variant machine (BOINC 2012). Debian version 6.0 – 64bit Linux (Debian, 2012) was installed with minimum modules.

2. Network configuration: The range of IPs in Jerusalem branch is 10.12.0.1 ~ 10.12.0.254, and for Bethany study center 10.13.0.1 ~ 10.13.0.254.

The created Debian machine was located in Jerusalem branch and was given following network configuration:

- Static IP 10.12.0.100
- Subnet mask : 255.255.0.0
- Default gateway: 10.12.0.1
- Host name: debian6

3. BOINC server dependencies installation: BOINC server needs the following components to run:

- make 3.79+, m4 1.4+, libtool 1.5+,
- autoconf 2.58+, automake 1.8+, GCC 3.0.4+ pkg-config 0.15+
- Python 2.2+ with MySQLdb module 0.9.2:+
- MySQL 4.0.9 or higher (with mysql-dev(el), and mysql-client)
- SQLite 3.1 or higher (packages sqlite-dev(el) and SQLite)
- Apache web server with mod_ssl and PHP5+
- PHP5 with cli support and the GD and MySQL modules (packages php5-cli and php5-gd)
- OpenSSL version 0.9.8+

These components were installed using Advanced Packaging Tool (*apt-get install*) command (Debian, 2012).

Note: email system such as "exim4" is needed for BOINC if it is run in a public environment, but in this case (local grid) there is no need to set up an email system.

4. BOINC server setup:

The BOINC server package can be installed using *apt-get* command. The following line was added to the */etc/apt/sources.list* file (SZTAKI, 2012):

```
deb http://www.desktopgrid.hu/debian/squeeze szdg
```

Afterwards, the following command is used to install the BOINC server:

```
apt-get install boinc-server
```

Note: this approach also installs the BOINC server dependencies automatically and no need to install them manually as in step 3 above.

E.2. Creating and running a BOINC project

A BOINC project is the environment under which the grid application runs (BOINC, 2012), project installation block diagram is shown in Fig (E.3):

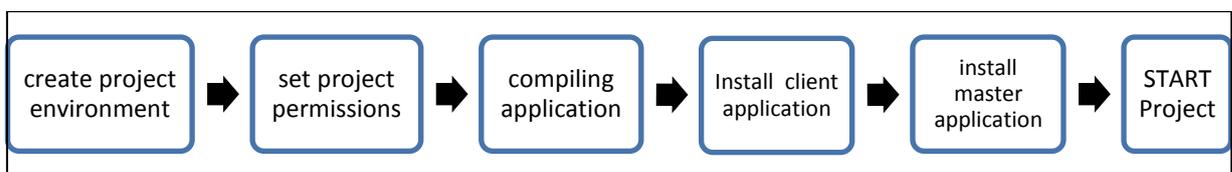


Fig.E.3: Creating a sample BOINC project

We ran the built-in `make_project` script as follows:

```
./make_project --url_base http://10.12.0.100 --test_app test
```

This command created a project named `test` and:

- Created the project directory and its subdirectories (`~/projects/test/..`) with required permissions.
- Created the project's encryption key.
- Created and initialized the MySQL database.
- Copied source and executable files into (`~/projects/test/`) folder.
- Generated the project's configuration file `config.xml`.

With the `--test_app` option, the project will compile a built-in test application and required daemons to generate and handle work for it. The test application is an example single-thread native BOINC application. This example application converts an input text file from lower case to upper case; source code is found in `/usr/lib/boinc-server/apps/upper_case`.

The BOINC server package comes with a web interface to simplify monitoring and control. The web page can be reached from the following URL in our case: <http://10.12.0.100/test/>

in order to control access to the web page we copied `test.htpd.conf` into the apache server root dir `/etc/apache2/httpd.conf`.

We also generated a username/password file for the administrative web interface using:

```
htpasswd -cb ~/projects/test/html/ops/.htpasswd bajrab Passw0rd123
```

The administrative page can be accessed from http://10.12.0.100/test_ops then typing in the above username and password (*bajrab/Passw0rd123*).

For testing purposes, we need to run the example application for a long period of time; this was done by running the *crontab -e*, and adding an entry to run the project's cron script:

```
*** 0,5,10,15,20,25,30,35,40,45,50,55 test/bin/start --cron
```

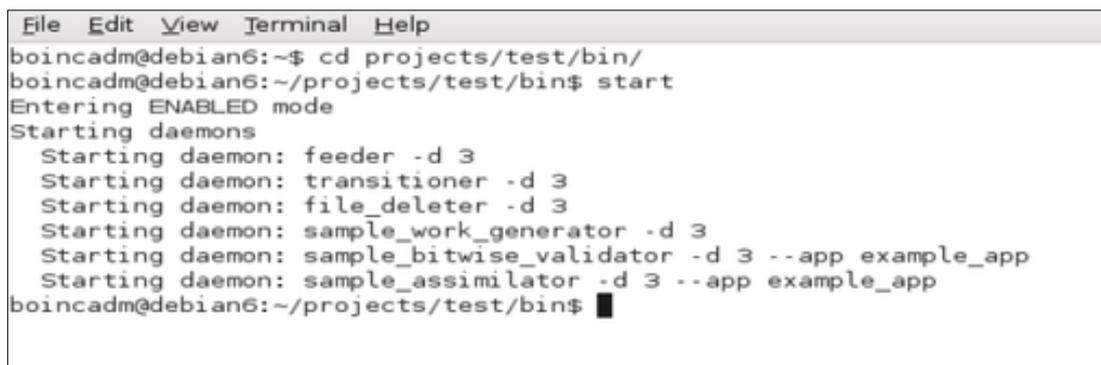
~/projects/test/bin/xadd is run to add platform and application records to the BOINC database.

Also *~/projects/test/bin/update_versions* is run in order to release new application versions. It creates database entries and copies files to the download directory.

The project is ready to run, running the project is done using the following command

```
~/projects/test/bin/start.
```

This command starts all BOINC server daemons related to the test project: feeder, transitioner, file_deleter, work generator, validator and assimilator, Fig. (E.4) illustrates the *start* command output.

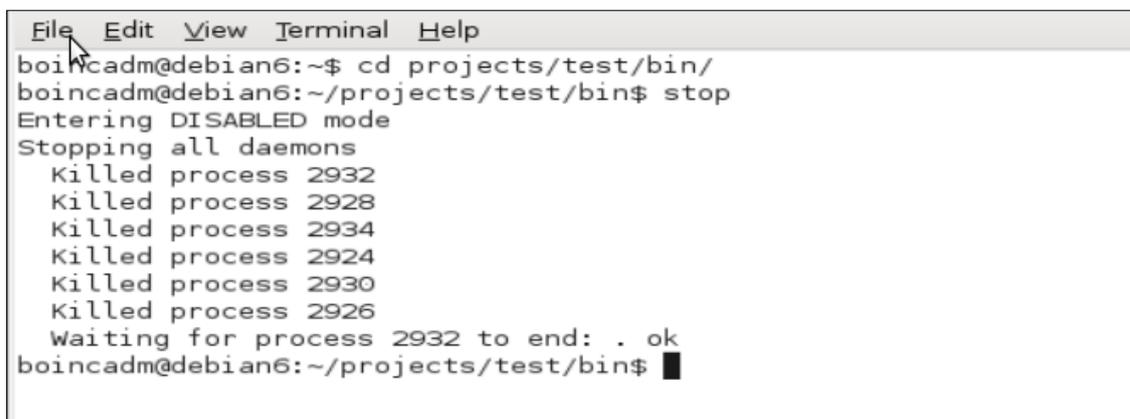


```
File Edit View Terminal Help
boincadm@debian6:~$ cd projects/test/bin/
boincadm@debian6:~/projects/test/bin$ start
Entering ENABLED mode
Starting daemons
Starting daemon: feeder -d 3
Starting daemon: transitioner -d 3
Starting daemon: file_deleter -d 3
Starting daemon: sample_work_generator -d 3
Starting daemon: sample_bitwise_validator -d 3 --app example_app
Starting daemon: sample_assimilator -d 3 --app example_app
boincadm@debian6:~/projects/test/bin$ █
```

Fig.E.4: Starting BOINC project

The project is now ready to send work units to clients and receive results from them.

We can stop the sample test project using the *stop* command as shown in Fig. (E.5)



```
File Edit View Terminal Help
boincadm@debian6:~$ cd projects/test/bin/
boincadm@debian6:~/projects/test/bin$ stop
Entering DISABLED mode
Stopping all daemons
Killed process 2932
Killed process 2928
Killed process 2934
Killed process 2924
Killed process 2930
Killed process 2926
Waiting for process 2932 to end: . ok
boincadm@debian6:~/projects/test/bin$ █
```

Fig.E.5: Stopping the BOINC project

E.3. Deploying BOINC Clients

The last part in the installation process after the BOINC server is up and running and the project is ready to handle jobs, is installing clients in lab PCs. Fig (E.6) shows a block diagram for the deployment process:

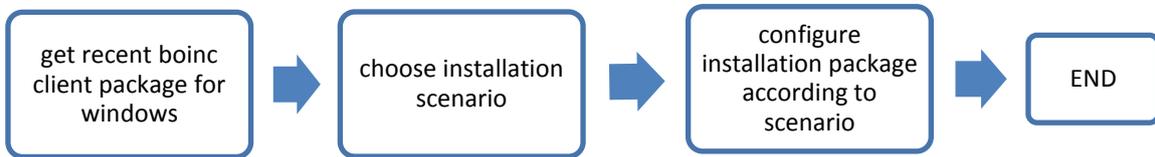


Fig.E.6: Deploying BOINC client block diagram

1. Downloading client:

The BOINC client was downloaded from <http://boinc.berkeley.edu/download.php> .
The last stable version was v7.0.28, size is 8.01 MB, filename:

boinc_7.0.28_windows_intelx86.exe , it supports windows 32bit (2000/XP/Vista/7).

2. Client Deployment:

We installed the BOINC client on one PC and attached it manually to the test project, this is illustrated in Fig. (E.7).

During installation process, we were prompted to create a project user, a user is created:

- User name: testgrid
- Email: bajrab@qou.edu
- Password: pa\$\$w0rD123

This user will be used to connect all experiment PCs.

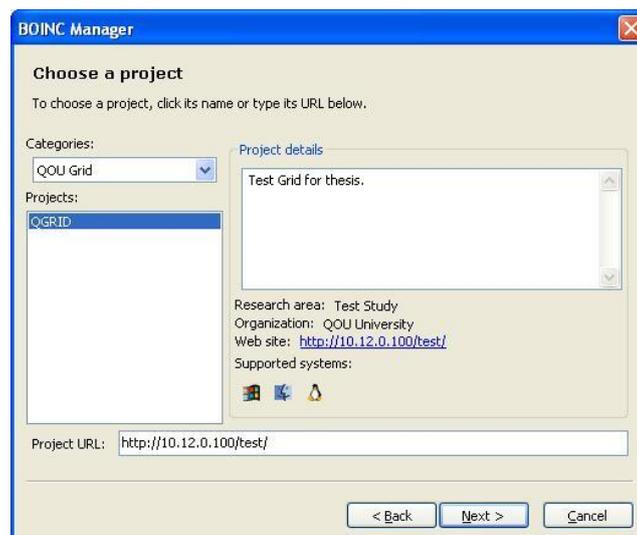


Fig.E.7: Attaching BOINC client to the test project

The experiment was successful, and the BOINC client started to download work units from BOINC server, processed them, and uploaded results back to the server. A screenshot of BOINC manager GUI is presented in Fig. (E.8)

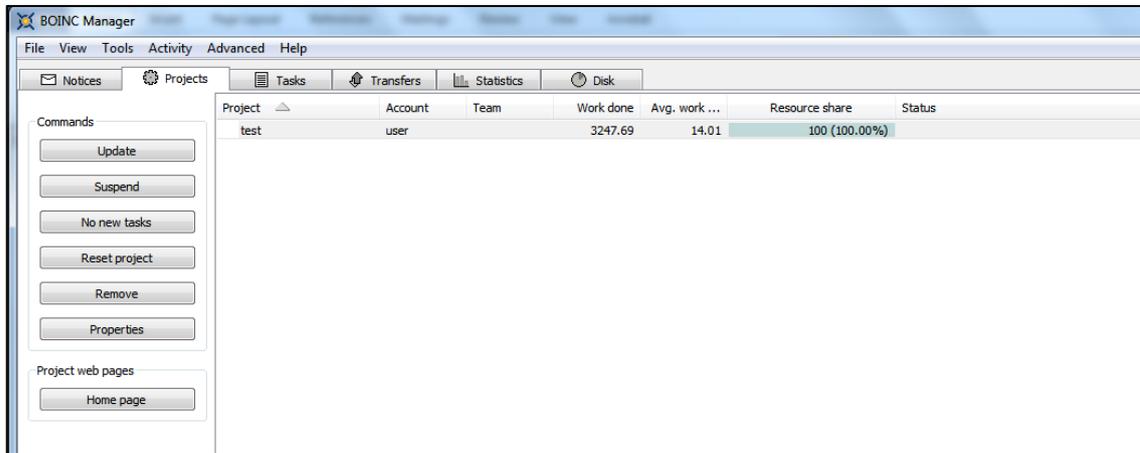


Fig.E.8: BOINC client attached to test project

Task status is presented in Fig. (E.9).

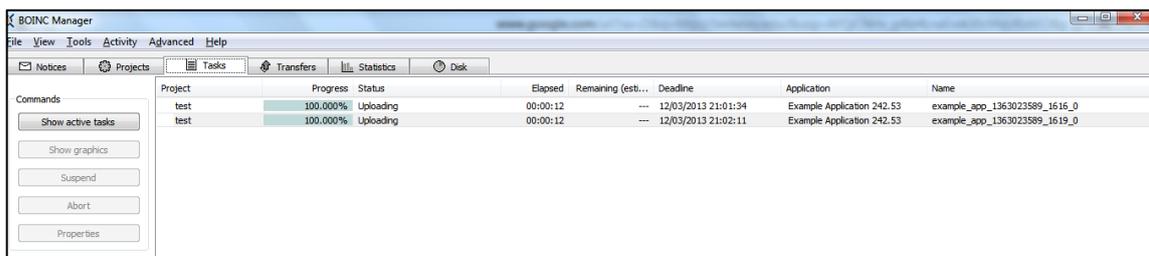


Fig.E.9: BOINC client uploading tasks

Fig. (E.9) presents the task tab that shows the task status of the BOINC client.

Figure (E.10) shows tasks being obtained from server.

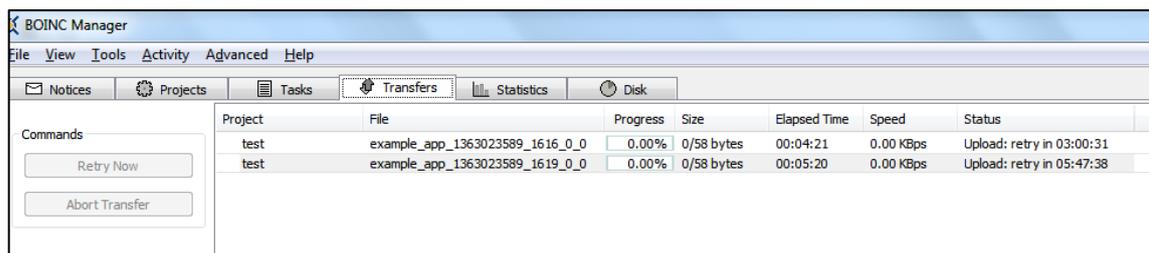


Fig.E.10: BOINC uploading finished jobs

The finished jobs or tasks are returned back to server as shown in Fig. (E.10)

The next step is to deploy BOINC client on experiment PCs.

The BOINC Client Software can be deployed on a large number of PCs in one of three methods: a single-user installation, shared installation, or service installation (BOINC wiki, 2013)

- In the single-user installation case, the BOINC manager only runs when the user that installed it logs in. Other users can't run the BOINC Manager .
- Shared installation allows any user that logs in to run BOINC manager.
- As for the service installation, the BOINC core client runs as a service that starts at boot time.

For the QOU campus environment, the BOINC client must be deployed to meet the following restrictions:

- The BOINC Client must run all the time (even when no one is logged in).
- Project attachment must be automatic.
- BOINC manager must be disabled: no user intervention is required in client configuration (the student cannot attach to or disconnect from a project, cannot create a new user, or start/stop/pause/exit the BOINC client).
- Screen saver must be disabled: no screensaver is needed because it looks strange for students, they may think the PC is occupied/busy and choose not to use it.
- To insure security, no internet connection is required for the BOINC client since it runs within the university's LAN.
- For a large number of PCs, deployment must be performed automatically, not manually.

The above restrictions can be achieved by obtaining and customizing the MSI (Microsoft installer) version of BOINC, not the *.exe* one.

We ran the *boinc_7.0.28_windows_intelx86.exe* with the `"/a` " option to get the MSI version of BOINC (BOINC.msi) in order to customize it. *boinc.msi* was edited using the advanced installer tool v.10 (Caphyon, 2013), and the following properties were altered:

- ENABLEPROTECTEDAPPLICATIONEXECUTION2 from 0 to 1
- ENABLESCREENSAVER from 1 to 0

Setting ENABLEPROTECTEDAPPLICATIONEXECUTION2 to 1 should cause the installer to install BOINC as a service.

Setting ENABLESCREENSAVER to 0 should prevent the installer from setting the screensaver by default.

After the *boinc.msi* file is ready, we copied the *account_10.12.0.100_test.xml* file from the BOINC data folder of a previously installed BOINC project to the installation package folder *"CommonAppData"*. This file contains URL for test project with encrypted authentication information.

A screenshot of the *account_10.12.0.100_test.xml* file is shown in Fig. (D.11).

```
<?xml version="1.0"?>
<account>
  <master_url>http://10.12.0.100/test/</master_url>
  <authenticator>1a8546cf856b777a79a5f8c493786e4d</authenticator>
  <project_name>test</project_name>
  <project_preferences> </project_preferences>
</account>
```

Fig.E.11: BOINC project configuration file on the client side

Fig. (E.11) shows the *account_10.12.0.100_test.xml* file, note the encrypted authenticator used to connect core client to BOINC server.

The deployment process can be automated in Active Directory domain using Group Policy (GP). A package was assigned per-machine, it was automatically and silently installed when the target client computers started.

Appendix F: Glossary

<i>AD</i>	Active Directory
<i>APIs</i>	Application Programming interfaces
<i>BAM</i>	BOINC account manager
<i>BOINC</i>	Berkeley Open Infrastructure for Network Computing
<i>CE</i>	Continuing Education
<i>CPU</i>	Central Processing Unit
<i>DG</i>	Desktop Grid
<i>DTW</i>	Dynamic Time Warping
<i>FCFS</i>	First Come First Served
<i>GFLOPS</i>	Giga Floating Point Operations per Second
<i>GPU</i>	Graphical Processing Unit
<i>GUH</i>	Geneva University Hospital
<i>GUI</i>	Graphical User Interface
<i>HEI</i>	Higher Education Institutions
<i>HPC</i>	High Performance Computing
<i>HTC</i>	High Throughput Computing
<i>ICT</i>	Information and Communication Technology
<i>ICTC</i>	Information & Communication Technology Center
<i>IPVPN</i>	Internet Protocol Virtual Private Network
<i>LAN</i>	Local Area Network
<i>MPLS</i>	MultiProtocol Label Switching
<i>OCR</i>	Optical Character Recognition
<i>OGR</i>	Optimal Golomb Ruler
<i>OS</i>	Operating System
<i>P2P</i>	Peer-to-Peer
<i>PC</i>	Personal Computer
<i>PRC</i>	Public Resource Computing
<i>QoS</i>	Quality of Service
<i>QOU</i>	Al-Quds Open University
<i>RAM</i>	Random Access Memory
<i>SC</i>	Study Center
<i>SDK</i>	Software Development Kit

<i>SQL</i>	Sequential Query Language
<i>UNIVA</i>	United Devices Inc.
<i>VCSC</i>	Virtual Campus Supercomputing Center
<i>VO</i>	Virtual Organization
<i>WAN</i>	Wide Area Network
<i>WU</i>	Work Units

الحوسبة الشبكية للحواسيب الشخصية في مؤسسات التعليم العالي

إعداد: بدر أحمد الأجر

إشراف: د. ليبي عرفة

ملخص

لقد شهد العقد الماضي ثورة تكنولوجية شاملة كان لها بالغ الأثر في نمط حياة البشر، حيث أنها سخرت موارد هائلة (من سرعة معالجة وسعة ذاكرة وسعة تخزينية وسرعة اتصال بشبكة الإنترنت) على أجهزة الحواسيب الشخصية. هذه الموارد لم تكن لتتوفر حتى على أجهزة الخوادم قبل أوقات ليست ببعيدة.

وقد فاضت هذه الموارد عن حاجات المستخدمين العاديين بحيث أصبحت موارد أجهزة الحواسيب الشخصية - وخاصةً المعالج - تكون خاملة معظم الوقت، فنشأت من هنا فكرة الحوسبة الشبكية المحلية لأجهزة الحواسيب الشخصية PC Grid Computing. ونشأت كذلك فكرة الحوسبة التطوعية Volunteer Computing بحيث يستطيع مستخدم الحاسوب أن يتبرع بالقدرات الفائضة لحاسوبه للمساهمة في أبحاث طبية وفلكية وكيميائية تخدم البشرية دون أدنى تأثير على استخدامه الشخصي للحاسوب.

ومن جهة أخرى، تحتاج مؤسسات التعليم العالي إلى قدرة حسابية عالية لإجراء الأبحاث العلمية والمحاكاة وتحليل البيانات الضخمة، ومثل هذه القدرة الحسابية تحتاج إلى أجهزة حواسيب فائقة supercomputers، وهي باهظة الثمن بحيث لا تستطيع معظم الجامعات الفلسطينية توفيرها. فنشأت الحاجة إلى تسخير الموارد الخاملة لأجهزة الحواسيب الشخصية المتوفرة في مختبرات الحاسوب في الحرم الجامعي لتتضافر معاً لتشكيل جهاز حاسوب ظاهري فائق القدرة بتكلفة منخفضة جداً.

لتحقيق هذا الأمر بشكل عملي قام الباحث ابتداءً بقياس معدل استخدام المعالجات في أجهزة الحواسيب الشخصية في مختبرات الحاسوب في فرع القدس ومركز خدمات العيزرية التابعين لجامعة القدس المفتوحة (ج.ق.م). شملت الدراسة 14 جهازاً، تم مراقبة أدائها في أيام الدوام الرسمي للجامعة من 08:00 حتى 15:30. وقد خلصت النتائج إلى أن معدل الاستخدام الفعلي اليومي للمعالج في هذه الأجهزة لا يتجاوز 10% في 90% من الوقت.

ثم قام الباحث ببناء نظام الحوسبة الشبكية من خلال تجربة حزميتين مفتوحتي المصدر في فرع القدس ومركز خدمات العيزرية كعينة استكشافية: إحداهما أداة (الكيمي.نت Alchemi .NET) من تطوير جامعة ميلبورن- أستراليا، والأخرى هي حزمة بوينك BOINC من تطوير جامعة بيركلي - الولايات المتحدة الأمريكية وهي مصممة أصلاً للحوسبة التطوعية ولكن تم تخصيصها لتخدم الحوسبة الشبكية المحلية.

وقد تبين للباحث أن تقنية الكيمي Alchemi وبالرغم من سهولة إعدادها وتشغيلها وأدائها العالي وبالرغم من قدرتها على تخفيض وقت تنفيذ البرامج المعقدة، إلا أن هذه التقنية لا يمكنها توزيع المهام الحوسبية على الشبكات الفرعية في فروع الجامعة المختلفة، كما أن الكيمي أظهر أداءاً سيئاً عند تجربته على نظام ويندوز 7 مقارنة بويندوز XP .

وقد تبين للباحث كذلك أن الكيمي يسبب ضغطاً على الشبكة حيث أن معظم وقت التنفيذ يضيع في التواصل الشبكي بين المدير Manager والمنفذين Executors. ناهيك عن عدم وجود آلية أو خيار فيه لتحديد نسبة استخدام موارد المعالج المتوفرة، فهو يمتص هذه الموارد كلها مما يسبب ارتفاعاً في درجة حرارة المعالج وبالتالي زيادة التكاليف في الطاقة الكهربائية.

وبالنسبة لتقنية بوينك، فإن خادم بوينك لا يعمل إلا على منصة لينوكس Linux وهذا يعني أنه لا بد من تخصيص جهاز خادم لهذا الغرض ويعني كذلك توفر مهارات عالية في العمل كمدبر نظام للينوكس، وهذه شكل صعوبة للباحث عند تشغيل بوينك مقارنةً بالكيمي.

من جهة أخرى فإن خادم بوينك يحتوي لوحة تحكم تمكن مدير النظام بتحديد العديد من الخيارات مثل نسبة استخدام العميل client للمعالج ووقت استخدامه.

وقد خلصت تجربة بوينك التي شملت 43 (71 نواة معالج) جهاز حاسوب شخصي في مختبرات ج.ق.م. موزعة بين فرعي القدس والعيزرية إلى التمكن من حصد 106 جيجافلوب/ث مع الأخذ بعين الاعتبار أنه تم تحديد استخدام المعالج إلى 50% فقط للتخفيض من استهلاك الطاقة، وحسب هذه النتيجة فيمكننا التنبؤ أن 3550 نواة معالج في جميع مختبرات ج.ق.م. ستنتج ما يقارب 5.27 تيرافلوب/ث مجانية ضمن محددات هذه الدراسة.

وقد تمكن الباحث من تقدير مدى تأثير نقل البيانات بين الخادم server والعملاء clients ليكون 4.55% للخط الذي سرعته 6 ميجابت/ث.

وعند فحص أداء عميل بوينك BOINC client تحت بيئة ويندوز 7 فإنه قد أظهر أداءً منافساً فاق فيه نظيره تحت بيئة ويندوز XP ضمن نفس محددات التجربة، حيث أنه كانت نسبة التحسن في الأداء تحت بيئة ويندوز 7 تصل من 19% - 57% قياساً لويندوز XP.

كلمات مفتاحية: الحوسبة الشبكية، الحوسبة التطوعية، الحواسيب الشخصية، مختبرات الحاسوب، بوينك، ألكيمي، جامعة القدس المفتوحة، أنوية.