Campus Grid Deployment with Automation

Okta Nurika[1], Low Tang Jung[2], Ahmed Abba Haruna[3]

[1]Green Education Centre, Faculty of Computing and Meta-Technology, Universiti Pendidikan Sultan Idris (Sultan Idris Education University), Tanjung Malim, 35900 Perak, Malaysia, oktanurika@meta.upsi.edu.my
[2]Computer and Information Sciences Department, Universiti Teknologi PETRONAS, 32610 Seri Iskandar, Perak, Malaysia, lowtanjung@utp.edu.my
[3]College of Computer Science and Engineering, University of Hafr Al Batin, Saudi Arabia, aaharuna@uhb.edu.sa

Campus grid is a feasible deployment of grid computing since campus environment is equally controlled and the managerial permission is simpler than any other industries. The usability of grid computing is also potentially high, because of the numerous demands from students or researchers in need of high-end computational power and data storage. However, automated efficient way of campus grid platform deployment has never been disclosed, therefore we propose a methodology to deploy a campus grid with automation based on the desktop-grid architecture. Some related issues and challenges that are currently being addressed, with improvements further to be explored, are presented in this paper. Large scale campus grid deployment in this campus involving multiple computer labs and hundreds of computers in total was accomplished, by combining both automation scripts and manual intervention. The chosen campus grid software system is Berkeley Open Infrastructure for Network Computing (BOINC). This practice is expected to guide future BOINC campus grid administrators to establish a working grid computing system, in order to provide grid-based computing and storage resources for running especially heavy simulation programs.

Keywords: AutoIt, BASIC script, BOINC, Campus Grid, Centralized Software Deployment, EMCO, Malaysian Grid.

1. Introduction

Grid Computing is an infrastructure that brings out integrated computing resources and services through network connections as defined by [1]. According to [2], generally there are two types of grid infrastructure, namely computational and data grid. Computational grid is a hardware and software framework to deliver dependable, consistent, pervasive, and affordable access to high-end computation capability [3], while data grid is "an infrastructure that manages huge amount of data files and provides resources across geographically distributed collaboration" [4].

The growth of Grid Computing trend in Malaysia has been accumulating since 2005, where

the Malaysian government has put Grid Computing development under technology agenda of Ninth Malaysia Plan [5]. To prepare for this agenda development, MYREN (Malaysian Research & Education Network) has built a high speed network that chained twelve Malaysian Universities [6], which include Universiti Malaya (UM), Universiti Putra Malaysia (UPM), Universiti Kebangsaan Malaysia (UKM), Universiti Sains Malaysia (USM), Universiti Teknologi Malaysia (UTM), Universiti Malaysia Sarawak (Unimas), Universiti Malaysia Sabah (UMS), Multimedia University (MMU), Universiti Teknologi PETRONAS (UTP), Universiti Tenaga Nasional (Uniten), Universiti Utara Malaysia (UUM), and Universiti Teknologi Mara (UiTM).

In June 2005, MYREN then held a roadshow where Universiti Malaya became the first Malaysian University to build campus grid infrastructure. This campus grid project was then named GERANIUM (Grid-Enabled Research Network and Infostructure of University of Malaya) [7].

Hundreds of other institutions have followed suit around the world, however none has shared the automated procedure to deploy campus grid platform, therefore we propose such method in this paper that would assist future campus grid developers. We deployed our method of automation at Universiti Teknologi PETRONAS (UTP) by utilizing the existing computers in the departmental labs. Most of these computers are based on Windows operating system. The labs are either located in the same or different buildings with different network segments, but are able to communicate with each other through routers.

The outline of this chapter is as follows. Section 2 describes the grid platform chosen by UTP campus grid team and the big picture of how the platform works. Section 3 informs the existing gaps within worldwide campus grid deployments. Section 4 explains the automated way of file sharing in our campus grid. Section 5 discusses our proposed automation to install and configure necessary client-side software and its vital issues. Section 5 also presents the workability test of the campus desktop-grid infrastructure. Finally, Section 6 concludes the overall performance of our campus grid and its potential improvements in the future.

The Choice of Grid Platform

Based-on the properties of the existing computers to be deployed as compute nodes, the desktop-grid software platform (such as Berkeley Open Infrastructure for Network Computing BOINC) [8], can perform the grid operations on Windows and be efficient enough to run 100Mbps network card speed. BOINC was there chosen to fulfill this requirement.

BOINC was actually built for volunteer and desktop- grid computing [8]. It is volunteer based be-cause the grid computing participants are willing to attach their computers to one or more grid projects. In this case, the participant will register an account at the BOINC server then install the BOINC client on their computer, so that it can be attached to one or more BOINC server's projects [9]. The participants can configure on the BOINC Client which projects they want to attach to and how their computer resources can be shared by the project tasks [8].

Since BOINC is designed for grid and runs under Internet or public interconnection, hence it is suitable to run in campus grid environment, because campus merely utilizes local network without having to go all the way through the Internet.

Furthermore, BOINC provides several methods to enable applications (proprietary or open-

source) to run on BOINC infrastructure. Applications written in C, C++, or Fortran can run in BOINC framework without any modification [8]. In this case the application can be wrapped using the BOINC Wrapper or GenWrapper (Generic Wrapper) for more functionalities [10]. For the BOINC Wrapper, it will understand how to run the application by reading an XML file that holds the application's attributes such as input, output, the fraction of job done by each task, etc [10]. This information is necessary since the BOINC server will distribute the tasks to the clients/compute nodes and based on the output generated by the clients, it will validate and assimilate the results [11].

GenWrapper is an easy wrapper and supports functionality like complex control flows (loops, branches, etc). Instead of reading an XML file to view the application's attributes, it reads a more generic POSIX-like shell script and use built-in commands like tar, awk, sed, zip, etc. [12]. GenWrapper tackles the limitations of the traditional BOINC Wrapper that reads a stiff XML file which can only describe the legacy tasks one after another. Gen-Wrapper is capable to execute an arbitrary set of legacy applications [13].

For the our campus grid, GenWrapper is chosen to wrap application binaries which are mostly compiled for Windows 32-bit machines. One of the first applications wrapped using GenWrapper was the 2D EM (2 Dimension Electro-Magnetic) simulation program that generates an output file of 146MB in size.

Application's source code can also be altered by including BOINC API (Application Programming Interface) which is a set of C++ functions in the C code's header. A richer API called DC-API is also available to provide more adapt-able API for BOINC-able applications to work on multiple grid environments [15]. To adapt to multiple grid environments, DC-API only supports a restricted master-worker programming model. Currently a DC-API sample application named "uppercase" has been tested to work on our campus grid.

Related Campus Grid Deployments Gaps

A combinatorial large scale software deployment experiment was built by collaborating JMX (Java Management eXtension) with ProActive [18]. Since they applied their framework under Java environment, hence their approach is only applicable to Java based applications and it is merely meant to distribute future applications in plan, or in other words, it is not an appropriate method to distribute an existing application. JMX delivers APIs for Java applications management. While ProActive is utilized by [18] to make the applications granular, hence they can be computed in parallel. Additionally, ProActive provides remotely accessible JMX connector in safe and asynchronous manner. Unfortunately, paper [18] does not brief how they managed to mass distribute/copy the necessary files to each client computers.

Furthermore, the application bundled with JMX and ProActive is then deployed over OSGi (Open Services Gate-way initiative) gateway infrastructures that provide management of services. Each gateway defines unique hardware specifications such as CPU type, type of operating system, or even the usage of resources. The ways to deploy these OSGi gateways are divided into two; we can deliver the same unit plan for all gateways or we give each gateway a different unit plan. Moreover, the work by [18] succeeded to reduce as much as 50% of service installation time com-pared to standard JMX-Java RMI (Remote method Invocation) synchronous connector.

A distributed computing platform utilizing vehicle charging stations was deployed in SUSTech campus [19]. It managed to reduce computational complexity by decomposing the scheduling problem into multiple sub-problems. Theoretically, this method is similar to BOINC's job scheduling, however the requirement of custom-made stations makes this solution expensive and difficult to obtain and deploy – compared to BOINC which is based on open-source software.

An advocacy of campus grid in African continent is led by Kenya that concludes various benefits of campus grid in collaborative research, such as shared instruments, peer sharing, and remote visualization [20]. However, the technical procedure or know-how has not been proposed. Therefore, our paper can fulfil this essential gap in implementation. On European continent, Finnish Grid and Cloud Infrastructure2 (FGCI2) project is comprised of fourteen (14) Finnish institutions that is assisting the local super-computers [21]. The provided services revolve around High Performance Computing and Cloud Computing, which cater for various natural scientific fields, engineering, and mathematics. Despite developing open-source software, FGCI2 does not disclose the procedure to establish their grid infrastructure, thus this gap signifies a common lacking of public information about grid development that justifies the contribution of this paper.

A part of campus grid design is job scheduling, where it deals with sequence of task allocations. A scheduling method that targets campus grid with inalienable resources has been developed, but its performance was only compared with simple First Come First Served (FCFS) method [22], hence doubting its general suitability for campus grid. Additionally, the overall grid establishment method is not disclosed, therefore the gap remains.

Related to job scheduling is another component of campus grid, which is workflow. It utilizes container-based schedulers in order to minimize performance loss, and hence gains higher resources efficiency. The proposed container schedulers architecture using Makeflow and Mesos has been able to achieve higher transfer throughput and resource utilization [23]. Similar to other reviewed campus grid works, the overall campus grid platform development procedure is not shared. Regardless of it, this work may be integrated to our proposed BOINC-based campus grid.

In spite of two (2) decades of campus grid history, no institution has publicly shared the automated efficient way of campus grid platform development. Thus, this paper will contribute to that area by detailing the necessary steps to automate files distributions and software configuration.

Proposed Client Files Distribution in Networked Campus Grid

Distributing the necessary files and installing the BOINC client in our campus grid environment was a tedious process since there were more than 40 labs to be installed, and every lab had 30 to 40 computers to be clients/compute nodes. On the other hand, the campus grid team consisted of only 25 student volunteers with the average attendance of 15 students for every installation session.

To accelerate the lab installation, therefore a centralized file distribution and software execution was needed. We utilized proprietary software named EMCO Remote Deployment to send and execute the necessary client files in a centralized manner. It is a Windows-based

software and it automates the software management including the distribution, installation, uninstallation, and update [16].

EMCO Remote Deployment will firstly scan the domain network to map the computers according to their do-main membership. Afterwards, specific domain or specific computers in a domain can be selected to receive the application files. And eventually EMCO Remote Deployment remotely executes the necessary files using its remote command prompt feature. In our case, it remotely executes the BOINC client software. It can distribute the files with-out the client computers be logged-in. The login is however needed once the files are to be executed using remote command prompt.

The drawback of using EMCO Remote Deployment is that it needs a considerable amount of time to scan all domains. For our case with a lab of 40 computers, it took more than 15 minutes to scan the domain. Later we found out that with a team of 15 students, it was faster to distribute the necessary files using traditional Windows-based file transfer via shared folder.

In a nutshell, with around 40 computers and 15 team members, it was more effective to distribute the application using traditional Windows shared folder than to have the team to manually browse and copy the files. However, for the case where the number of client computers much exceed the number of team members or no team member at all (single administrator), it is worth to consider using EMCO Remote Deployment software.

Proposed Campus Grid Client Software Auto Configuration

EMCO Remote Deployment has a limitation. It executes the client application only in one shot i.e. it only initiates the application but without the capability to proceed the installation process once the software's Graphical User Interface (GUI) prompts out. Thus, it is not possible to remotely click the on-waiting "Next" buttons of the BOINC client software and other subsequent on-screen configurations.

Furthermore, the BOINC client software needs to be configured after installation, for example it needs to connect to the BOINC server's hostname, login to the BOINC server, and attach itself to a project. These are additional tasks to be remotely auto-configured for each client computer.

To execute the above automated tasks, Windows scripting was the feasible solution. A freeware named AutoIt provides the capability to automate software installation and GUI configuration using BASIC-like script to simulate keystrokes, mouse movement, and window/control manipulation [17]. This AutoIt script can be easily converted to Windows executable file (.exe) using the conversion feature of AutoIt.

Two AutoIt scripts were created for our mission. The first one (boinc.exe) was to control the BOINC client software on-screen installation, and the second script (boincmgr.exe) was to configure the BOINC client software after its installation for it to connect to the BOINC server and attach to project. Below are the flow charts of the two AutoIt scripts.

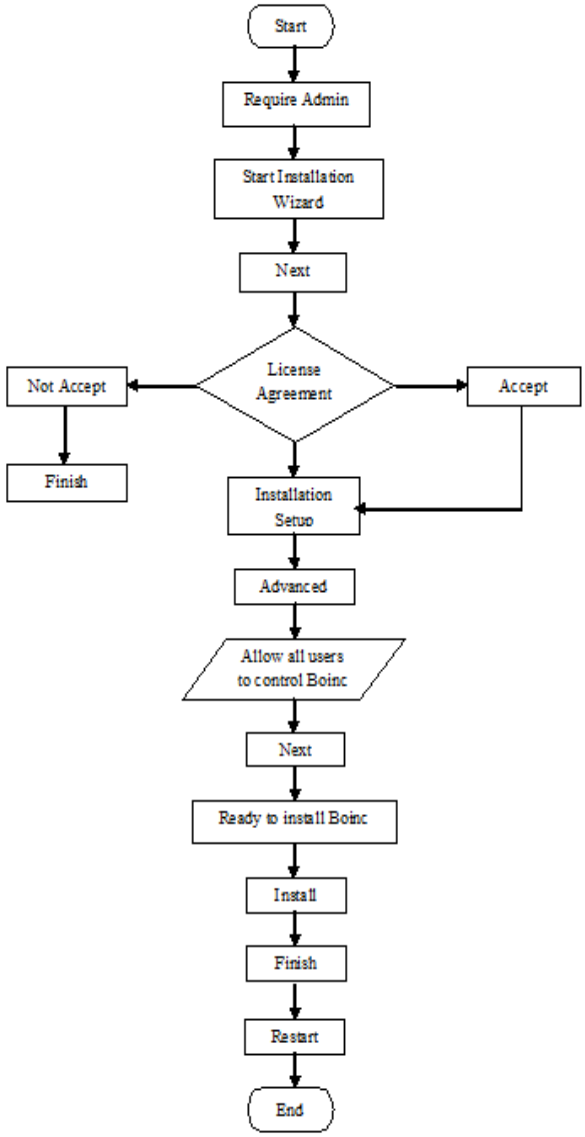

Fig. 1. BOINC Client Installation AutoIt Script Flow Chart

The procedure for the previous Figure 1 flow chart is explained as follows:

1. Require admin authorization to install Boinc Client.
2. Start the installation wizard.
3. Accept the license agreement to continue install software.

If (Accept)

Continue the installation process to step 4

Else

Terminate the installation process

4. Start the software installation setup.
5. Select advance to choose some setup options.
6. Tick the “Allow all users on this computer to control Boinc” box.
7. Continue installation process.
8. Click finish after installation complete.
9. Restart the PC after finishing installation.

Next is the flow chart for the AutoIt script to configure the BOINC client software after installation.

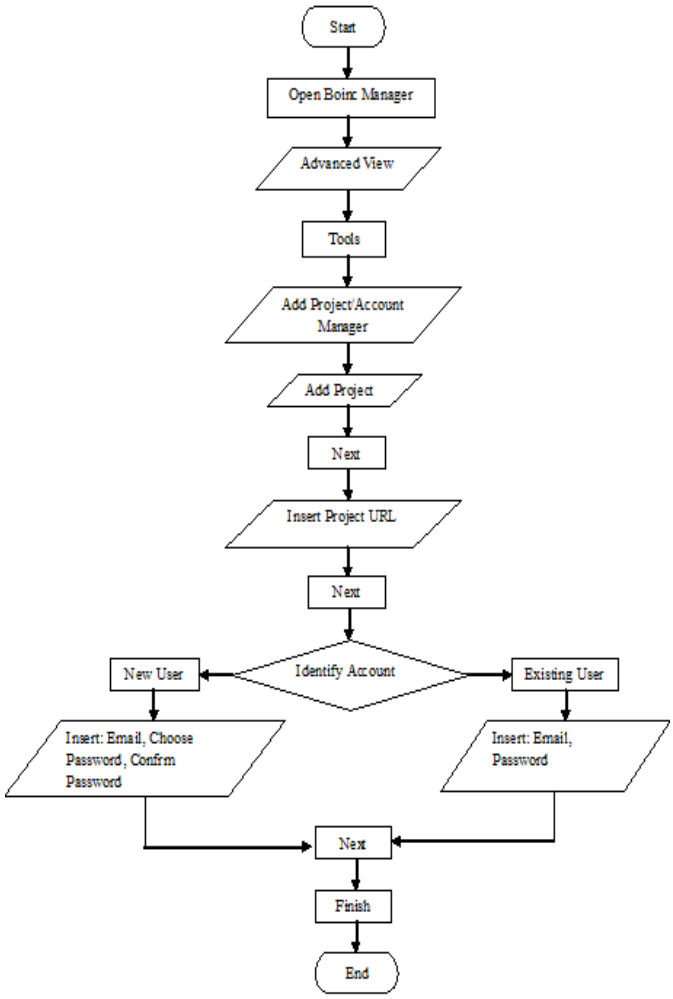

Fig. 2. Flow Chart of BOINC Client Configuration after Installation

The procedure for the above Figure 2 flow chart is described below:

1. Open the Boinc Client Manager.
2. Click at Advanced View
3. Select tools.
4. Choose Add Project/Account Manager.
5. Select Add Project.
6. Click next to continue attach the project.
7. Insert Project URL.
8. Click next to create account.
9. Identify user account,

   If (new user)

   Insert: Email Address, Choose Password and Confirm Password

   Else (Existing user)

   Insert: Email Address and Password

10. Click next to complete the project attachment.
11. Click finish after successful attaching the project.

The BOINC client was called by the boinc.exe script. However, it was found that some client computers have the installation process being blocked randomly In such cases, it was necessary to click manually the relevant button or even to fill the GUI forms.

After the BOINC client proper installation, next script to execute via EMCO Remote Deployment remote command prompt was the boincmgr.exe for calling the BOINC client manager program to be configured based on the content of the boincmgr.exe script. In this step, the process often stopped again somewhere, and the solution was the same as the installation process; either we manually click the relevant button or even fill the GUI forms by ourselves.

Another problem faced during the installation, was that although the "Allow all users on this computer to control BOINC" box was checked at the BOINC client installation phase, the BOINC client could only work on Administrator account and failed on Guest account. To escalate Guest account for it to run BOINC client, we execute the following command at command prompt under Administrator account:

net localgroup boinc_admins "guest" /add

Subsequently, the client computer was restarted for the command to take effect. It was initially attempted to execute the above command with EMCO Remote Deployment remote command prompt but it did not succeed, thus this function was executed manually on each computer to accomplish the task.

At last, the attachment of the BOINC client computers to the BOINC server was tested. It was done by submitting a job from the BOINC server, and the event log of the BOINC client computer was checked to verify if it was receiving the task. Additionally, any error message was checked for existence.

In brief, the proposed deployment with automation of campus grid involves the steps below:

1. Login to all client computers as Administrator
2. Distribute necessary application files (boinc.exe script, boincmgr.exe script, and BOINC installer) using Windows shared folder or EMCO Remote Deployment
3. Execute boinc.exe using EMCO Remote Deployment remote command prompt feature
4. Check on each computer if the BOINC client installation proceeds smoothly or be stuck somewhere.
5. Restart the installed client computers
6. Login back as Administrator
7. Execute boincmgr.exe using EMCO Remote Deployment remote command prompt feature
8. Check on each computer if it is correctly attached to the BOINC server.
9. On each client computer, open command prompt then type: net localgroup boinc_admins “guest” /add
10. Restart the computer to take effect of the command and then login as Guest account
11. Login to the BOINC server and submit multiple jobs
12. Check on each BOINC client computers if they properly accept tasks

2. Conclusion

The effort and time invested to deploy campus grid with automation are worthwhile as the grid is highly supporting students/researchers and lecturers to execute simulation programs that require intensive computation and data storage.

Improvements on the mass BOINC client deployment are aspiring us, especially to automate the remote login to each client computer, to speed up overall installation process.

The overall proposed method to deploy campus grid infrastructure copes with the distribution, execution, and remote management. And it is flexible because it applies easily to existing application without any necessity to alter the application code, thus the programming language of the application can be preserved.

References

1. Wang, J., Yang, Z., and Kou, W. (2007). A Solution for Building Campus Grid. First IEEE

International Symposium on Information Technologies and Applications in Education. ISITAE '07 [Paper]. pp. 545-548.
2. Al-iesawi, A. M. and Samat, M. I. M. (2010). A case study on Implementati on of Grid Computing to Academic Institution. 2010 International Symposium in Information Technology (ITSim) [Paper]. 3, pp. 1525-1530.
3. Foster, I. and Kesselman, C. (1999). The Grid: Blueprint for a New Computing Infrastructure. San Fransisco: Morgan Kauffman.
4. Chervenak, A., Foster, I., Kesselman, C., Salisbury, C., and Tuecke, S. (2009). The Data Grid : Towards An Architecture For The Distributed Management And Analysis Of Large Scientific Datasets. Journal of Network and Computer Applications [Paper], 23(3), pp. 187-200.
5. Ninth Malaysian Plan. http://www.parlimen.gov.my/news/eng-ucapan_rmk9.pdf
6. MYREN High Speed Network. http://www.myren.net.my/sites/default/files/story/2010/07/HamdanIsmail.pdf
7. Por, L. Y., Su, M. T., Ling, T. C., Liew, C. S., Ang, T. F., and Phang, K. K. (2006). Issues of Establishing a Campus-wide Computational Grid Infrastructure in the GERANIUM Project. Proceedings of the International Conference on Networking, International Conference on Systems and International Conference on Mobile Communications and Learning Technologies (ICNICONSMCL'06) [Paper].
8. BOINC Intro. (2018). http://boinc.berkeley.edu/trac/wiki/BoincIntro.
9. BOINC Basic Concepts. (2022). http://boinc.berkeley.edu/trac/wiki/BasicConcepts.
10. BOINC Wrappers. (2022). http://boinc.berkeley.edu/trac/wiki/WrapperApp.
11. BOINC Job Processing. (2022). http://boinc.berkeley.edu/trac/wiki/JobIntro.
12. GenWrapper. (2023). http://genwrapper.sourceforge.net/.
13. Marosi, A. C., Balaton, Z., and Kacsuk, P. (2009). GenWrapper: A Generic Wrapper for Running Legacy Applications on Desktop Grids. 3rd Workshop on Desktop Grids and Volunteer Computing Systems (PCGrid 2009) [Paper].
14. BOINC API. (2022). http://boinc.berkeley.edu/trac/wiki/BasicApi.
15. The DC-API. (2022). http://www.desktopgrid.hu/index.php?page=34
16. EMCO. (2023). http://emcosoftware.com/remote-deployment.
17. AutoIt Introduction. (2022). http://www.autoitscript.com/autoit3/docs/introduction.htm.
18. Baude, F., Contes, V. L., and Lestideau, V. (2007). Large-scale Service Deployment - Application to OSGi. Third International Conference on Autonomic and Autonomous Systems (ICAS 2007) [Paper]. pp. 19-24.
19. Shang, Y., Liu, M., Shao, Z., Jian, L. (2020). A centralized vehicle-to-grid scheme with distributed computing capacity engaging internet of smart charging points: Case study. International Journal of Energy Research, Special Issue: Smart Energy Technologies, vol. 45, issue 1, pp. 841-863.
20. Murumba, J. and Micheni, E. (2017). Grid Computing For Collaborative Research Systems In Kenyan Universities. The International Journal of Engineering and Science (IJES), vol. 6, issue 4, pp. 24-31.
21. Shillanpaa, A. and Salmela, V. (2019). FGI - Finnish Grid Infrastructure. https://wiki.eduuni.fi/display/cscfgi/FGI+-+Finnish+Grid+Infrastructure.
22. Uzdenov, T. A. (2022). A new approach for dispatching task flows in GRID systems with inalienable resources. Journal of Edge Computing, vol. 1, issue 1, pp. 68-80.
23. Zheng, C., Tovar, B., and Thain, D. (2017). Deploying High Throughput Scientific Workflows on Container Schedulers with Makeflow and Mesos. 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid 2017), pp. 130-139, doi: 10.1109/CCGRID.2017.9.