International Institute for
Applied Systems Analysis
Schlossplatz 1
A-2361 Laxenburg, Austria

Tel: +43 2236 807 342
Fax: +43 2236 71313
Email: publications@iiasa.ac.at
Web: www.iiasa.ac.at

**Interim Report IR-09-052**

# A Virtualized SGE-based Computational Cluster for Heterogeneous Environments

Muhammad Asif (asifsamiarain@gmail.com)

**Approved by**

Marek Makowski (marek@iiasa.ac.at)

Leader, Integrated Modeling Environment Project

October, 2009

## Foreword

This report describes the research carried out by the author during participation in the Young Scientists Summer Program 2009 (YSSP) with the Integrated Modeling Environment Project. The main motivation for the research described in this paper was to explore how the distributed and parallel computing technologies can be implemented at IIASA for a better utilization of the available computing hardware. Given the large amount of idle time of many PCs, the focus was on the PC-desktop hardware. Main reservations for sharing the desktop computing resource include security (of the data stored on the local disks), reliability (relatively frequent needs for rebooting), performance of resource-sharing applications, and incompatibilities of different versions of MS-Windows. To address these issues, the author has created a Linux-based environment for virtual machines sharing one PC. Such virtual machines were running different operating systems (different versions of Linux and MS-Windows). Each of the machine had a well isolated share of the common disk-space, thus the privacy of the data was assured. The author also implemented this approach on a cluster of PC. The common resources (especially the priorities of tasks run by the virtual machines) were managed by the Sun Grid Engine (batch job scheduler and manager for cluster/grid). Moreover, the author also explored possibilities of parallel computing within the developed distributed computing environment.

The developed virtualized SGE-based cluster for heterogeneous environments has been successfully tested using various compiled applications, and java applications. In particular the data and process security of physical and virtual machines was proven. All basic functionality of individual jobs (such as reading and writing data, including redirection of standard input and output) was achieved. In other words, it was shown that the virtualized SGE-based computational cluster is able to execute heterogeneous jobs in heterogeneous environments. The report contains a number of useful hints for deployment of a PC-based heterogeneous computational cluster.

The reported, three-month research provides a good basis for further developments. It shows a road to effective and safe utilization of huge amount of idle time of many PC. However, to achieve this goal more work is needed, especially for development of user-friendly environment for creation of jobs and interactions with virtual machines by users without knowledge of operating systems. Also more work is needed for embedding more job execution types, such as shells, python, Matlab, etc, as well as other parallel programming models.

## Abstract

The computing and modeling environment of IIASA was studied in the context of computation-intensive and resource-demanding applications/models which are being developed and used by the researchers/scientists of IIASA. High Performance Computing applications can be classified into two broad computing fields; sequential distributed and parallel distributed applications and these applications has been developed for heterogeneous operating system architectures such as Linux, Windows and Solaris etc. Majority of IIASA applications/models belong to the latter class of computing and these applications are resource demanding when the extensive and repetitive use of these applications is required according to the need of some research study. Not every sequential application can be easily parallelized; therefore, instead of re-programming sequential applications into parallel ones, the idea of distributing such applications on computing cluster/grid is often an effective approach for accelerating the work. In the light of available computing resources and modest modeling environment of IIASA, the virtualization and Sun Grid Engine (batch job scheduler and manager for cluster/grid) was efficiently exploited and designed, built and tested. This resulted in a computational cluster supporting multiple operating systems and multiple sequential distributed and parallel distributed applications/models along with multiple job execution types such as binaries and JAVA.

**Keywords:** High Performance Computing (HPC), Resource Demanding Applications, Sequential and Parallel Distributed Applications, Virtualization, Sun Grid Engine (SGE)

## Acknowledgements

This report and the all the research described here were done during my participation in the Young Scientist Summer Program (YSSP) 2009 at the International Institute for Applied System Analysis in Laxenburg, Austria.

I would like to thank Mr. Bartosz Kozlowski for his supervision and a lot of good advice and practical assistance on my research he has given to me during the program. I also give many thanks to Dr. Marek Makowski, Mr. Jacek Wojciechowski and Dr. Hongtao Ren for their advices and cooperation. Moreover, I am grateful to Ms. Varvara Fazalova for sharing her computational code for testing purpose.

Finally, I would like to thank the ICT department of IIASA for all technical support, and all other departments of IIASA that provided financial and administrative support and thus made my participation in the YSSP possible.

## About the Author

Muhammad Asif is currently working at Global Change Impact Studies Center (GCISC) Islamabad, Pakistan as Scientific Officer. His research interests include use of tools and techniques and design algorithms for the deployment of High Performance Computing Cluster/Grid for computation intensive applications. Muhammad's recent research objective is to study the computing and modeling environment of IIASA and design, build and test a virtualized SGE-based computational cluster for heterogeneous environments to support variety of sequential and parallel distributed applications.

## Table of Contents

## List of Tables and Figures

# A Virtualized SGE-based Computational Cluster for Heterogeneous Environments

**Muhammad Asif** (asifsamiarain@gmail.com) * **

## Background and Motivation

Among the community of scientists/researchers who are actively involved in simulations and modeling, the computing terminologies like high performance computing (HPC) and resource intensive/demanding computing are popular and have immense importance. HPC is a rapidly growing field of computing since the last few decades, many research organizations and companies like IBM, Intel, HP, SUN, Microsoft, ANL (Argonne National Laboratory) are engaged to develop specialized hardware and software to achieve the optimum computing power. Meanwhile, small and medium size companies and research organizations have the requirements to enhance their computing power for the accomplishment of their work and research studies in optimum time with the help of HPC infrastructure.

The idea of studying and designing a feasible computational cluster (HPC environment) for the modest modeling environment of IIASA and particularly the research question how to deploy and manage the cluster of workstations comprised of heterogonous operating environments for (sequential and parallel distributed) resource demanding programs/models (loosely coupled and tightly coupled applications all together) by using virtual machines and Sun Grid Engine (SGE) motivated me.

There are generally two types of computing; sequential and parallel; however, for the IIASA modest computing environment we consider the following categories of high performance computing:

- Sequential, Not Distributed

  This class of computing deals with the simple and traditional computing environment, in which most users are familiar with at least with one computer language that meets their computing needs. The programs coded for such environments utilize only one available processor and the user actually does not distribute the computational task among several processors.

*Integrated Modeling Environment Project, IIASA

** COMSATS Institute of Information & Technology (CIIT)

The question here is how the traditional sequential computing could become a class of high performance computing. For example, if the user is interested to execute the sequential program many times (e.g., for many different sets of parameters), then even a rather simple sequential program becomes computation-intensive, and requires a substantial amount of computing resources.

In the modest modeling environment of IIASA, approximately 65-70% programs/models belong to this class of computing.

- Sequential, Distributed

  In this class, all sequential programs/models addressed in previous class are distributed among several processors by using a cluster of workstations along with some batch job management system like Sun Grid Engine (SGE), Portable Batch System (PBS) etc.

  Currently, 15-20% IIASA programs/models use such type of computational environment for the solution of their computing needs.

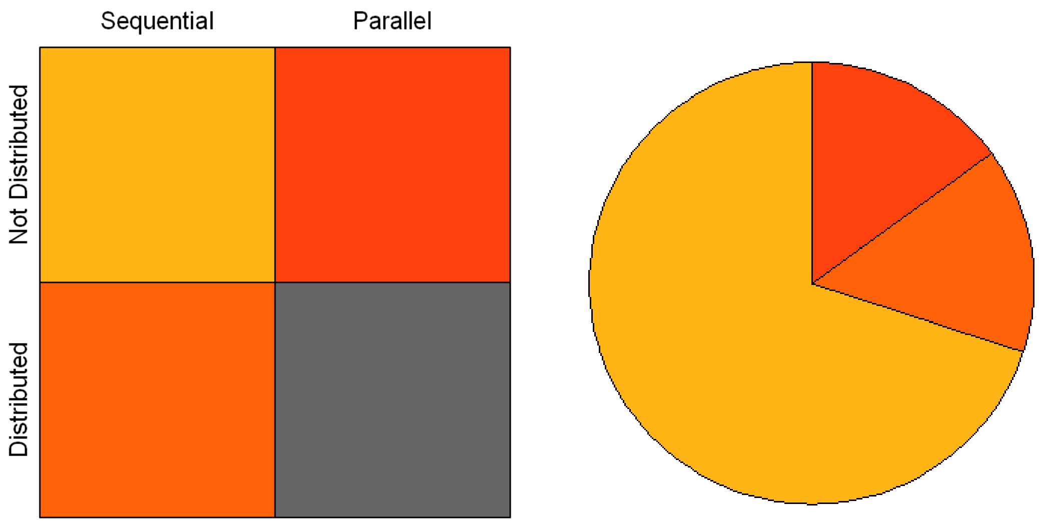


**Figure 1: High Performance Computing Classifications**

Based on: Bartosz Kozlowski. *Crash Course in Resource-Demanding Computing.* PJIIT Lectures 2005

- Parallel, Not Distributed

  In the context of real resource demanding programs/models, the high performance computing is always considered as parallel computing. Particularly, this class deals with parallel computing without using the cluster of workstations (in other words computations are “not distributed” amongst workstations). In fact such parallel programs will be using the power of all available processors atone computer; considering the rapid advancements in processor architecture; multiple processors are now available within a single computer in the form of Dual Core, Quad Core etc.

Such sequential programs can be ported into parallel programs by using parallel versions of compilers for commonly used programming languages, e.g., C, C++ and FORTRAN, supported by a parallel programming model like Parallel Virtual Machine (PVM) or Message Passing Interface (MPI). But the user must keep in mind that such type of porting would require a substantial amount of time to first learn the parallel programming tools and techniques, and then to reprogram (at least parts of) the applications.

In light of IIASA's modeling environment, about 10% programs/models belongs to this class of computing.

- Parallel, Distributed

  This class actually deals with real high performance parallel computing clusters being used worldwide to solve the highly resource demanding problems in optimum time like Global Climate Models (GCM) and Regional Climate Models (RCM) etc.

  To the best of author's knowledge, IIASA researchers have not dealt with any computation intensive model that belong to this class of computing, and consequentially there is currently no need for high performance parallel computing clusters.

Therefore, in the context of high performance computing classes defined above, for the modest modeling environment of IIASA it was rational to explore a possibility of implementation of a computational cluster for heterogeneous applications and operating environments. Considering the costs of deployment and maintenance of such environments, the author focused on the virtualization-based approaches.

## 1. Research Questions

Taking into account the above summarized arguments, the author decided to setup and test a cluster of virtual machines by using a few PCs, the virtual machine creation tool VirtualBox, and the powerful batch job scheduling and execution system Sun Grid Engine (SGE). In order to implement this approach, the following research questions were addressed during the YSSP period.

- Exploitation of Virtualization
- Exploitation of Sun Grid Engine (SGE)
  - Installation/Configuration of Master Host (always on Linux/UNIX)
  - Installation/Configuration of Execution Host

        - Linux/UNIX
        - WinXP
- Exploitation of suitable Parallel Programming Model
    - Embed MPICH2 for Linux/UNIX into SGE
    - Embed MPICH2 for WinXP into SGE
- Design/Build/Test the computational cluster of virtual machines

## 2. Experimental Design and Methodology

The first step in deploying the planned environment was to setup the computing hardware infrastructure, which was the basis for the deployment of cluster of workstations/virtual machines and exploitation of the Sun Grid Engine. Therefore, in order to fulfill the computing hardware needs, the author explored the virtualization, and worked with the virtual machine creation tool VirtualBox (details are provided in Appendix I). With the help of this tool, multiple virtual machines can be created on a physical computer, and each virtual machine can have either the same or a different operating system. The virtual machine actually mimics a real machine, and as far as the operating system installation, utilization and behavior is concerned, such a set-up is in fact similar to a single operating system running on a single computer. In addition, it could be said that the virtual machine is a mimic of a real machine, but its operating system is in fact a real one.

Therefore, with the help of two physical machines (the specification of the machines are shown in Table 1) eight virtual machines were created (the specification of the virtual machines are shown in Table 2), and both computers were connected via a 100 Mbps Ethernet switch. In this way, a small network of virtual machines was created by using a private IP addressing scheme; the network consisted of five Linux virtual machines and three Windows virtual machines as shown in Figure 1.

Table 1: Specifications of Real Computers

| | **Computer Name** | **Processor** | **Processor Speed** | **Hard Disk Space (GB)** | **RAM Space (GB)** | **Operating System** |
|---|---|---|---|---|---|---|
| S1 | pc98192 | Intel(R) Core(TM)2 Quad Q9650 | 3.00 GHz | 110 | 4.00 | Windows Vista |
| S2 | Pc98131 | Intel(R) Core(TM)2 Quad Q9650 | 3.00 GHz | 300 | 4.00 | Windows Vista |

| Table 2: Specifications of Virtual Machines | | | | | |
|---|---|---|---|---|---|
| | | | | | |
| | | **Virtual Machine Name** | **Hard Disk Space (GB)** | **RAM Space (MB)** | **Virtual Memory Space (MB)** |
| S1 | 1 | vm-ubuntu1 | 8 | 512 | 12 |
| S1 | 2 | vm-ubuntu2 | 8 | 512 | 12 |
| S1 | 3 | vm-ubuntu3 | 8 | 512 | 12 |
| S1 | 4 | vm-winxp1 | 8 | 512 | 12 |
| S2 | 5 | vm-winxp2 | 8 | 512 | 12 |
| S2 | 6 | vm-winxp3 | 8 | 512 | 12 |
| S2 | 7 | vm-ubuntu4 | 8 | 512 | 12 |
| S2 | 8 | vm-ubuntu5 | 8 | 512 | 12 |

32-Bit Ubuntu Linux and WinXP were the operating systems used for the cluster. After successful installation of operating systems on virtual machines, one needs to configure the network and the machines to behave like a cluster of virtual machines. Prior to the deployment of Sun Grid Engine, the required configurations have been addressed in Appendix II in detail. Moreover, the details of the deployment of Sun Grid Engine are summarized in Appendix III.

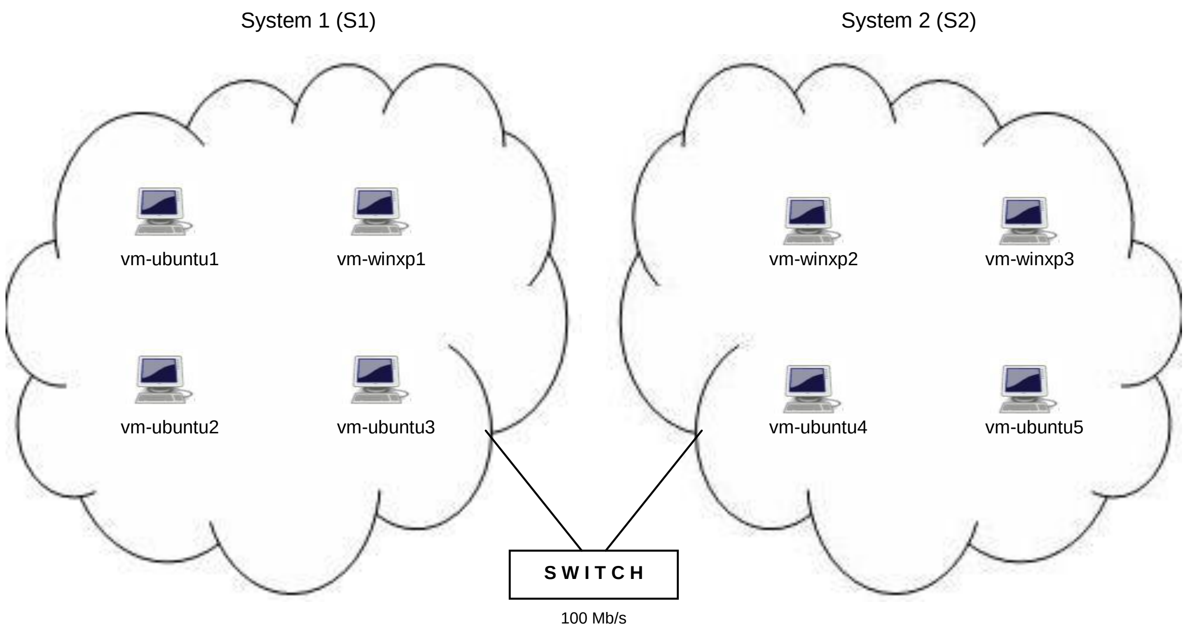


**Figure 2: Virtualized Computational Cluster**

According to the Sun Grid Engine specification and design, there must be one host acting as master host, and other hosts act as execution hosts. In fact, the master host is the brain of Sun Grid Engine running the scheduler, keeping the track of all the computing resources configured, and keeping the accounting and reporting information about the executed jobs. The master host can act as execution host as well but this is not necessary (please refer the Appendix III for the solution of technical issues related to the deployment of SGE).

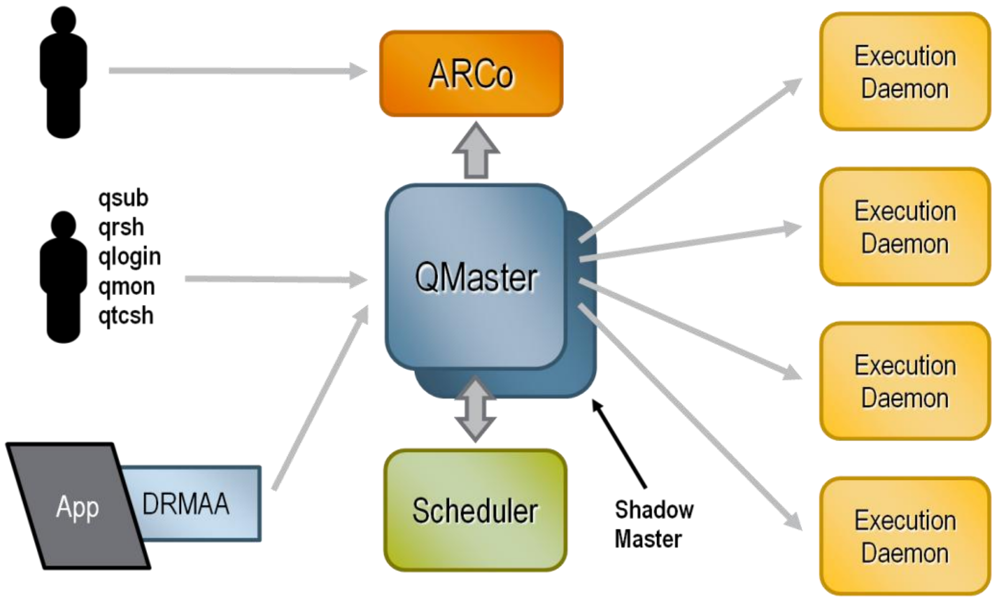


**Figure 3: Sun Grid Engine Components**

Based on: Sun Grid Engine Guide

In the light of our experimental setup the master host of Sun Grid Engine is the only host which is acting as submit host and configured at hostname “vm-ubuntu1” and keep in mind that the master of SGE is always configured on Linux/UNIX operating system. The submit host of the Sun Grid Engine has great importance and this is actually the entry point from where the computational job is going to submit by user to master host of SGE then on the bases of idle resources among the available resources, the master host dispatch the job to execution hosts. After successful completion of job the results gathered back at “vm-ubuntu1”. In fact the “vm-ubuntu1” is the host which is acting as brain for our computing infrastructure. In order to keep the network traffic load minimized, for the explicit communication among hosts other than the SGE communication mechanism, passwordless ssh among all the hosts was preferred rather NFS configurations.

## 3. Results and Discussions

By using the selected computing resources composed of two physical PCs, multiple virtual machines were installed with the help of VirtualBox (a virtual machine creation tool) and Sun Grid Engine (for batch job scheduling and execution) designed by Sun Microsystems; in this way  a virtualized SGE-based computational cluster was designed as a heterogeneous environment supporting multiple types of applications/models.

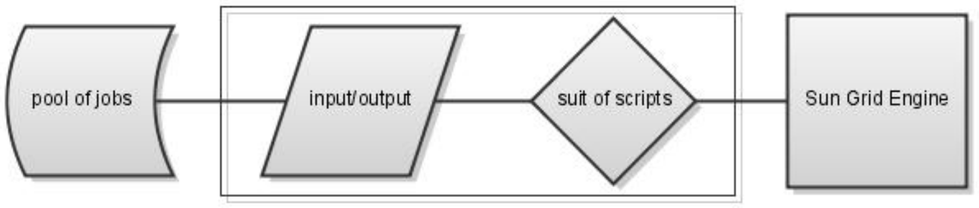


**Figure 4: Main flow diagram of job execution process**

Figure 4 summarizes the job execution process. As the master host is the only host that interacts with the users, and also acts as the SGE submit host. Therefore a pool of jobs has been created at master host, where the user can submit the computational job pack. The computational job pack is composed of executable(s), input data sets and the following necessary information:

- MPI
- Multithreading
- Number of Processors
- Architecture(s)
- Command Line Arguments
- Data Files
- Repetitions (ITERS)
- Parameterization etc

While processing the job pack at master host, two important elements are considered: first, the job type (either it is a sequential distributed job or a parallel distributed job); second, the suitable operating system (a specific version of Windows, or Linux, etc). Moreover, the SGE has its own efficient mechanism of queuing, where number of machines (execution hosts) can be gathered with respect to hardware architecture, software architecture or by using some other queue configuration criteria according to the needs of applications. On the bases of information provided by the user in job pack and availability of computing resources at particular appropriate queue type, the computational job are submitted to cluster as shown in Figure 5.

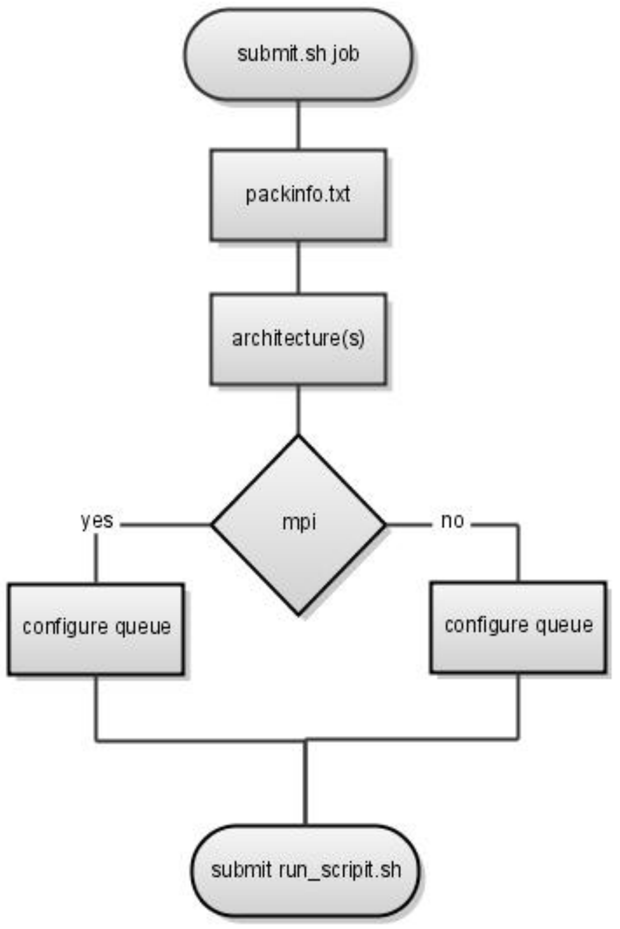


**Figure 5: Flow diagram of job execution process at master host**

Once a computational job has been submitted, the SGE will schedule and dispatch the job and assign any available, appropriate and idle execution host(s) according to the needs of the job. Figure 6 illustrates the job execution process at execution host, where the execution host fetches and reads the job pack from the pool of the jobs via password-less SSH mechanism. The cluster is able to support multiple types of job executables like binaries, JAVA and shell etc but at this stage, we have implemented and tested two types of job executables such as binaries and JAVA. Binaries have further two types of job executables like MPI or NOT MPI. In short, after carefully identifying the job executable type, an appropriate job execution environment will be configured, and finally the job will be executed. After successful execution of the job, the execution host will submit the results back at the pool of jobs from where the job was initially fetched, and clean the utilized local disk space to make it available for future jobs.

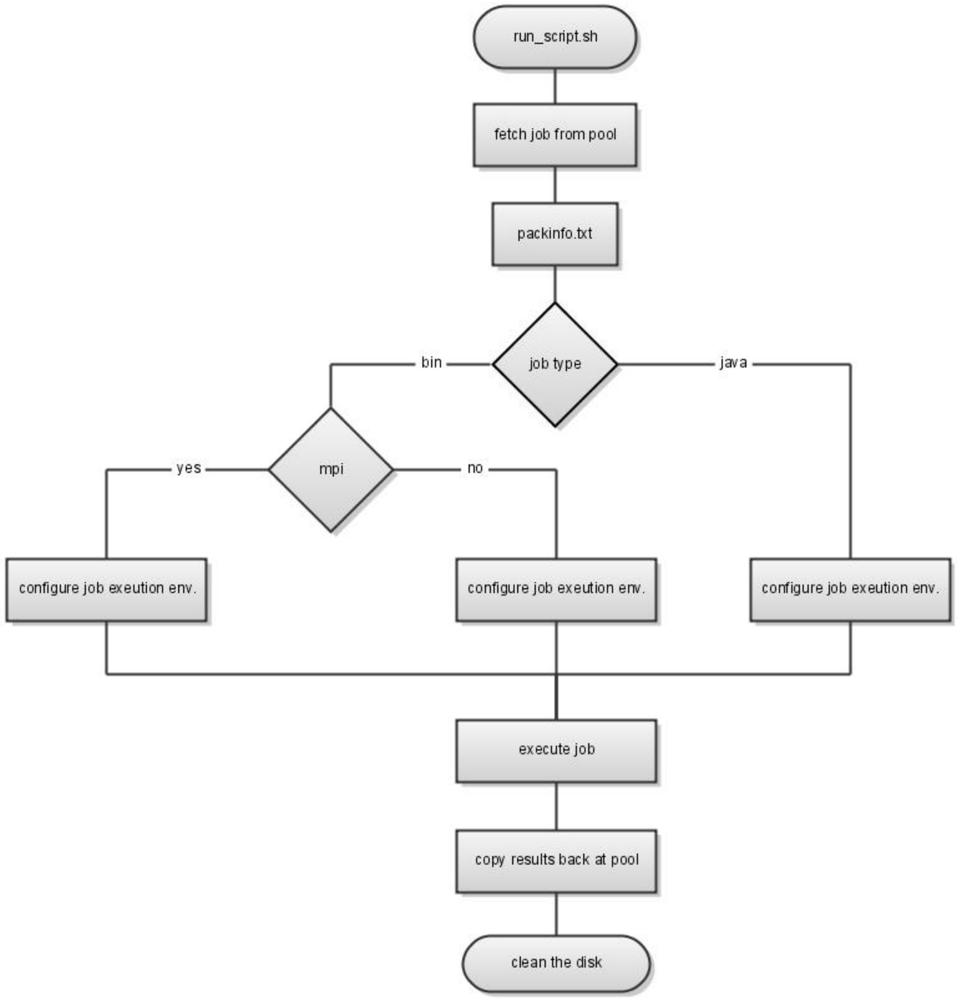

run_script.sh
fetch job from pool
packinfo.txt
job type
bin
java
mpi
yes
no
configure job exeution env.
configure job exeution env.
configure job exeution env.
execute job
copy results back at pool
clean the disk

**Figure 6: Flow diagram of job execution process at execution host**

## 4. Summary and Future Work

In the context of exploring the High Performance Computing environments suitable for the computing needs of IIASA, a virtualized SGE-based cluster for heterogeneous environments has been designed, built and tested successfully. The doubts and fears related to the data and process security of physical machines by the use of virtual machines were found to be unjustified, i.e., each job is executed in a dedicated space, and has no access to other parts of the file system; if the job causes a crash of the operating system, then it crashes only the virtual machine on which the job runs, and this does not affect other virtual machines. In addition, the Sun Grid Engine was deployed with multiple types of operating systems and tested its job execution mechanism with simple applications and on the bases of its batch job scheduling and execution mechanism multiple preliminary conducted such as reading and writing into a file, reading and writing from standard input and output and writing into the standard error etc. Basic purpose of these tests was to be assured about the job execution mechanism and related basic operations which are always required by any type of computational job. Moreover, to execute the MPI and JAVA jobs, the respective tools and environments were configured and tested with simple MPI and JAVA applications successfully. Therefore the virtualized SGE-based computational cluster is able to execute heterogeneous jobs for heterogeneous environments.

In future, there will be requirement to embed more job execution types such shell, python, Matlab, CPLEX etc and there will be requirement to introduce other parallel programming models as well like PVM etc. From the user perspective, an efficient graphical user interface (GUI) is highly desired to a user-friendly support of users in specification and submission of computational jobs, as well as monitoring the job and finding the computation results. Although the Sun Grid Engine provides the basic scheduling and accounting information mechanism regarding the computing resources and executed jobs through its scheduler and accountant, it is recommended to design and implement a hyper-scheduler and accountant dedicated to IIASA computing environment.

## References

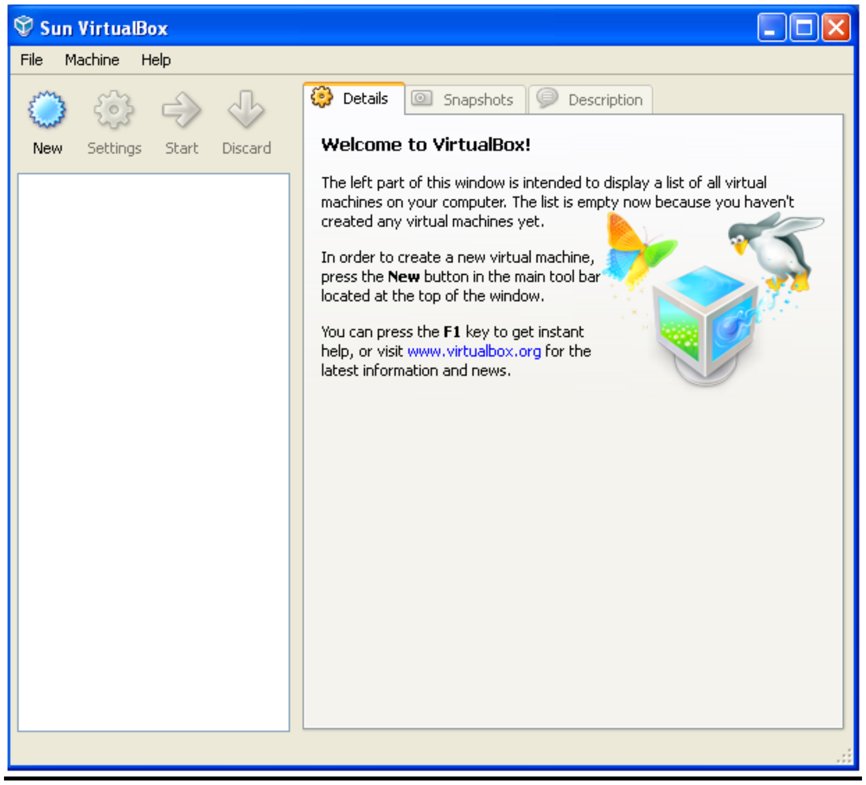
1. Virtualization tool: Information http://www.virtualbox.org/
2. Sun Grid Engine packages: Information http://www.sun.com/software/sge/
3. http://gridengine.sunsource.net/
4. http://wikis.sun.com
5. http://docs.sun.com/app/docs/coll/1017.3
6. http://biowiki.org/
7. http://shef.ac.uk/wrgrid/
8. Introduction to Microsoft Windows Services for UNIX 3..5: http://technet.microsoft.com/en-us/library/bb463212.aspx
9. http://shum.huji.ac.il/~agay/sge/blog.cgi?notes
10. http://www.interopsystems.com/downloads/Configuring_OpenSSH.pdf
11. http://www.ks.uiuc.edu/Research/namd/wiki/index.cgi?NamdOnGridEngin e
12. http://hifi.metalabs.org/docs/sge.pdf

## Appendix I

### VirtualBox

VirtualBox is a freely available (http://www.virtualbox.org/) powerful tool that allows the users to create virtual machines by using a physical machine for testing a variety of applications. The users can build and test a computational cluster for sequential and parallel distributed applications by using VirtualBox; it supports multiple virtual machines with multiple operating systems running on a single physical computer.

VirtualBox supports the Microsoft Windows (Windows 2000, Windows 2003, WinXP and Windows Vista), Linux, Solaris, BSD, IBM/OS2 etc operating systems. For the reported research, VirtualBox version VirtualBox-2.2.4-47978-Win was used for creating eight virtual machines on two PCs running Windows Vista. The following snapshots are just introducing the way for creating a virtual machine; note that the use of operating system installed on each virtual machine is exactly the same as for the stand-alone installation of the operating system.

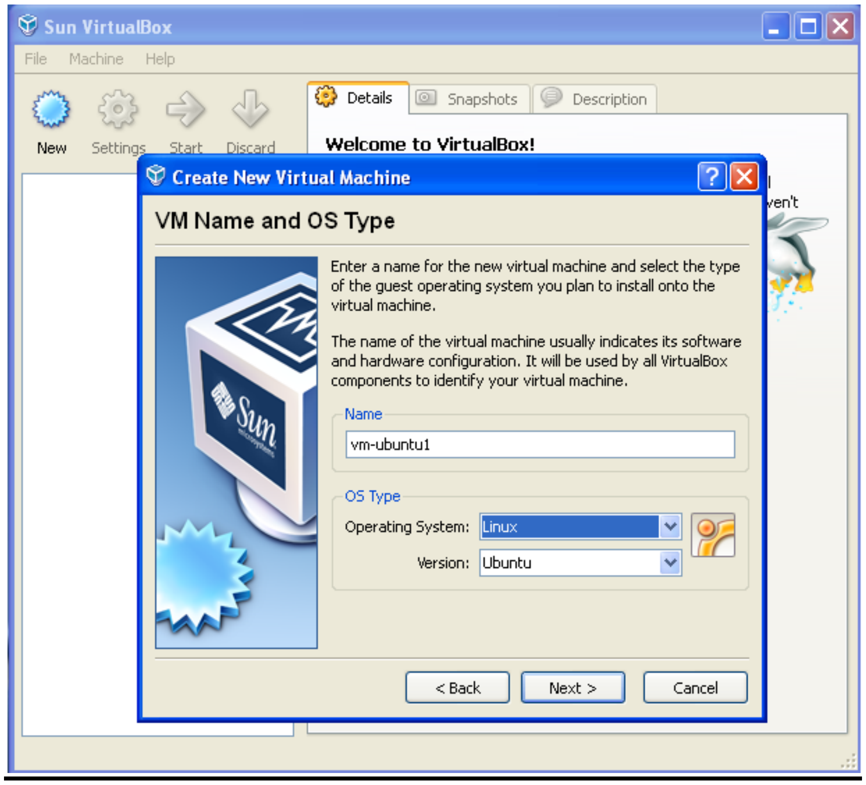


Welcome to VirtualBox!

The left part of this window is intended to display a list of all virtual machines on your computer. The list is empty now because you haven't created any virtual machines yet.

In order to create a new virtual machine, press the **New** button in the main tool bar located at the top of the window.

You can press the **F1** key to get instant help, or visit www.virtualbox.org for the latest information and news.



VM Name and OS Type

Enter a name for the new virtual machine and select the type of the guest operating system you plan to install onto the virtual machine.

The name of the virtual machine usually indicates its software and hardware configuration. It will be used by all VirtualBox components to identify your virtual machine.

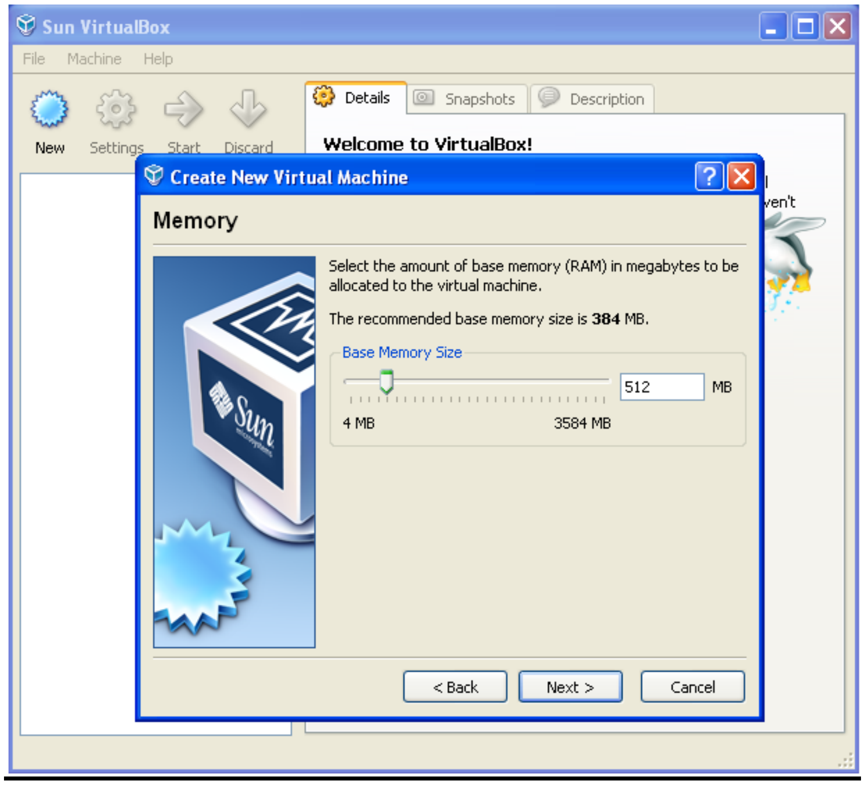
Name

vm-ubuntu1

OS Type

Operating System: Linux

Version: Ubuntu



Memory

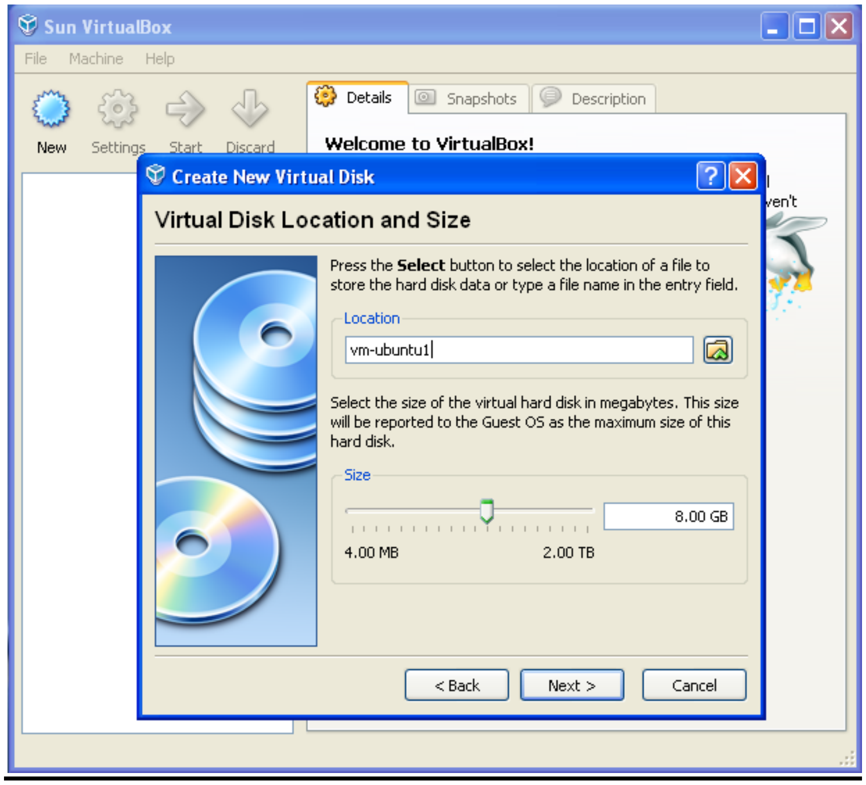
Select the amount of base memory (RAM) in megabytes to be allocated to the virtual machine.

The recommended base memory size is **384** MB.

Base Memory Size

512 MB

4 MB 3584 MB


Create New Virtual Disk

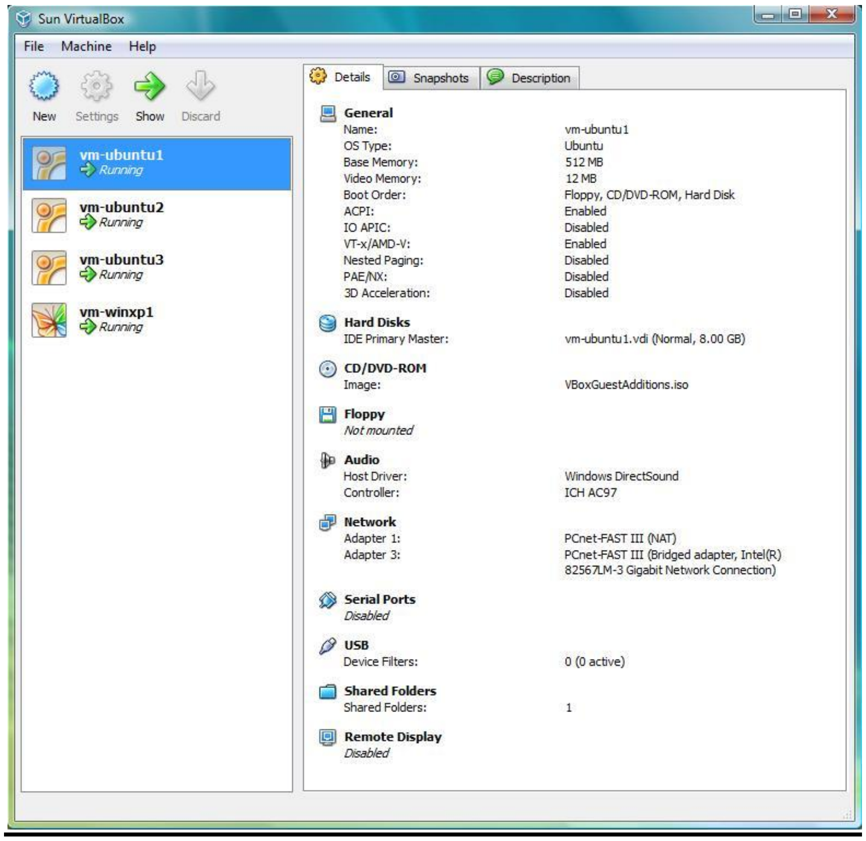
Virtual Disk Location and Size

Press the **Select** button to select the location of a file to store the hard disk data or type a file name in the entry field.

Location: vm-ubuntu1

Select the size of the virtual hard disk in megabytes. This size will be reported to the Guest OS as the maximum size of this hard disk.

Size: 8.00 GB (4.00 MB – 2.00 TB)



vm-ubuntu1 *Running*
vm-ubuntu2 *Running*
vm-ubuntu3 *Running*
vm-winxp1 *Running*

**General**

| | |
|---|---|
| Name: | vm-ubuntu1 |
| OS Type: | Ubuntu |
| Base Memory: | 512 MB |
| Video Memory: | 12 MB |
| Boot Order: | Floppy, CD/DVD-ROM, Hard Disk |
| ACPI: | Enabled |
| IO APIC: | Disabled |
| VT-x/AMD-V: | Enabled |
| Nested Paging: | Disabled |
| PAE/NX: | Disabled |
| 3D Acceleration: | Disabled |

**Hard Disks**

| | |
|---|---|
| IDE Primary Master: | vm-ubuntu1.vdi (Normal, 8.00 GB) |

**CD/DVD-ROM**

| | |
|---|---|
| Image: | VBoxGuestAdditions.iso |

**Floppy**
*Not mounted*

**Audio**

| | |
|---|---|
| Host Driver: | Windows DirectSound |
| Controller: | ICH AC97 |

**Network**

| | |
|---|---|
| Adapter 1: | PCnet-FAST III (NAT) |
| Adapter 3: | PCnet-FAST III (Bridged adapter, Intel(R) 82567LM-3 Gigabit Network Connection) |

**Serial Ports**
*Disabled*

**USB**

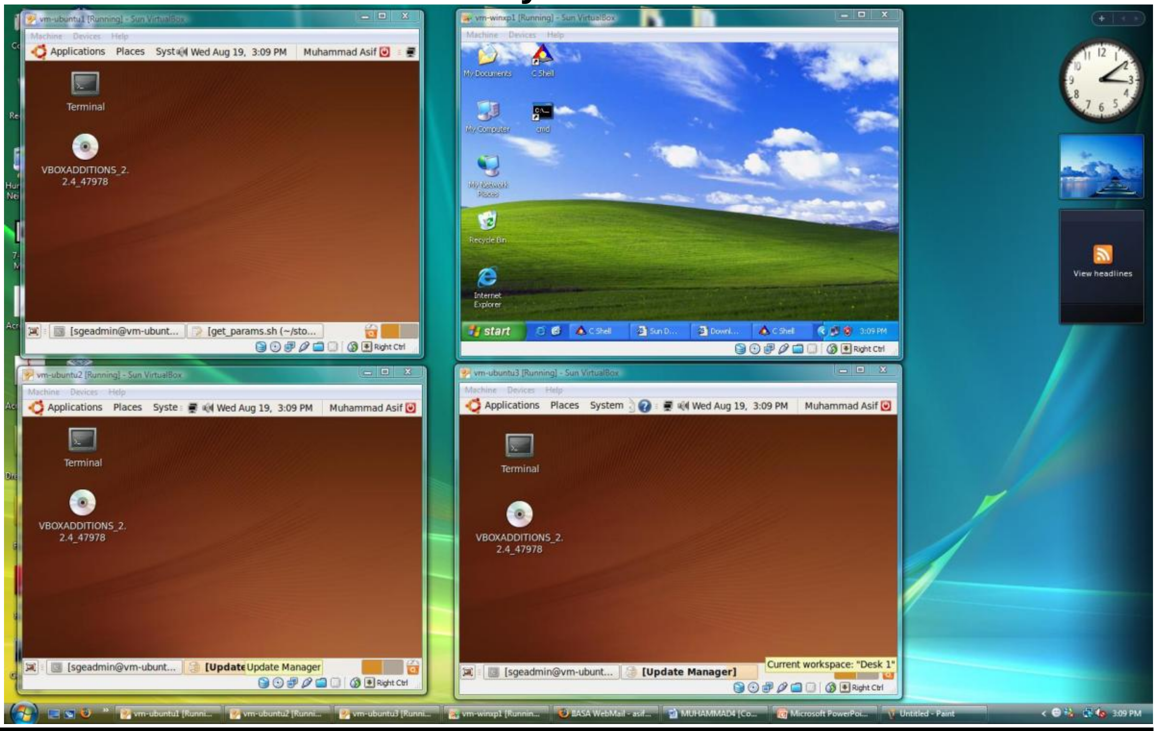
| | |
|---|---|
| Device Filters: | 0 (0 active) |

**Shared Folders**

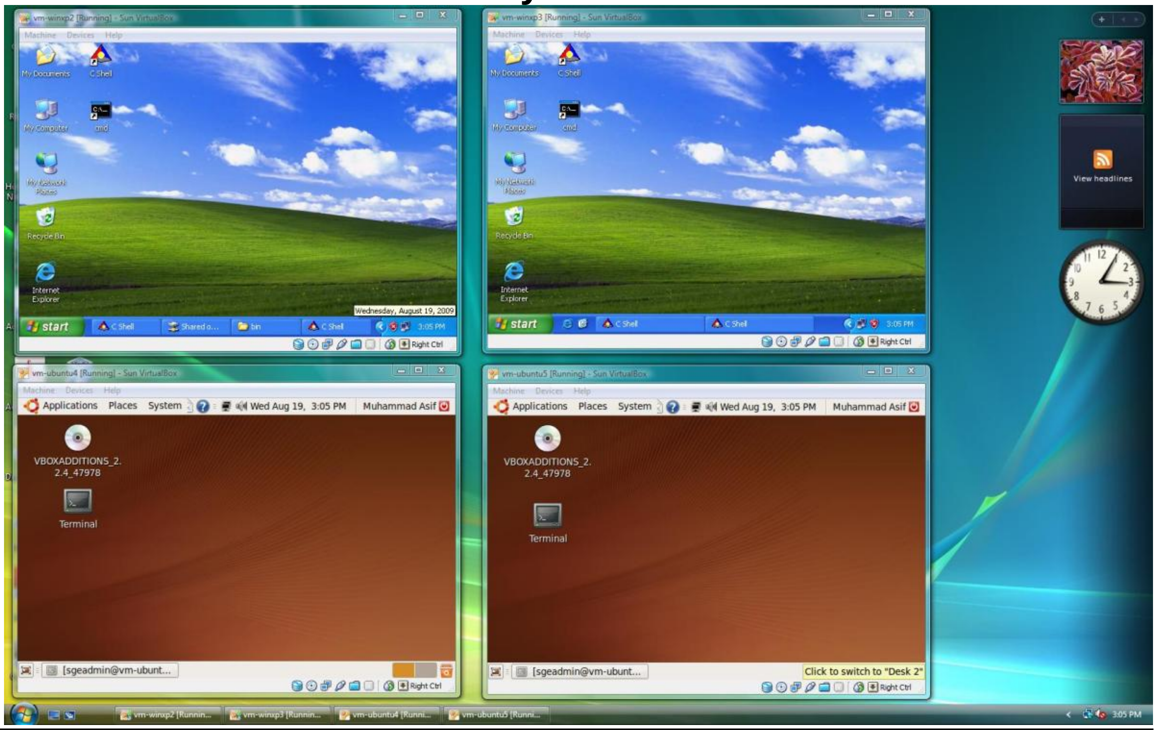
| | |
|---|---|
| Shared Folders: | 1 |

**Remote Display**
*Disabled*

## System 1


vm-ubuntu1 [Running] - Sun VirtualBox
vm-winxp1 [Running] - Sun VirtualBox
vm-ubuntu2 [Running] - Sun VirtualBox
vm-ubuntu3 [Running] - Sun VirtualBox
Applications Places System
Wed Aug 19, 3:09 PM
Muhammad Asif
Terminal
VBOXADDITIONS_2.2.4_47978
[sgeadmin@vm-ubunt...
[get_params.sh (~/sto...
[Update Manager]
Current workspace: "Desk 1"
View headlines
start

## System 2


vm-winxp2 [Running] - Sun VirtualBox
vm-winxp3 [Running] - Sun VirtualBox
vm-ubuntu4 [Running] - Sun VirtualBox
vm-ubuntu5 [Running] - Sun VirtualBox
Applications Places System
Wed Aug 19, 3:05 PM
Muhammad Asif
VBOXADDITIONS_2.2.4_47978
Terminal
Wednesday, August 19, 2009
[sgeadmin@vm-ubunt...
Click to switch to "Desk 2"
View headlines
start

## Appendix II

### Prior to the Installation/Configuration of Sun Grid Engine (SGE)

Use of the Sun Grid Engine requires a prior setup of the hardware and operating system software for cluster of workstations. For the research described in this paper we used two physical machines, eight virtual machines (four virtual machines installed on each system), and a dedicated network of virtual machines. Among these eight virtual machines, there were five 32-Bit Linux/UNIX machines and three 32-Bit Microsoft Windows machines.

After successful setup of hardware and proper installation of operating environments follow the instructions below which are essential for installation/configuration of SGE.

### Linux/UNIX Installation/Configuration

Sun Grid Engine supports different versions of Linux/Unix operating systems; the master host of the SGE is always configured on a Linux/UNIX platform, and controls the system.

In the described work we used the Ubuntu Linux (ubuntu-9.04-desktop-i386) on our Linux master host and on Linux/UNIX execution hosts; the main reason for this choice was that it is a reliable operating system, freely available on the Internet, and can be easily updated as well. After creating virtual machines with the help of VirtualBox we installed the Ubuntu Linux by providing its installation image or CD/DVD and following the default instructions. There are a few things which should be considered during the installation process; these are familiar for Unix administrators, and include selection of the hostname, IP address and the user login name and its password for the system and try to keep the configurations simple and similar (if possible), for example all the virtual machines have the same user login name and its password (like in our case that is “sgeadmin” and “imeasif” respectively).

VirtualBox has its own DHCP server which always sets the IP addresses for the network interfaces, there is one network interface having IP address 10.0.0.15 on every virtual machine and this is used by the VirtualBox as default communication link to control and command the virtual machine(s). Thus, for the deployment of the SGE and its operation, there is need to enable another interface from the VirtualBox settings option, and set the IP address for this interface manually.

After settings and testing the network interfaces, there is need to update the necessary packages and tools by following the instructions given below.

- sudo apt-get update
- sudo apt-get install build-essential

- sudo apt-get install g++
- sudo apt-get install fort77
- sudo apt-get install ssh

## Microsoft Windows and SFU Installation/Configuration

Microsoft Windows XP is the operating system for the applications which supports Windows architecture in our setup. The creation of virtual machine is very much similar like Ubuntu Linux created in Appendix I and after successful creation of having virtual machine(s) for WinXP then simply follows the traditional WinXP installation instructions to put the virtual machine into function.

Before starting the installation of execution host for WinXP, it is recommended to install the master host of the SGE and one execution host for Linux/UNIX platform.

Installation of the execution host of SGE on WinXP is not possible directly, therefore a software suit "Services for UNIX" (SFU) needs to be used which has been designed by Microsoft to enable Linux/UNIX services for Windows environment and this is freely available for 32-Bit WinXP on Microsoft website.

Login the WinXP as "Administrator" user and complete the following tasks before installation of SGE execution host for WinXP.

### I. Create files passwd and group

There is need of two files (passwd and group) which are required during the installation process of SFU. To create these files login the master host of SGE as "sgeadmin" user (in our case) and follow the following procedure.

```
sgeadmin@vm-ubuntu1:~/sge$ fgrep sgeadmin /etc/passwd

sgeadmin:x:1000:1000:Muhammad Asif,,,:/home/sgeadmin:/bin/bash
```

Now with the help above output write the required files like as follow

- passwd

  ```
  sgeadmin:x:1000:1000:SGE admin:/home/sgeadmin:/bin/csh
  ```

- group

  ```
  Users:x:1000:sgeadmin
  ```

### II. Disable DEP

Open the system properties window then go to “Advanced” tab, the option of Startup and Recovery has the “Settings” button. By clicking the settings button, another window will appear having “Edit” button and click it. A notepad will appear by opening “boot” system file with booting options. So, to disable the Data Execution Prevention modify the “noexecute” option like the following:

/noexecute=always off

## III. Install SFU/SUA

The system is ready for the installation of SFU now, download the SFU35SEL_EN and follow the instructions below (the un-selected options are crossed-out).

- Custom
    - Utilities
    - Interix GNU Components
        - Interix GNU Utilities
        - Interix GNU SDK
    - ~~NFS~~
        - ~~Client for NFS~~
        - ~~Server for NFS~~
    - ~~Password Synchronization~~
    - ~~Remote Connectivity~~
        - ~~Windows Remote Shell Service~~
    - ~~Authentication tools for NFS~~
        - ~~User Name Mapping~~
        - ~~Server for NFS Authentication~~
        - ~~Server for PCNFS~~
    - ~~Interix SDK~~
    - ~~Active State Perl~~
- Enable setuid
- Enable case sensitive filesystem
- Local User Name Mapping Server
    - Password and group files
- Leave file paths empty

## IV. Installation of OpenSSH for SFU/SUA

SUA community (http://www.suacommunity.com/pkg_install.htm) is actively designing the packages and tools for SFU Interix Sub system for Windows. There is need to install the bootstrap installer tool immediate after SFU installation for online updating of required packages and tools. Login the WinXP as “Administrator” user and download the bootstrap (pkg-current-bootstrap35.exe) tool from the above web link and install it.

Now login the “C Shell” as “Administrator” user and dispatch the following command.

```
$ pkg_update –L openssh
```

This will install the openssh and necessary openssl libraries. After the successful installation of openssh, you will find the “sshd” daemon running and ready to accept the ssh connections like we observe in Linux/UNIX environment.

## V. System Settings

Prior to the installation of the SGE execution host, there is need to perform necessary settings. Therefore, after successful installation of SFU and bootstrap tools, open the “C Shell” and login as “Administrator”.

Modify the /etc/inetd.conf file and enable the following services:

- telnet
- shell
- ftp
- login (if not enable yet)

Modify the /etc/hosts file for accessing the master host of SGE via hostname. The system name of WinXP will be the hostname of “C Shell”.

Modify the /etc/services file and add the following ports for SGE.

- sge_qmaster 6444/tcp
- sge_qmaster 6444/udp
- sge_execd 6445/tcp
- sge_execd 6445/udp

Create a directory “home” directory first on “/” path, inside the home directory create another directory “sgeadmin” for user and change the ownership of this sgeadmin directory as “sgeadmin” user.

```
$ chown –R VM-WINXP1+sgeadmin /home/sgeadmin
```

Now simply exit the “C Shell” and create another user “sgeadmin” from the control panel, the user must be member of “Users” group and set the profile path for this user like where we have recently created the “sgeadmin” directory by using “C Shell”.

## Domain Name System (DNS) Configuration

The configuration of Domain Name System (DNS) is not the essential part of SGE installation but it is recommended for the ease of work and easily remembering a lot of things going on in its operation on many hosts. In fact, it is not difficult to configure the DNS in Linux/UNIX environment; there is need to setup IP address of same IP scheme on all the systems which are going to play any role in Sun Grid Engine. Like in our case, we have eight (8) machines and the SGE master host has the following entries in its /etc/hosts file.

```
127.0.0.1 localhost

192.168.1.11 vm-ubuntu1

192.168.1.12 vm-ubuntu2

192.168.1.13 vm-ubuntu3

192.168.1.14 vm-ubuntu4

192.168.1.15 vm-ubuntu5

192.168.1.21 vm-winxp1

192.168.1.22 vm-winxp2

192.168.1.23 vm-winxp3
```

Therefore for the full operation of DNS, simply replicate the all above entries on all the /etc/hosts files of all Linux/UNIX and WinXP/SFU virtual machines.

Now all the machines can ping each other by supplying their hostnames instead of IP addresses and you will find that the DNS successful configured.

## Password-less SSH Configuration

As far as SGE configurations are concerned, the password-less SSH configuration for the cluster of workstations is not essential. In fact, SGE has its own powerful mechanism of communication among execution hosts. But as far as user defined data needs to be transferred to and fro SGE master host and SGE execution hosts dynamically, the password-less SSH makes the life easy. In addition, for MPI based parallel applications, it is a necessary part of cluster.

The cluster of workstations can be configured as password-less SSH by adopting multiple procedures; the main thing is to create a pair of keys (public and private) and distribute these keys among hosts. For achieving this one can follow the procedure summarized below; it is a simple procedure to make the cluster password-less SSH enabled. Login the SGE master host as “sgeadmin” user (as like in our case) and issue the following commands.

- rm -rf ~/.ssh/*
- ssh-keygen -t rsa
- cat ~/.ssh/id_rsa.pub  >> ~/.ssh/authorized_keys
- chmod 700 ~/.ssh
- chmod 644 ~/.ssh/authorized_keys

Now the last step is to copy the .ssh directory to all other hosts by using scp command, for this time and in fact for the last time it will ask about the password and the whole cluster of workstations will become password-less SSH configured (in our case the .ssh directory is being copied to all seven machines as shown below)

- scp ~/.ssh vm-ubuntu2:~/
- scp ~/.ssh vm-ubuntu3:~/
- scp ~/.ssh vm-ubuntu4:~/
- scp ~/.ssh vm-ubuntu5:~/
- scp ~/.ssh vm-winxp1:~/
- scp ~/.ssh vm-winxp2:~/
- scp ~/.ssh vm-winxp3:~/

## Appendix III

### Installation/Configuration of SGE

SGE is the batch scheduling system designed by Sun Microsystems. Due to its powerful features, excellent control over hosts and wide community of users, this is very popular batch scheduling system among industries and research institutes/organizations to efficiently manage the computing resources for their computational intensive (sequential and parallel) jobs.

Deployment of the Sun Grid Engine appears to be a little bit difficult, therefore it is helpful to be familiar with some tricks, especially important for configuring the execution host of Microsoft Windows and setting the Microsoft Windows applications and their environments. Therefore we provide below a brief overview of a rathe simple way of installation/configuration of Sun Grid Engine 6.2 Update 3 on 32-Bit Ubuntu Linux and 32-Bit WinXP.

### Prior to Installation of SGE

Prior to the installation of Sun Grid Engine make sure that all the following steps are perfectly working.

- Master host and execution host must be able to ping each other
- Similarly, the master host and execution host has to be able to resolve each other by their hostnames
- Password-less ssh access working among master host and execution hosts is optional
- Try to make the things simple and consistent on each host (either master or execution); for example:
    i. Create the set of default users, like “sgeadmin”
    ii. Keep the directory structure of SGE similar as well like “SGE_ROOT=/home/sgeadmin/sge”

### Installation of Master Host

The master host of SGE should always be on Linux/UNIX operating system. With respect to the architecture of the operating system, download the SGE suit from Sun’s website.

Login to the master host as sgeadmin user and unzip the SGE suit in some directory like in our case it is “/home/sgeadmin/sge’.

```
sgeadmin@vm-ubuntu1:~/sge$ ./inst_sge –m
```

-m switch is for installing master host of SGE, if master host is also going to perform as execution host in cluster then you may add another switch –x for that purpose while starting this bash script (inst_sge). It is not necessary to install master and execution hosts at the same time; in fact, after successful installation of master host, the execution host can later be installed on the same machine where master host is running.

The SGE scripts have been coded well. Therefore, in order to ease later installations of Windows execution hosts, one should - during the installation of the master host - generate a few special keys for enabling the Windows execution host support. In the end, the master host (sge_qmaster) will be successfully running, and can be checked by issuing the following command.

```
sgeadmin@vm-ubuntu1:~/sge$ ps aux | grep sge_
```

Before the installation of the execution host on another machine, one should perform the following set of actions:

1. Identify the ports on which sge_qmaster program is listening requests from execution hosts.

    - sge_qmaster 6444/tcp
    - sge_execd 6445/tcp (if an execution host is also installed on the same machine)

2. Select a name of the grid engine cell, which is the "default". There will be a folder in the SGE_ROOT directory with the same name. As far as installation of execution host is concerned this default directory is very important. Therefore, one should copy this default directory by some mean (scp, pen drive etc) into the SGE_ROOT directory of all execution hosts.
3. One should add the execution hosts into the list of admin hosts at master host. So, in our case for the time being, we are going to add seven execution hosts four Linux/UNIX execution hosts ("vm-ubuntu2" to "vm-ubuntu5") and three Microsoft Windows execution hosts ("vm-winxp1" to "vm-winxp3").

    - qconf –ah ubuntu2 (add (admin) host, similarly add rest of Linux hosts)
    - qcong –ah vm-winxp1 (Similarly add rest of WinXP hosts)
    - qconf –sh (this command will show the list of admin hosts)

**Installation of Execution Host on Linux/UNIX**

Login the system as sgeadmin user and unzip the SGE suit and issue the following command.

```
sgeadmin@vm-ubuntu2:~/sge$ ./inst_sge –x
```

As the username, directory structure and grid engine cell (default) all are same on execution host. In addition, the execution host has already added into the admin host list of master host. So, the things will be smooth and the execution host (sge_execd) will be successfully running on the system and can be verified like issuing the following command.

```
sgeadmin@vm-ubuntu2:~/sge$ ps aux | grep sge_
```

As far as the installation of execution host is concerned for this the execution host needs to be present in the list of admin hosts at the master host and rest of things the SGE will accommodate itself. After the successful installation of execution host login at master host again and assign the following job designations to the execution host.

- Submit Host (Submit hosts are the hosts which are able to submit the job on Grid Engine) (optional)
  - qconf –as vm-ubuntu2 (add submit host)
  - qconf –ss (this command will show the list of submit hosts)
- Execution Host (Execution hosts are the hosts which are able to execute the job)
  - qconf –ae vm-ubuntu2 (add execution host)
  - qconf –sel (this command will show the list of execution hosts)

Similarly rest of Linux/UNIX execution hosts can also be added into the Sun Grid Engine.

**Installation of Execution Host on WinXP**

Open the “C Shell” and login as “sgeadmin” user and rest of the procedure is similar as the execution host configured for Linux. Then download the SGE suit for windows from the Sun’s website, unzip the pack by keeping the directory structure same, and copy the grid cell directory from master host of the SGE (in our case the directory named “default”).

```
$ ./inst_sge –x
```

Then the execution host (sge_execd) should be successfully running on the system; this can be checked by issuing the following command:

```
$ ps aux | grep sge_
```

After a successful installation of execution host, its name shall be present in the list of admin host at the master host.To check this, login at the master host again, and assign the following job designations to the execution host.

- Submit Host (Submit hosts are the hosts which are able to submit the job on Grid Engine) (optional)
  - qconf –as vm-winxp1 (add submit host)
  - qconf –ss (this command will show the list of submit hosts)

- Execution Host (Execution hosts are the hosts which are able to execute the job)
    - qconf –ae vm-winxp1 (add execution host)
    - qconf –sel (this command will show the list of execution hosts)

Similarly all the rest of Microsoft Windows execution hosts can be added.

## Appendix IV

### Installation/Configuration of MPICH2 for Linux/UNIX and WinXP

For a high performance parallel processing on the cluster of workstations and massively parallel systems, one should choose suitable parallel programming model for the required parallel applications. Parallel Virtual Machine (PVM) and Message Passing Interface (MPI) are famous parallel programming models and popular among the community of parallel programmers.

In the context of this work, the MPI is our parallel programming model with the support of C, C++ and FORTRAN programming languages. The MPI is the specification for Application Programming Interface (API), according to MPI specification many organizations and research teams are constantly developing and updating various libraries. Among the cluster and grid computing community, the freely available versions of MPI like MPICH2 for Linux/UNIX and Microsoft Windows is preferred for high performance computing.

As far as parallel programming models are concerned, models configured for Linux/UNIX and those for Windows are not able to work together. Similarly, a single parallel application cannot be executed on Linux and Windows platforms together unless successfully recompiled for other platform.

### Installation of MPICH2 for Linux/UNIX

The version of MPICH2 (mpich2-1.1.1p1.tar) developed by Argonne National Laboratory (ANL) is used to embed with Sun Grid Engine for Linux/UNIX architecture and is freely available on the link below:

http://www.mcs.anl.gov/research/projects/mpich2/

Install/configure the MPICH2 on every workstation separately and then start the demons collectively from the host which is selected as master host of Sun Grid Engine by using “mpdboot” command.

To make the MPICH2 jobs operational with Sun Grid Engine, there are multiple methods to couple both MPICH2 and SGE. The major thing which is common for every method is to create some parallel environment by using the command “qconf –ap mpi” (“mpi” might be any name) and modify the option “pe_list” with this created parallel environment “mpi” at specific queue where the MPICH2 job will be submitted. The template for “mpi” is available within the suit of SGE in the folder “mpi” by the name of “mpi.template”.

## Installation of MPICH2 for WinXP

The MPICH2 installation/configuration for Microsoft Windows differs from that for Linux/UNIX architecture. Before the installation, one should install Microsoft .NET Framework and Microsoft Visual C++. The version of MPICH2 (mpich2-1.1.1p1-win-ia32.msi) actually requires WinXP (Service Pack 3). We used the MPICH2 (mpich2-1.0.8-win-ia32) version, which is freely available on the following link, and works correctly with WinXP (Service Pack 2):

http://www.mcs.anl.gov/research/projects/mpich2/downloads/tarballs/1.0.8/

After successful installation of MPICH2 on all WinXP workstations, the daemon “smpd” will be running on every WinXP workstation for computation of WinXP parallel jobs. As far as the submission of parallel job for WinXP architecture is concerned by using SGE, the method is almost same as the things happening in the case of Linux/UNIX architecture but for this the queue management is different.

# Appendix V

## Listing of developed Scripts

## submit.sh

```
#!/bin/sh

# $1 - name of the computational pack at pool

########################################
STORAGE=/home/sgeadmin/storage      # pool of computational packs
INPUT=$STORAGE/$1
RUN_SCRIPT=run_script.sh
LIN=linux.q
WIN=win.q
LINWIN=all.q
ARRAY=1

########################################
MPI=`cat $INPUT/META-INF/packinfo | grep "MPI" | cut -d "=" -f 2`;
NUM_PROC=`cat $INPUT/META-INF/packinfo | grep "NUM_PROC" | cut -d "=" -f 2`;

########################################
CMD_ARG=`cat $INPUT/META-INF/packinfo | grep "CMD_ARG" | cut -d "=" -f 2`;
ITERS=`cat $INPUT/META-INF/packinfo | grep "ITERS" | cut -d "=" -f 2`;

if [ $CMD_ARG =  ]; then
        lines=`wc -l $INPUT/META-INF/run_params.tsv | cut -d " " -f 1`
        ARRAY=$((ITERS*lines))
else
        ARRAY=$ITERS
fi

########################################
ARCH_TYPE=`cat $INPUT/META-INF/packinfo | grep "^EXE_" | cut -b 5- | cut -d "=" -f 1`
count=0

for arch in $ARCH_TYPE
do
        if [ $arch = lx24-x86 ]; then
                QUEUE=$LIN
                count=$((count+1))
        elif [ $arch = win32-x86 ]; then
                QUEUE=$WIN
                count=$((count+1))
        else
                echo "Architecture mismatch in packinfo"
                exit
        fi
done

########################################
if [ $MPI = yes ]; then
        if [ $count -eq 1 ]; then
```

```
		echo qsub -q $QUEUE -pe mpi $NUM_PROC -t 1:$ARRAY:1 $RUN_SCRIPT
$INPUT
		qsub -q $QUEUE -pe mpi $NUM_PROC -t 1:$ARRAY:1 $RUN_SCRIPT $INPUT
	elif [ $count -gt 1 ]; then
		echo "MPI Job can not be submitted for general queue"
	fi
elif [ $MPI = no ]; then
	if [ $count -eq 1 ]; then
		echo qsub -q $QUEUE -t 1:$ARRAY:1 $RUN_SCRIPT $INPUT
		qsub -q $QUEUE -t 1:$ARRAY:1 $RUN_SCRIPT $INPUT
	elif [ $count -gt 1 ]; then
		QUEUE=$LINWIN
		echo qsub -q $QUEUE -t 1:$ARRAY:1 $RUN_SCRIPT $INPUT
		qsub -q $QUEUE -t 1:$ARRAY:1 $RUN_SCRIPT $INPUT
	fi
fi

########################################
```

**run_script.sh**

```
#!/bin/sh

#$ -S /bin/sh
# $1 - name of computation pack at pool with path

########################################
PROGRAM=$1
QMASTER_HOSTNAME=vm-ubuntu1
OUTPUT=$JOB_ID.$SGE_TASK_ID.$HOSTNAME
LOG=jobinfo.txt

########################################
scp -r $USER@$QMASTER_HOSTNAME:$PROGRAM $OUTPUT

########################################
MPI=`cat $OUTPUT/META-INF/packinfo | grep "MPI" | cut -d "=" -f 2`;
REQS=`cat $OUTPUT/META-INF/packinfo | grep "REQS" | cut -d "=" -f 2`;
CMD_ARG=`cat $OUTPUT/META-INF/packinfo | grep "CMD_ARG" | cut -d "=" -f 2`;
STDIN_NAME=`cat $OUTPUT/META-INF/packinfo | grep "STDIN_NAME" | cut -d "=" -f 2`;


if [ $CMD_ARG =  ]; then
	CMD_ARG=`$OUTPUT/META-INF/get_params.sh $OUTPUT/META-INF/run_params.tsv
$SGE_TASK_ID`;
fi

if [ $MPI = yes ]; then
	MACHINES=`awk '{print $1}' $TMPDIR/machines`
	for machine in $MACHINES
	do
		if [ $machine != `hostname` ]; then
		`scp -r $OUTPUT $USER@$machine:~/`
		fi
	done
```

```
fi

#######################################
S_TIME=`date`

cd $OUTPUT

if [ $MPI = yes ] && [ $REQS = bin ]; then
cp $TMPDIR/machines .
        if [ $SGE_ARCH = lx24-x86 ]; then
                EXE_FILE=`cat META-INF/packinfo | grep "EXE_"$SGE_ARCH"=" | cut -d "=" -f
2`;
                MPI_ROOT=/home/sgeadmin/mpich
                $MPI_ROOT/bin/mpiexec -machinefile machines -np $NSLOTS ./$EXE_FILE
$CMD_ARG < META-INF/$STDIN_NAME
        elif [ $SGE_ARCH = win32-x86 ]; then
                EXE_FILE=`cat META-INF/packinfo | grep "EXE_"$SGE_ARCH"=" | cut -d "=" -f
2`;
                MPI_ROOT=/dev/fs/C/SFU/bin/mpich
                $MPI_ROOT/bin/mpiexec.exe -machinefile machines -np $NSLOTS
./$EXE_FILE $CMD_ARG < META-INF/$STDIN_NAME
        fi

elif [ $MPI = no ] && [ $REQS = bin ]; then
        if [ $SGE_ARCH = lx24-x86 ]; then
                EXE_FILE=`cat META-INF/packinfo | grep "EXE_"$SGE_ARCH"=" | cut -d "=" -f
2`;
                ./$EXE_FILE $CMD_ARG < META-INF/$STDIN_NAME
        elif [ $SGE_ARCH = win32-x86 ]; then
                EXE_FILE=`cat META-INF/packinfo | grep "EXE_"$SGE_ARCH"=" | cut -d "=" -f
2`;
                ./$EXE_FILE $CMD_ARG < META-INF/$STDIN_NAME
        fi

elif [ $MPI = no ] && [ $REQS = java ]; then
        if [ $SGE_ARCH = lx24-x86 ]; then
                EXE_FILE=`cat META-INF/packinfo | grep "EXE_"$SGE_ARCH"=" | cut -d "=" -f
2`;
                java -jar $EXE_FILE $CMD_ARG < META-INF/$STDIN_NAME
        elif [ $SGE_ARCH = win32-x86 ]; then
                EXE_FILE=`cat META-INF/packinfo | grep "EXE_"$SGE_ARCH"=" | cut -d "=" -f
2`;
                JAVA_ROOT=/dev/fs/C/SFU/bin/java/jdk1.6.0_14
                $JAVA_ROOT/bin/java.exe -jar $EXE_FILE $CMD_ARG < META-
INF/$STDIN_NAME
        fi
fi

E_TIME=`date`

#######################################
cd ..

if [ $SGE_TASK_ID = undefined ]; then
        cp $JOB_NAME.o$JOB_ID $OUTPUT/META-INF/stdout.txt
        cp $JOB_NAME.e$JOB_ID $OUTPUT/META-INF/stderr.txt
        if [ $MPI = yes ]; then
```

```
		cp $JOB_NAME.po$JOB_ID $OUTPUT/META-INF/stdpout.txt
		cp $JOB_NAME.pe$JOB_ID $OUTPUT/META-INF/stdperr.txt
	fi
else
	cp $JOB_NAME.o$JOB_ID.$SGE_TASK_ID $OUTPUT/META-INF/stdout.txt
	cp $JOB_NAME.e$JOB_ID.$SGE_TASK_ID $OUTPUT/META-INF/stderr.txt
	if [ $MPI = yes ]; then
		cp $JOB_NAME.po$JOB_ID.$SGE_TASK_ID $OUTPUT/META-INF/stdpout.txt
		cp $JOB_NAME.pe$JOB_ID.$SGE_TASK_ID $OUTPUT/META-INF/stdperr.txt
	fi
fi

echo JOB_NAME=$JOB_NAME >$OUTPUT/META-INF/$LOG
echo QUEUE_NAME=$QUEUE >>$OUTPUT/META-INF/$LOG
echo HOSTNAME=$HOSTNAME >>$OUTPUT/META-INF/$LOG
if [ $MPI = yes ]; then
	echo MACHINES= `cat $TMPDIR/machines` >>$OUTPUT/META-INF/$LOG
fi
echo ARCH=$SGE_ARCH >>$OUTPUT/META-INF/$LOG
echo TASK_ID=$SGE_TASK_ID >>$OUTPUT/META-INF/$LOG
echo EXE_FILE=$EXE_FILE >>$OUTPUT/META-INF/$LOG
echo STARTED=$S_TIME >>$OUTPUT/META-INF/$LOG
echo FINISHED=$E_TIME >>$OUTPUT/META-INF/$LOG

########################################
if  [ $MPI = yes ]; then
	for machine in $MACHINES
	do
		if [ $machine != `hostname` ]; then
			`ssh $USER@$machine rm -rf $OUTPUT`
		fi
	done
fi

scp -r $OUTPUT $USER@$QMASTER_HOSTNAME:~/
rm $JOB_NAME.*
rm -rf $OUTPUT

########################################
```

**get_params.sh**

```
#!/bin/sh

# $1 - file name "run_params.tsv"
# $2 - SGE_TASK_ID which is pointing to the line no. in fact

########################################
fname=$1

exec < $fname
i=0
while read line
do
	i=$((i + 1))
	if [ $i = $2 ]; then
		echo $line
```

```
                exit 0
        fi
done
echo "Wrong line number supplied, not enough lines in the file ****$i";
exit 1;
###########################################
```