UNIUNEA EUROPEANĂ

GUVERNUL ROMÂNIEI

Instrumente Structurale
2007-2013

Programul Operational Sectorial Cresterea Competitivitatii Economice, cofinantat prin Fondul European de Dezvoltare Regionala
Investitii pentru viitorul dumneavoastra

The NCIT Cluster Resources User's Guide

Version 3.1

Release date: 7 April 2010

Contents

1 Latest Changes

This is a document concerning the use of NCIT and CoLaborator cluster resources. It was developed during both the GridInitiative 2008 Summer School and the GridInitiative 2010 Summer School.

Several other versions will follow, please consider this simply as a rough sketch.

v.1.0	jul 2008	S.A, A.G	Initial release
v.1.1	2 nov 2008	H.A	Added examples, reformatted LaTex code
v.2.0	jul 2009	A.L, C.I.	Added chapters 7, 8 and 9 Updated chapters 3, 5 and 6
v.3.0	jul 2010	R.C., A.F.	Added chapters 5.2.1, 6.1.2, 6.1.4, 6.3, 6.4.2 Updated chapters 1-6
v.3.1	april 2011	H.A	Added new equipment, networking and capabilities

2 Introduction

The **NCIT** (National Center for Information Technology) of "Politehnica" University of Bucharest started back in 2001 with the creation of **CoLaborator**, a Research Base with Multiple Users (R.B.M.U) for High Performance Computing (HPC) , that took part of a World Bank project. **CoLaborator** was designed as a path of communication between Universities, at a national level, using the national network infrastructure for education and research.

Back in 2006, **NCIT**'s infrastructure was enlarged with the creation of a second, more powerful, computing site, more commmonly referred to as the **NCIT Cluster**.

Both clusters are used in research and teaching purposes by teachers, PhD students, grad students and students alike. In the near future a single sign-on (SSO) scenario will be implemented, with the same users' credentials across both sites, using the already existing LDAP infrastructure behind the `http://curs.cs.pub.ro` project.

This document was created with the given purpose of serving as an introduction to the parallel computing paradigm and as a guide to using the clusters' specific resources. You will find within the next paragraphs descriptions of the various existing hardware architectures, operating and programming environments, as well as further (external) information as the given subjects.

2.1 The Cluster

Although the two computing sites are different administrative entities and have different locations, the approach we will use throughout this document will be that of a single cluster with various platforms. This approach is justified also by the future 10Gb Ethernet link between the two sites.

Given both the fact that the cluster was created over a rather large period of time (and it keeps changing, since new machines will continue to be added), and that there is a real need for software testing on multiple platforms, the clusters structure is a heterogenous one, in terms of hardware platforms and operating/programming environments.

There are, currently, four different platforms available on the cluster:

- Intel x86 Linux

- Intel x86_64 Linux
- Sun SPARC Solaris
- Sun Intel x86_64 Solaris

Not all of these platforms are currently given a frontend, the only frontends available at the moment being **fep.hpc.pub.ro** and **fep.grid.pub.ro** machines (**F**ront **E**nd **P**rocessors). The machines behind them are all running non-interactive, being available **only** to jobs sent **through** the frontends.

2.2 Software Overview

The tools used in the **NCIT and CoLaborator cluster** for software developing are Sun Studio, OpenMP and OpenMPI. For debugging we use the TotalView Debugger and Sun Studio and for profiling and performance analysis - Sun Studio Performance Tools.

Sun Studio and GNU compilers were used to compile our tools. The installation of all the tools needed was done using the repository available online at `http://repository.grid.pub.ro/NCITRepo/NCITRepo.repo`.

2.3 Further Information

The latest version of this document will be kept online at `http://cluster.grid.pub.ro`.For any questions or feedback on this document, please feel free to write us at **emil.slusanschi@cs.pub.ro**.

3 Hardware

This section covers in detail the different hardware architectures available in the cluster.

3.1 Configuration

The following table contains the list of all the nodes available for general use. There are also various machines which are curently used for maintenance purposes. They will not be presented in the list below.

Model	Processor Type	Sockets/Cores	Memory	Hostname
IBM HS21 28 nodes	Intel Xeon E5405, 2 GHz	2/8	16 GByte	quad-wn
IBM HS22 4 nodes	Intel Xeon E5630, 2.53 GHz	2/16	32 GByte	nehalem-wn
IBM LS22 14 nodes	AMD Opteron	2/12	16 GByte	opteron-wn
IBM QS22 4 nodes	BECell Broadband	2/4	8 GByte	cell-qs
Fujitsu Esprimo 61 nodes	Intel P4 3 GHz	1/1	2 GByte	p4-wn
Fujitsu Celsius 2 nodes	Intel Xeon 3 GHz	2/2	2 GByte	dual-wn

Additionally, our cluster has multiple partnerships and you can choose to run your code to these remote sites using the same infrastructure and same id. What you must be carefull to consider is the different architecture of the remote cluster and the delay involved moving your data through the VPN links involved. (see section 3.6 for more info)

Model	Processor Type	Sockets/Cores	Memory	Hostname
ICF IBM HS21 65 nodes	Intel Xeon E5405, 2 GHz	2/8	16 GByte	quad-wnXX.hpc-icf.ro
CNMSI VCluster 120 nodes	Intel Xeon X5570, 2.93 GHz	1/1*	1 GByte	cnmsi-wn

(*) This is a virtualized infrastructure using VMWare ESX.

3.2 Processor Datashees

3.2.1 Intel Xeon Processors

The Intel Xeon Processor refers to many families of Intel's x86 multiprocessing CPUs for dual-processor (DP) and multi-processor (MP) configuration on a single motherboard targeted at non-consumer markets of server and workstation computers, and also at blade servers and embedded systems.The Xeon CPUs generally have more cache than their desktop counterparts in addition to

multiprocessing capabilities.

Our cluster is currently equipped with the Intel Xeon 5000 Processor sequence. (click link to see the datasheets)

Here is a quicklist to the processor datasheets available in our cluster.

CPU Name	Version	Speed	L2 Cache	Datasheet
Intel Xeon	E5405	2 Ghz	12Mb	click here
Intel Xeon	E5630	2.53 GHz	12MB	click here
Intel Xeon	X5570	2.93 Ghz	12MB	click here
Intel P4	-	3 Ghz	-	-

3.2.2 AMD Opteron

The Six-Core AMD Opteron processor-based servers deliver performance efficiency to handle real world workloads with a good energy efficiency. There is one IBM Chassis with 14 inifiniband connected Opteron blades available and the rest are used in our Hyper-V virtualization platform.

Our cluster is currently equipped with the Six-Core AMD Opteron Processor series. (click link to see datasheets)

CPU Name	Version	Speed	L2 Cache	Datasheet
AMD Opteron	2435	2.6Ghz	6x512Kb	click here

3.2.3 IBM Cell Broadband Engine

The QS22, based on the new IBM PowerXCell 8i multicore processor, offers extraordinary single-precision and double-precision floating-point computing power to accelerate key algorithms, such as 3D rendering, compression, encryption, financial algorithms, and seismic processing.
Our cluster is currently equipped with four Cells B.E.. (click link to see documentation link)

CPU Name	Version	Speed	L2 Cache	Datasheet
IBM PowerXCell 8i	QS22	3.2Ghz	-	CELL Architecture / PowerPC Architecture

Please reffer to the Computer Architecture Course (ASC) for a quick introduction on how to program on this platform. (only available in romanian language).

3.3 Server Datashees

Here is a shortlist of the server platforms we use.

3.3.1 IBM Blade Center H

There are five chassis and each can fit 14 blades in 9U. You can find general information about the model here. Currently there are four types of blades installed: Intel based **HS21** and **HS22** blades, AMD based **LS22** Cell based **QS22**.

3.3.2 HS 21 blade

There are 32 H21 blades of which 28 are used for the batch system and 4 are for developement and virtualization projects. Each blade has an Intel Xeon quad-core processor at 2Ghz with 2x6MB L2 cache, 1333Mhz FSB and 16GB of memory. Full specifications here.

You can access these machines using the **ibm-quad.q** queue in SunGridEngine and their hostname is *dual-wnXX.grid.pub.ro - 172.16.3.X*

3.3.3 HS 22 blade

There are 14 H22 blades of which 4 are used for the batch system and 10 are dedicated to the Hyper-V virtualization environment. Each blade has two Intel Xeon core processor at 2.53Ghz with 12MB L2 cache, 1333Mhz FSB and 32GB of memory. Full specifications here.

If you have a need for it, you can use 17 blades in the Hyper-V environment for HPC applications using Microsoft Windows HPCC. (but you're responsible to set-up your environment). All HS22 blades have FibreChannel connected storage and use high-speed FibreChannel Disks. These disks are only connected on demand. What you get using the batch system is a local disk.

You can access these machines using the **ibm-nehalem.q** queue in SunGridEngine and their hostname is *nehalem-wnXX.grid.pub.ro - 172.16.9.X*

3.3.4 LS 22 blade

There are 20 LS22 blades of which 14 are available for batch system use, the rest you can use in the Hyper-V Environment. Each blade has an Opteron six-core processor at 2,6Ghz. Full specifications here.

You can access these machines using the **ibm-opteron.q** queue in SunGridEngine and their hostname is *opteron-wnXX.grid.pub.ro - 172.16.8.X*

3.3.5 QS 22 blade

The Cell based QS22 blade features two dual core 3.2 GHz IBM PowerXCell 8i Processors, 512 KB L2 cache per IBM PowerXCell 8i Processor, plus 256 KB of local store memory for each eDP SPE. Memory capacity 8GB.

They have no local storage, ergo they boot over the network. For more features QS 22 features.
You can access these machines using the **ibm-cell-qs22.q** queue in SunGridEngine and their hostname is *cell-qs22-X.grid.pub.ro - 172.16.6.X*. You can also connect to these systems using a load-balanced connection at **cell.grid.pub.ro** (SSH).

3.3.6 Fujitsu Celsius R620

The Fujitsu-SIEMENS Celsius are workstations equipped with two Xeon processors. Because of their high energy consumption, they were migrated beginning January 2011 in the training and in the pre-production lab. A couple of them is still accessible using the batch system, and they are

used to host CUDA-capable graphics cards.

You can access theese machines using the fs-dual.q in SunGridEngine. Their hostname is *dual-wnXX.grid.pub.ro - 172.16.3.X*

3.3.7 Fujitsu Esprimo Machines

There are curently 61 Fujitsu Esprimo, model P5905, available. They each have an Intel Pentium 4 3.0Ghz CPU, with 2048KB L2 cache, 2048MB DDR2 man memory (upgradable to a maximum of 4GB) working at 533Mhz. Storage SATAII (300MB/s) 250 GB. More information can be found here.

You can acces theese machines using the fs-p4.q in SunGridEngine. If you have special projects that require physical and dedicated access to the machines, this will be the queue that you'll be using. Their hostname is *p4-wnXXX.grid.pub.ro - 172.16.2.X*.

3.3.8 IBM eServer xSeries 336

The IBM eServer xSeries 336 servers available at **NCIT Cluster** are 1U rack-mountable corporate business servers, each with one Intel Xeon 3.0 GHz processor with Intel Extended Memory 64 Technology and upgrade possibility, Intel E7520 Chipset Type and a Data Bus Speed of 800MHz.

They come with 512MB DDR2 SDRAM ECC main memory working at 400 MHz (upgradable to a maximum of 16GB), one Ultra320 SCSI integrated controller and one UltraATA 100 integrated IDE controller. Network wise, they have two network interfaces, Ethernet 10Base-T/100Base-TX/1000BaseT (RJ-45).
More information on the IBM eServer xSeries 336 can be found on IBM's support site, here.

Currently, these servers are part of the core system of the NCIT cluster and you do not have direct access to them.

3.3.9 Fujitsu-SIEMENS PRIMERGY TX200 S3

The Fujitsu-SIEMENS PRIMERGY TX200 S3 servers available at **NCIT Cluster** have each two Intel Dual-Core Xeon 3.0Ghz with Intel Extended Memory 64 Technology and upgrade possibility, Intel 5000V Chipset Type and a Data Bus Speed of 1066MHz. These processors have 4096KB of L2 Cache, ECC.

They come with 1024MB DDR2 SDRAM ECC main memory, upgradable to a maximum of 16GB, 2-way interleaved, working at 400 MHz., one 8-port SAS variant controller, one Fast-IDE controller and a 6-port controller.

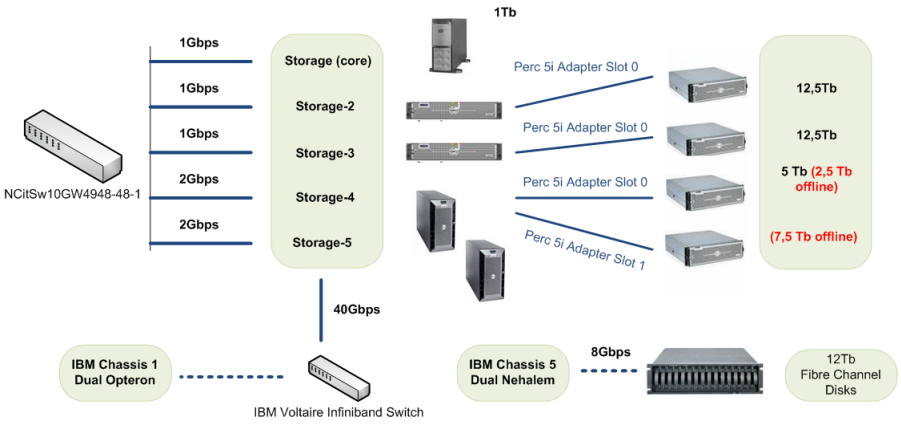
They have two network interfaces, Ethernet 10Base-T/100Base-TX/1000BaseT(RJ-45).
More information on the Fujitsu-SIEMENS PRIMERGY TX200 S3 can be found on Fujitsu-SIEMENS's site, here.

Currently, these servers are part of the core system of the NCIT cluster and you do not have direct access to them.

3.4 Storage System

The storage system is composed of the following DELL solutions: 2 PowerEgde 2900 and 2 PowerEdge 2950 servers, and 4 PowerVault MD1000 Storage Arrays.

There are four types of disk systems you can use **local disks, NFS, LustreFS and FibreChannel disks**.



All home directories are NFS mounted. There are several reasons behind this approach: many profiling tools can not run over LustreFS because of its locking mechanism and second, if the cluster is shut down, the time to start the Lustre filesystem is much greater than starting NFS.

The NFS partition is under: */export/home/ncit-cluster*. Jobs with high I/O are forbidden on the NFS directories.

Each user has also acces to a LustreFS directory. e.g. *~alexandru.herisanu/LustreFS* (symbolic link to */d02/home/ncit-cluster/prof/alexandru.herisanu)*. Opteron nodes are connected to the LustreFS Servers through Infinband, all the other nodes use one of the 4 LNET Routers to mount the filesystem. There are curently 3 OST servers and 1 MDS node, either available on Infiniband and TCP. Last but not least, each job has a default local scratch space available created by out batch system.

Type	Where	Observations
NFS	~HOME (/export/home/ncit-cluster)	Do not use I/O jobs here
LustreFS	~HOME/LustreFS (/d02/home)	
Local HDD	/scratch/tmp	Local on each node

Beginning December 2011, our virtualization platform has received a well-deserved upgrade - a new FibreChannel storage system, a 12Tb IBM DS 3950. We have an additional licence available so if necesarry, so if you have a really intensive I/O application and LustreFS is not an option, we can map some harddisks to one of the nehalems.

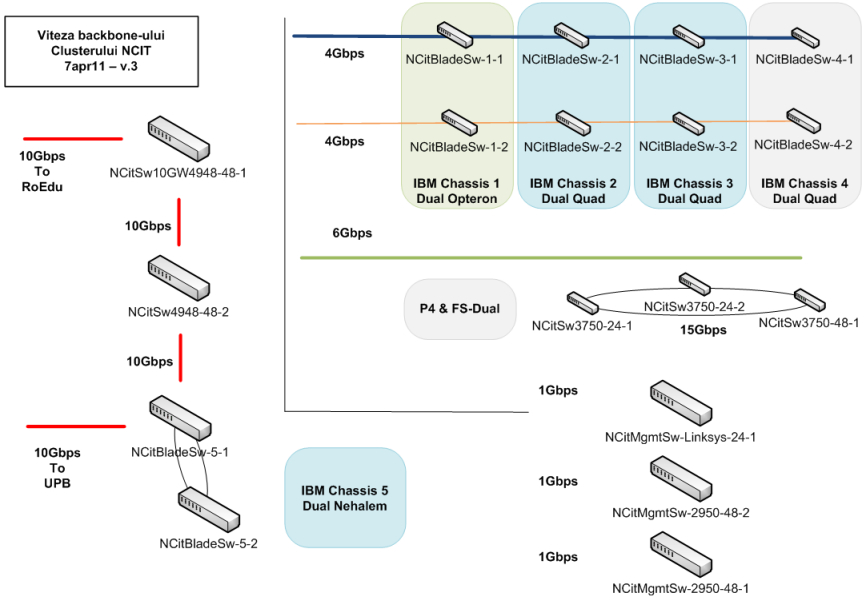
Here is a list of the ip's of the storage servers. The NFS Server is storage-2.

Hostname	Connected Switch	Port	IP
batch	NCitSw10GW4948-48-1	Gi1/37	172.16.1.1
	NCitSw10GW4948-48-1	Gi1/38	141.85.224.101
	NCitSw10GW4948-48-1	Gi1/39	N/A*
	Infiniband Voltaire	-	N/A*
storage	NCitSw10GW4948-48-1	Gi1/2	141.85.224.10
	NCitMgmtSw-2950-48-1	Fa0/4	172.16.1.10
storage-2	NCitSw10GW4948-48-1	Gi1/13	172.16.1.20
	NCitSw10GW4948-48-1	Gi1/14	141.85.224.103
	Infiniband Voltaire	-	192.168.5.20
storage-3	NCitSw10GW4948-48-1	Gi1/15	172.16.1.30
	Infiniband Voltaire	-	192.168.5.30
storage-4	NCitSw10GW4948-48-1	Gi1/3	172.16.1.40
	Infiniband Voltaire	-	192.168.5.40
storage-5	NCitSw10GW4948-48-1	Gi1/4	172.16.1.60
	NCitSw10GW4948-48-1	Gi1/7	141.85.224.49
	Infiniband Voltaire	-	192.168.5.60

(*)Will be the next MDS for the Lustre system beginning june 2011.
(**) See 3.5.2 to get the ipv6 addresses.

3.5 Network Infrastructure

The NCIT cluster has a full IPv6 routed network. The mainly used means of connecting is the Gigabit Ethernet. However, there are 14 of the 16 available Opteron six-core machines which are connected through InfiniBand (40 Gbps), all of them in the IBM Chassis 1.



We have a backbone speed of 10Gbps, composed of three main Cisco Switches, but please consider the network architecture when you're trying to profile your application.

The IBM Chassis 1-4 are connected to the same Catalyst4948 switch with 2x4 Gpbs links. Due to the fact that the two I/O modules are seen as different switches you can not bond the two interfaces on each blade, so you must manually load-balance your traffic through each connection. For MPI applications using OpenMPI, the framework does this for you automatically. It detects that it has two private vlans connecting node A and node B and load-balances automatically. The first interface of each blade is in Vlan6 - ClusterNodes network, the second is a trunk interface that gives you access to Vlan7 - MPI DataTransfer. IBM Chassis 5 uses the new I/O module and in this configuration, these switches are stacked together. The network configuration throughout the BladeChassis is identical.

The other hosts, Intel P4 and DualXeon only use the Vlan6 - ClusterNodes network. Please also note that you must consider blade placement when profiling your application and interpreting monitoring data. Here is a list on how the nodes are connected to the switches. You can ask the batch system to place your jobs on the same chassis and you can use this port mapping to get the right monitoring data from Cacti. (storage ips are here)

Hostname	Location	Connected Switch
opteron-wn{01-14}	IBM Chassis 1	NCitBladeSw-1-1 / NCitBladeSw-1-2
quad-wn{01-14}	IBM Chassis 2	NCitBladeSw-2-1 / NCitBladeSw-2-2
cell-qs22-{01-4}	IBM Chassis 3	NCitBladeSw-3-1 / NCitBladeSw-3-2
quad-wn{15-28}	IBM Chassis 4	NCitBladeSw-4-1 / NCitBladeSw-4-2
nehalem-wn{11-14}	IBM Chassis 5	NCitBladeSw-5-1 (stacked)
p4-wn{001-061}	Module D-H	NCitSw3750-48-1 (stacked)
dual-wn{01-02}	Module B	NCitSw3750-48-1 (stacked)

The port mapping in each switch is blade1 goes into port1. The QS22's use ports 11,12,13,14 in blade chassis 3.

The switches have a backbone speed ranging from 1,4,6 and 10Gbps. Here's our main switches.

Switch	Uplink ports	Uplink to	Ports in uplink switch	Speed
NCitSw10GW4948-48-1	Ten1/49	RoEdu	-	10Gbps
NCitSw10GW4948-48-2	Ten1/49	NCitSw10GW4948-48-1	Ten1/50	10Gbps
NCitSw3750-48-1	Gi2/0/19-24	NCitSw10GW4948-48-1	Gi0/43-48	6Gbps
NCitBladeSw-5-1	Ten1/0/1	NCitSw10GW4948-48-2	Ten1/50	10Gpbs
	Ten2/0/1	UPB	-	10Gbps
NCitBladeSw-1-2	Gi1/16-20	NCitSw10GW4948-48-2	Gi1/1-4	4Gbps
NCitBladeSw-1-2	Gi1/16-20	NCitSw10GW4948-48-2	Gi1/5-8	4Gbps
NCitBladeSw-2-1	Gi1/16-20	NCitSw10GW4948-48-2	Gi1/9-12	4Gbps
NCitBladeSw-2-2	Gi1/16-20	NCitSw10GW4948-48-2	Gi1/13-16	4Gbps
NCitBladeSw-3-1	Gi1/16-20	NCitSw10GW4948-48-2	Gi1/17-20	4Gbps
NCitBladeSw-3-2	Gi1/16-20	NCitSw10GW4948-48-2	Gi1/21-24	4Gbps
NCitBladeSw-4-1	Gi1/16-20	NCitSw10GW4948-48-2	Gi1/25-28	4Gbps
NCitBladeSw-4-2	Gi1/16-20	NCitSw10GW4948-48-2	Gi1/29-32	4Gbps

3.5.1 Available VLANs

The NCIT Cluster is not only a HPC Cluster. It is a playground for new projects and technologies, so if you need a special setup you cand use any of the following VLAN list to achieve your goal.

Vlan Name	VlanID	Ip range	Observations
Internet	2	141.85.224.0/24	Old internet RoEdu connction. We must release this range
Training	3	192.168.2.0/24	Training VLAN, this VLAN you get when you're using VPN
ClusterNodes	6	172.16.0.0/24	All cluster nodes are here, see server descriptions for actual ranges
		192.168.5.0/24	IP over Infiniband connection
MPI-DataTransfer	7	192.168.4.0/24	Used for load-balancing MPI on the two links
Cloud-VirtualMachines	9	10.42.0.0/16	All your experimental virtual machines go in here
NCIT-Internet	10	141.85.241.0/24	New internet range using 10GE
NCIT-DMZ	11	192.168.6.0/24	All your long term virtual machines go in here. Usually you'll get public access ports here.
NCIT-Exp-A		-	Experimental Vlan. Do what you like
NCIT-Exp-B		-	Experimental Vlan. Do what you like
NCIT-Exp-C		-	Experimental Vlan. Do what you like
Ipsec-GW		-	Other clusters. see 3.6.

3.5.2 Network Connections

Our main worker node router is a debian based machine. (*141.85.241.163, 172.16.1.7, 192.168.6.1, 10.42.0.1*). It also acts as a name-caching server. If you get a public ip directly, your routers are *141.85.241.1* and *141.85.224.1*, depending on the Vlan.

DNS Servers: *141.85.241.15, 141.85.164.62*

Our IPv6 network is **2001:b30:800:f0::/54**. To get the ipv6 address of a host just use the following rule:

```
IPv4 address: 172.16.1.7  | -> IPv6 address: 2001:b30:800:f006:172:16:1:7/64
Vlan: 6 (Cluster Nodes) |
```

f0[06], because f0 is part of the network and 06 because the host is in Vlan6. 172:16:1:7 is the IPv4 address of the host.

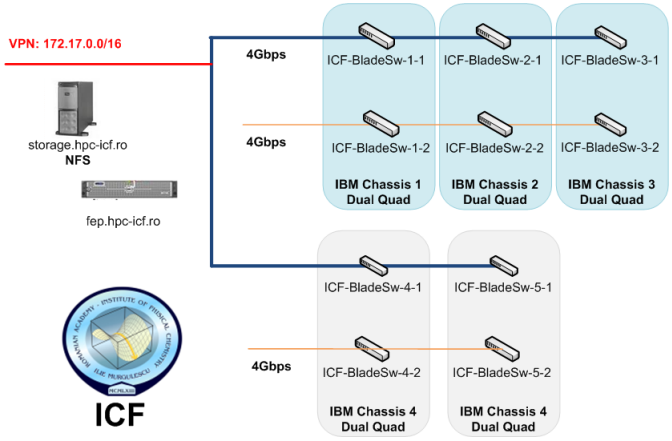
3.5.3 Configuring VPN

Sometimes you need acces to resources that are not reachable from the internet. If you have a **NCIT AD** account, then you can connect through VPN in the training network. Write us if you need a VPN account.

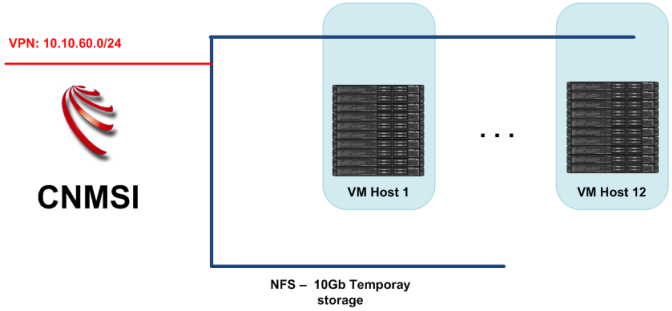
VPN Server: *win2k3.grid.pub.ro (141.85.224.36)*
We do not route your traffic so deselect the following option. Right click on the VPN Connection - Properties - Networking - TCP/IP v4 - Properties - Advances, deselect *Use default gateway on remote network.*

3.6 HPC Partner Clusters

You can also run jobs on partner clusters like HPC ICF and CNMSI Virtual Cluster. The ICF cluster uses IBM BladeCenter Chassis the same generation as our quad-wn nodes and the CNMSI Cluster uses a lot of virtualized machines with limited memory and limited storage space. Although you can use these systems in any project you like, please take note of the networking and storage architecture involved.



The HPC Cluster of the Institute of Physical Chemistry (http://www.icf.ro) is nearly identical to ours. There are 65 HS21 blades available with dual quad-core Xeon processors. The home directories are nfs-mouted through autofs. There are five chassis with 13 blades each. Currently, your home directory is mounted over the VPN link so it is advisable that you store your data locally using the scratch directory provided by SunGridEngine or a local NFS temporary storage. The use of the icf-hpc-quad.q queue is restricted. The ip range visible from our cluster is *172.17.0.0/16 - quad-wnXX.hpc-icf.ro.*



The second cluster is a collaboration between UPB and CNMSI (http://www.cnmsi.ro). They provide us with access to 120 virtual machines, each witch 10Gb of hard-drive and 1Gb of memory. The use of the *cnmsi-virtual.q* queue is also restricted. The ip range visible from our cluster is *10.10.60.0/24 - cnmsi-wnXXX.grid.pub.ro*.

4 Operating systems

There is only one operating system running in the **NCIT Cluster** and that is Linux. The cluster is split into a HPC domain and a virtualization domain. If you need to run windows applications we can provide you with the necessary virtualized machines, documentation and howtos, but you have to set it up yourself.

4.1 Linux

Linux is the UNIX-like operating system. It's name comes from the Linux kernel, originally written in 1991 by Linus Torvalds. The system's utilities and libraries usually come from the GNU operating system, announced in 1983 by Richard Stallman. The Linux release used at the **NCIT Cluster** is a RHEL (Red Hat Enterprise Linux) clone called Scientific Linux, co-developed by Fermi National Accelerator Laboratory and the European Organization for Nuclear Research (CERN).

The Linux kernel version we use is:

```
$ uname -r
2.6.18-194.26.1.el5
```

whereas the distribution release:

```
$ cat /etc/issue
Scientific Linux SL release 5.5 (Boron)
```

4.2 Addressing Modes

Linux supports 64bit addressing, thus programs can be compiled and linked either in 32 or 64bit mode. This has no influence on the capacity or precision of floating point numbers (4 or 8 byte real numbers), affecting only memory addressing, the usage of 32 or 64bit pointers. Obviously, programs requiring more than 4GB of memory have to use the 64bit addressing mode.

5 Environment

5.1 Login

Logging into UNIX-like systems is done through the secure shell (**SSH**). You can log into each one of the cluster's frontends from your local UNIX machine, using the ssh command. If you're a student at our faculty, it's the the same username as cs.curs.pub.ro. If you don't have an account please use the account request form available on cluster.grid.pub.ro.

```
$ ssh username@fep.grid.pub.ro
$ ssh username@cell.grid.pub.ro (connect directly to the QS22s)
```

Logging into one of the frontends from Windows is done by using Putty.

We provide three ways to connect to the cluster having a graphical environment. You can use **X11 Tunneling, VNC or FreeNX** to run GUI apps.

5.1.1 X11 Tunneling

The simple way to get GUI access is to use ssh X11 Tunneling. This is also the slowest method.

```
$ssh -X username@fep.grid.pub.ro
$xclock
```

Depending on your local configuration it may be necessary to use the -Y flag to enable the trusted forwarding of graphical programs. (Especially if you're a MAC user). If you're running Windows, you need to run a local X Server. A lightweight server (2Mb) is XMing.

To connect using Windows, run XMing, run Putty and select *Connection - SSH - X11 - Enable X11 forwarding*, and connect to fep.

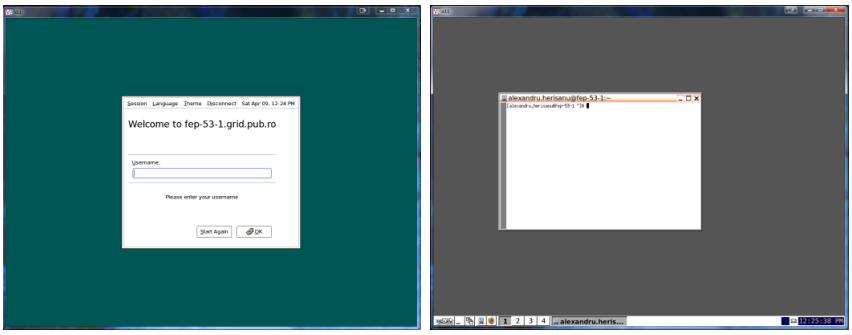
5.1.2 VNC

Another method is to use VNC. Due to the fact that VNC is not encrypted we use SSH port forwarding just to be safe. The frontend runs a configuration named VNC Linux Terminal Services, meaning that if you connect on port 5900 you'll get a VNC server with 1024x768 resolution, 5901 is 800x600 and so on. You can not connect directly so you must use ssh.

```
ssh -L5900:localhost:5900 username@fep.grid.pub.ro

On the local computer:
vncviewer localhost
```

First line connects to fep and creates a tunnel from your host port 5900 to fep (localhost:5900) port 5900. On your computer use vncviewer to connect to localhost.
If you use windows, use RealVNC Viewer and Putty. First configure tunneling in putty. Run putty

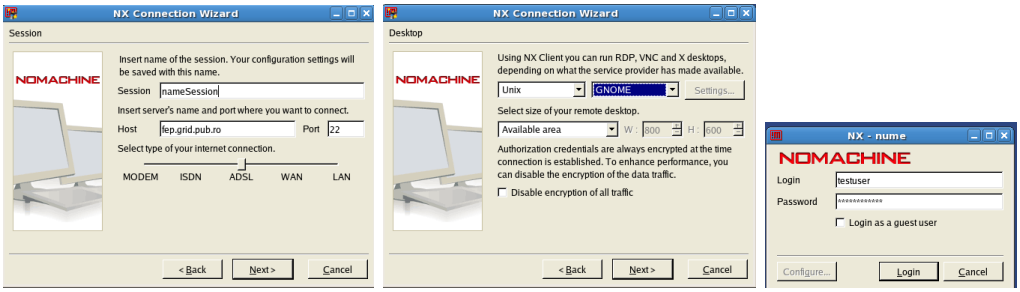
and select *Connection - SSH - Tunnels*. We want to create a tunnel from our machine, port 5900 to fep after we connect. So select Source port: 5900 and Destination localhost:5900 and click Add. Connect to fep and then use RealVnc and connect to localhost. You should get this:



Select ICE VM from Sessions. There is no GNOME or KDE Installed.

5.1.3 FreeNX

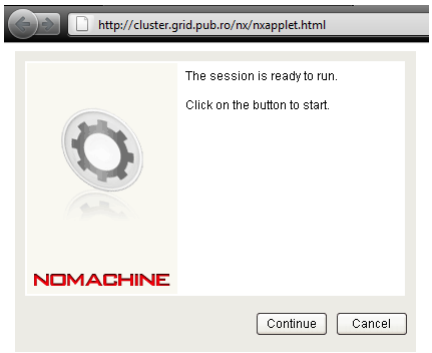
FreeNX uses a propietary protocol over a secondary ssh connection. It is the most efficient remote desktop for linux by far, but requires a client to be installed. NX Client is used both on Linux and Windows. After installing the client, run the NX Connection Wizard like in the steps below.



(*) You must actually select *Unix - Custom* in step2, as we do not use either Gnome or KDE but IceWM.

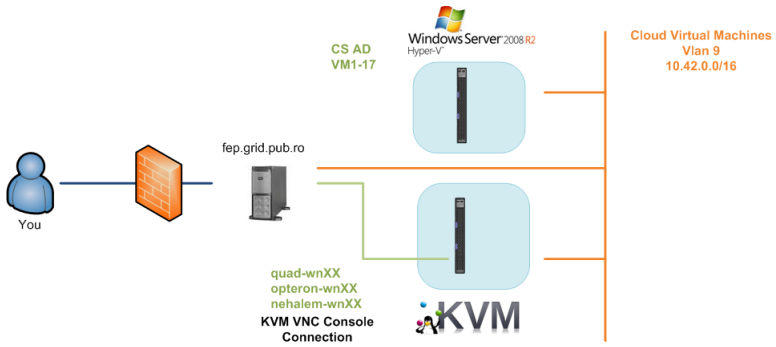
A more thorough howto can be found here. Our configuration uses the default FreeNX client keys.

You could also use our NXBuilder App to download, install and configure the connection automatically. Just point your browser to http://cluster.grid.pub.ro/nx and make sure you have java installed.



5.1.4 Running a GUI on your VirtualMachine

If you wish you can run your own custom virtual machine on any machine you like (Please refer to 5.4.4 for more details), but depending on the Virtual domain used, you may not have inbound internet acces. You can use ssh tunneling to access your machine and this is how to do it.



First of all, you must decide on one of the following methods you want to use: X11 tunneling, VNC or FreeNX. If you use KVM, then you can also connect to the console of the virtual machine directly.

KVM Console

When you run the virtual machine with KVM, you will specify the host or the queue where your machine runs. Using *qstat* or the output of *apprun.sh* get the machine that's running the VM. Ex: *opteron-wn01.grid.pub.ro*, port 11. Connect to the machine that hosts your VM directly using vncviewer either through X11 tunnelling or port forwarding.

```
a. X11 tunneling
$ssh -X username@fep.grid.pub.ro
$vncviewer opteron-wn01.grid.pub.ro:11

b. SSH Tunneling (tunnel the vncport on the localhost)
$ssh -L5900:opteron-wn01.grid.pub.ro:5911 username@fep.grid.pub.ro
on the local host
$vncviewer localhost
```

You can use any of the methods described ealier to get a GUI on fep. (X11,VNC or FreeNX). Use this method if you do not know your ip.

X11 Tunneling / VNC / FreeNX

If you know your ip, jst install SSH/VNC or FreeNX on your VM, connect to fep and connect with -X. For example if the ip of your machine is *10.42.8.1*:

```
a. X11 tunneling
$ ssh -X username@fep.grid.pub.ro
(fep)$ ssh -X root@10.42.8.1
(vm)$ xclock
```

```
b. SSH Tunneling (tunnel the remote ssh port on the localhost)
$ ssh -L22000:10.42.8.1:22 username@fep.grid.pub.ro
on the local host
$ ssh -P22000 -X root@localhost
```

All theese three methods rely on services you install on your machine. The best way is to port forward the remote port locally. In case of X11 and FreeNX you will tunnel the SSH port (22), in case of VNC, port 5900+.

For more information check theese howtos: http://wiki.centos.org/HowTos/VNC-Server (5. Remote login with vnc-ltsp-config) and http://wiki.centos.org/HowTos/FreeNX.

5.2 File Management

At the time of the writing of this section there were no quota limits on how much disk space you can use. If you really need it, we can provide it. Every user of the cluster has a home directory on an NFS shared filesystem within the cluster and a LustreFS mounted directory. Your home directory is usually **$HOME=/export/home/ncit-cluster/role/username**.

5.2.1 Tips and Tricks

Here are some tips on how to manage your files:

SCP/WinSCP

Transfering files to the cluster from your local UNIX-like machine is done through the secure copy command scp, e.g:

```
$ scp localfile username@fep.hpc.pub.ro:~/
$ scp -r localdirectory username@fep.grid.pub.ro:~/
(use -r when transferring muliple files)
```

The default directory where scp copies the file is the home directory. If you want to specify a different path where to save the file, you should write the path after ": For example:

```
$ scp localfile username@fep.hpc.pub.ro:your/relative/path/to/home
$ scp localfile username@fep.hpc.pub.ro:/your/absolute/path
```

Transfering files back from the cluster goes the same way:

```
$ scp username@fep.hpc.pub.ro:path/to/file /path/to/destination/on/local/machine
```

If you use Windows, use WinSCP. This is a scp client for Windows that provides a graphical file manager for copying files to and from the cluster.

SSH-FS Fuse

This is actually the best method you can use if you plan to edit files locally and run them remotely. First install sshFs (most distros have this package already). Carefull, you must use the absolute path of your home directory.

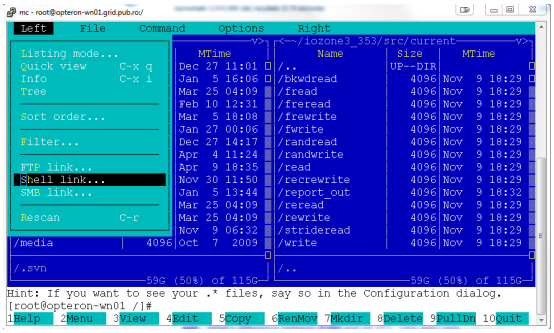
```
$ sshfs alexandru.herisanu@fep.grid.pub.ro:/export/home/\
ncit-cluster/prof/alexandru.herisanu /mnt
```

This allows you to use for example eclipse locally and see your files as local. Because it's a mounted file system, the transfer is transparent. (Don't forget to unmount). To see what your full home directory path is do this:

```
[alexandru.herisanu@fep-53-1 ~]$ echo $HOME
/export/home/ncit-cluster/prof/alexandru.herisanu
```

SSH Fish protocol

If you use Linux, you can use the Fish protocol. Install mc, F9 (Left) - Shell link - fep.grid.pub.ro. You now have a in the left pane of the file manager all your remote files.



The same protocol can be used from GNOME and KDE. From Gnome, *Places - Connect to Server* .... Select service type: SSH, Server: fep.grid.pub.ro, Port: 22, Folder: full path of the home directory (/export...), User Name: your username, Name for connection: NCIT Fep. After doing this you will have a desktop item that Nautilus will use to browse your remote files.

Microsoft Windows Share

Your home directory is also exported as a samba share. Connect through VPN and browse storage-2. You can also mount your home partition as a local disk. Contact us if you need this feature enabled.

Wget

If you want to copy an archive from a web-link in your current directory do this: Use *Copy Link Location* (from your browser) and paste the link as parameter for wget. For example:

```
wget http://link/for/download
```

5.2.2 Sharing Files Using Subversion / Trac

The NCIT Cluster can host your project on its SVN and Trac Servers. Trac is an enhanced wiki and issue tracking system for software development projects. Our SVN server is https://svn-batch.grid.pub.ro and the Trac system is here https://ncit-cluster.grid.pub.ro.

Apache Subversion, more commonly known as Subversion (command name svn) is a version control system. It is mostly used in software development projects, where a team of people may alter the same files or folders. Changes are usually identified by a number (sometimes a letter) code, called the "revision number", "revision level", or simply "revision". For example, an initial set of files is "revision 1". When the first change is made, the resulting set is "revision 2", and so on. Each revision is associated with a timestamp and the person making the change. Revisions can be compared, restred, and, with most types of files, merged.

First of all, a repository has to be created in order to host all the revisions. This is generally done using the create command as shown below. Note that any machine can host this type of repository, but in some cases such as our cluster you are required to have certain rights of access in order to create one.

```
$ svnadmin create /path/to/repository
```

Afterwards, the other users are provided with an address which hosts their files. Every user must install a svn version (e.g.: subversion-1.6.2) in order to have access to the svn commands and utilities. It is recommended that all the users involved in the same project use the same version.

Before getting started setting the default editor for the svn log messages is a good idea. Choose whichever editor you see fit. In the example below I chose vi.

```
$ export SVN_EDITOR=vim
```

Here are a few basic commands you should master in order to be able to use svn properly and efficiently:

- import - this command is used only once, when file sources are added for the first time; this part has been previously referred to as adding "revision 1".

```
$ svn import /path/to/files/on/local/machine  /SVN_address/New_directory_for_your_project
```

- add - this command is used when you what to add a new file to the ones that are already existent. Be careful though - this phase itself does not commit the changes. It must be followed by an explicit commit command.

```
$ svn add /path/to/new_file_to_add
```

- commit - this command is used when adding a new file or when submitting the changes made to one of the files. Before the commit, the changes are only visible to the user how makes them and not to the entire team.

```
$ svn commit /path/to/file/on/local/machine -m "explicit message explaining your changes"
```

- checkout - this command is used when you want to retrieve the latest version of the project and bring it to your local machine.

```
$ svn checkout /SVN_address/Directory_of_your_project /path/to/files/on/local/machine
```

- rm - this command is used when you want to delete an existing file from your project. This change is visible to all of your team members.

```
$ svn rm /address/to/file_to_be_deleted -m message explaining your action
```

- merge - this command is used when you want to merge two or more revisions. M and N are the revision numbers you want to merge.

```
$ svn merge sourceURL1[@N] sourceURL2[@M] [WORKINGPATH]
```

OR:

```
$ svn merge -r M:HEAD /SVN_address/project_directory  /path/to/files/on/local/machine
```

The last variant merges the files from revision M with the last revision existent.

- update - this command is used when you want to update the version you have on your local machine to the latest revision. It is also an easy way to merge your file with the changes made by your team before you commit your own changes. Do not worry. Your changes will not be lost. If by any chance, both you and the other members have modified the same lines in a file, a conflict will be signaled and you will be given the opportunity to choose the final version of that line.

```
$ svn update
```

For further information and examples check theese links SVN Redbook and SVN Tutorial.

5.3 Module Package

The **Module package** provides for the dynamic modification of the user's environment. Initialization scripts can be loaded and unloaded to alter or set shell environemnt variables such as `$PATH` or `$LD_LIBRARY_PATH`, to choose for example a specific compiler version or use software packages.

The advantage of the modules system is that environment changes can easily be undone by unloading a module. Dependencies and conflicts can be easily controlled. If, say, you need mpi with gcc then you'll just have to load both the gcc compiler and mpi-gcc modules. The module files will make all the necessary changes to the environment variables.

Note: The changes will remain active only for your current session. When you exit, they will revert back to the initial settings.

For working with modules, the **module** command is used. The most important options are explained in the following.

To get help about the module command you can either read the manual page (man module), or type

```
$ module help
```

To get the list of available modules type:

```
$ module avail
--------------------------- /opt/modules/modulefiles ---------------------------
apps/bullet-2.77                java/jdk1.6.0_23-32bit
apps/codesaturn-2.0.0RC1        java/jdk1.6.0_23-64bit
apps/gaussian03                 mpi/Sun-HPC8.2.1c-gnu
apps/gulp-3.4                   mpi/Sun-HPC8.2.1c-intel
apps/hrm                        mpi/Sun-HPC8.2.1c-pgi
apps/matlab                     mpi/Sun-HPC8.2.1c-sun
batch-system/sge-6.2u5          mpi/intelmpi-3.2.1_mpich
batch-system/sge-6.2u6          mpi/openmpi-1.3.2_gcc-4.1.2
blas/atlas-9.11_gcc             mpi/openmpi-1.3.2_gcc-4.4.0
blas/atlas-9.11_sunstudio12.1   mpi/openmpi-1.3.2_pgi-7.0.7
cell/cell-sdk-3.1               mpi/openmpi-1.3.2_sunstudio12.1
compilers/gcc-4.1.2             mpi/openmpi-1.5.0_gcc-4.1.2
compilers/gcc-4.4.0             mpi/openmpi-1.5.1_gcc-4.1.2
compilers/intel-11.0_083        oscar-modules/1.0.5(default)
compilers/pgi-7.0.7             tools/ParaView-3.8.1
compilers/sunstudio12.1         tools/ROOT-5.28.00
debuggers/totalview-8.4.1-7     tools/celestia-1.6.0
debuggers/totalview-8.6.2-2     tools/eclipse_helios-3.6.1
grid/gLite-UI-3.1.31-Prod       tools/scalasca-1.3.2_gcc-4.1.2
```

An available module can be loaded with

```
$ module load [module name] -> $ module load compilers/gcc-4.1.2
```

A module which has been loaded before but is no longer needed can be removed using
`$ module unload [module name]`

If you want to use another version of a software (e.g. another compiler), we strongly recom- mend switching between modules.
`$ module switch [oldfile] [newfile]`

This will unload all modules from bottom up to the oldle , unload the oldle , load the newle and then reload all previously unload modules. Due to this procedure the order of the loaded modules is not changed and dependencies will be rechecked. Furthermore some modules adjust their environment variables to match previous loaded modules.

You will get a list of loaded modules with
`$ module list`

A short information about the software initialized by a module can be obtained by

```
$ module whatis [file]
e.g.: $ module whatis compilers/gcc-4.1.2
compilers/gcc-4.1.2  : Sets up the GCC 4.1.2 (RedHat 5.3) Environment.
```

You can add a directory with your own module files with
`$ module use path`

Note : If you loaded module files in order to compile a program, you probably have to load the same module files before running that program. Otherwise some necessary libraries may not be found at program start time. This is also true if using the batch system!

5.4 Batch System

A batch system controls the distribution of tasks (batch jobs) to the available machines or resources. It ensures that the machines are not overbooked, to provide optimal program execution. If no suitable machines have available resources, the batch job is queued and will be executed as soon as there are resources available. Compute jobs that are expected to run for a large period of time or use a lot of resources should use the batchsystem in order to reduce load on the frontend machines.

You may submit your jobs for execution on one of the available queues. Each of the queues has an associated environment.

To display queues summary:

```
$ qstat -g c [-q queue]
CLUSTER QUEUE                   CQLOAD   USED    RES  AVAIL  TOTAL aoACDS  cdsuE
--------------------------------------------------------------------------------
all.q                             0.45      0      0    256    456    200      0
cnmsi-virtual.q                   -NA-      0      0      0      0      0      0
fs-p4.q                           -NA-      0      0      0      0      0      0
ibm-cell-qs22.q                   0.00      0      0      0     16      0     16
ibm-nehalem.q                     0.03     15      0     49     64      0      0
ibm-opteron.q                     0.72    122      0     46    168      0      0
ibm-quad.q                        0.37     90      0    134    224      0      0
```

5.4.1 Sun Grid Engine

To submit a job for execution over a cluster, you have two options: either specify the command directly, or provide a script that will be executed. This behavior is controlled by the "-b y—n" parameter as follows: "y" means the command may be a binary or a script and "n" means it will be treated as a script.

Some examples of submitting jobs (both binaries and scripts).

```
$ qsub -q [queue]  -b y [executable] -> $ qsub -q queue_1 -b y /path/my_exec

$ qsub -pe [pe_name] [no_procs] -q [queue]  -b n [script]
e.g: $ qsub -pe pe_1 4 -q queue_1 -b n my_script.sh
```

To watch the evolution of the submited job, use **qstat**. Running it without any arguments shows information about the jobs submited by you alone.

To see the progress of all the jobs use the -f flag. You may also specify which queue jobs you are interested in by using the -q [queue] parameter, e.g:

```
$ qstat [-f] [-q queue]
```

Typing "watch qstat" will automatically run qstat every 2 sec. To exit type "Ctrl-C".
In order to delete a job that was previously submitted invoke the qdel command, e.g:

```
$ qdel [-f] [-u user_list] [job_range_list]
```

where:
-f - forces action for running jobs
-u - users whose jobs will be removed. To delete all the jobs for all users use -u "*".

An example of submitting a job with SGE looks like that:

```
$ cat script.sh
#!/bin/bash
‘pwd‘/script.sh
$ chmod +x script.sh
$ qsub -q queue_1 script.sh   (you may omit -b and it will behave like -b n)
```

To display the sumited jobs of all users(-u "*") or a specified user, use:

```
$ qstat [-q queue] [-u user]
```

To display extended information about some jobs, use:

```
$ qstat -t [-u  user]
```

To print detailed information about one job, use:

```
$ qstat -j job_id
```

MPI Jobs need so called paralell environments. There are two MPI integration types: tight and loose. Tight integration means that sun grid engine takes care of running all MPI daemons for you on each machine. Loose, means that you have to boot up and tear down the MPI ring yourself. The Openmpi libraries use a tight integration (you use mpirun directly), and the Intel MPI library uses a loose integration (you must use mpdboot.py, mpiexec and mpdallexit). Each configured PE has a different scheduling policy. To see a list of paralell environments type:

```
[alexandru.herisanu@fep-53-1 ~]$ qconf -spl
make
openmpi
openmpi*1
sun-hpc
sun-hpc*1
```

qsub -q ibm-quad.q -pe openmpi 5 means you want 5 slots from the ibm-quad.q. Depending on the type of job you want to run simple/smp/mpi/hybrid (see **??**), you may need to know the scheduling type used on each paralell environment.

Basically *-pe openmpi 5* will schedule all five slots on the same machine, and *-pe openmpi*1 5* will schedule each mpi process on a different machine so you can use openmp or have a better I/O throughput.

You can find a complete howto for Sun GridEngine here:
http://wikis.sun.com/display/gridengine62u5/Home

5.4.2 Easy submit: MPRUN.sh

Mprun.sh is a helper script provided by us for an easier application profiling. When you test your application, you want to run the same application using different environment settings, compiler settings and so on. The mprun.sh script lets you run and customize your application using the command line.

```
$ mprun.sh -h
Usage: mprun.sh --job-name [job-name] --queue [queue-name] \
                --pe [Paralell Environment Name] [Nr. of Slots]   \
                --modules [modules to load] --script [My script]  \
                --out-dir [log dir] --show-qsub --show-script     \
                --batch-job

Example:
   mprun.sh --job-name MpiTest --queue ibm-opteron.q \
        --pe openmpi*1 3 \
        --modules "compilers/gcc-4.1.2:mpi/openmpi-1.5.1_gcc-4.1.2" \
        --script exec_script.sh \
        --show-qsub --show-script

   => exec_script.sh <=
```

```
# This is where you put what to run ...

mpirun -np $NSLOTS ./a.out

# End of script.
```

For example, you have an mpi program named a.out and you wish to test it using 1, 4 and 8 mpi processes on different queues and different scheduling options. All you need is a run script like:

```
mprun.sh --job-name MpiTest --queue ibm-opteron.q --pe openmpi 1 \
        --modules "compilers/gcc-4.1.2:mpi/openmpi-1.5.1_gcc-4.1.2" \
        --script exec_script.sh --show-qsub --show-script
mprun.sh --job-name MpiTest --queue ibm-nehalem.q --pe openmpi 2 \
        --modules "compilers/gcc-4.1.2:mpi/openmpi-1.5.1_gcc-4.1.2" \
        --script exec_script.sh --show-qsub --show-script
mprun.sh --job-name MpiTest --queue ibm-quad.q --pe openmpi*1 4 \
        --modules "compilers/gcc-4.1.2:mpi/openmpi-1.5.1_gcc-4.1.2" \
        --script exec_script.sh --show-qsub --show-script
```

Using this procedure, you can run your program, modify it and run the program using the same conditions. A more advanced feature is using different compilers end environmental variables for example you can use this script to run your program either with a tight integrated OpenMPI integration or a loose Intel MPI one.

```
MY_COMPILER=‘gcc‘
mprun.sh --job-name MpiTest --queue ibm-nehalem.q --pe openmpi 2 \
        --modules "compilers/gcc-4.1.2:mpi/openmpi-1.5.1_gcc-4.1.2" \
        --script exec_script.sh --show-qsub --show-script \
        --additional-vars MY_COMPILER
MY_COMPILER=‘intel‘
mprun.sh --job-name MpiTest --queue ibm-quad.q --pe openmpi*1 4 \
        --modules "compilers/intel-11.0_083:mpi/intelmpi-3.2.1_mpich" \
        --script exec_script.sh --show-qsub --show-script \
        --additional-vars MY_COMPILER
```

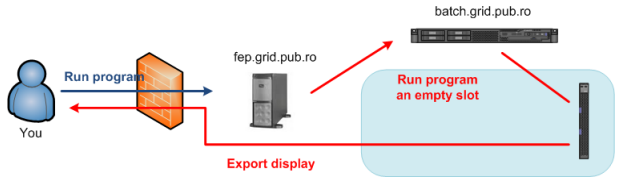
Your exec_script.sh must reflect these changes. When you run the execution script, you will also have access to the MY_COMPILER variable.

```
# This is where you put what to run ...

if [[ $MY_COMPILER == ‘‘intel" ]]; then
cat $PE_HOSTFILE | cut -f 1 -d ’ ’ > hostfile
mpdboot --totalnum=$NSLOTS --file=hostfile --rsh=/usr/bin/ssh
mpdtrace -l
mpiexec -np $NSLOTS ./hello_world_intel
mpdallexit
else
```

```
mprun -np $NSLOTS ./a.out
fi

# End of script.
```

5.4.3 Easy development: APPRUN.sh

Another tool build ontop of the SunGridEngine capabilities is Apprun. You can use apprun to easily run programs in the batch system and export the display home. This is how it works:



You connect to fep.grid.pub.ro using a GUI-capable connection (see). Use apprun.sh eclipse for example to schedule a job that will run eclipse on an empty slot. The graphical display will be exported through fep back to you. For example:

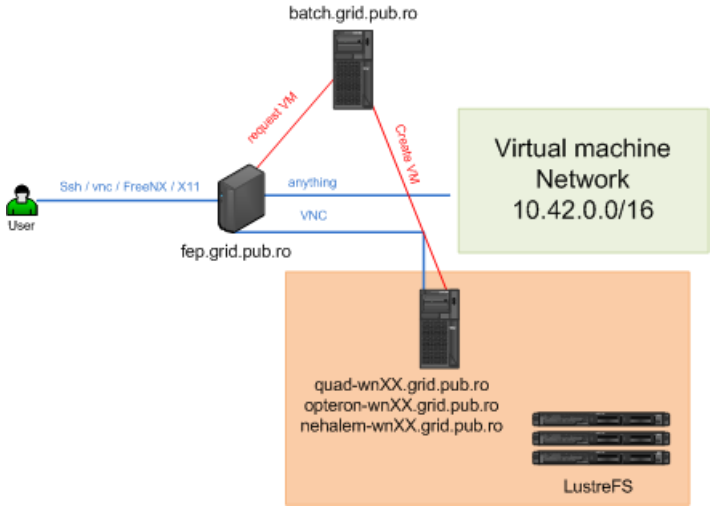
```
$ apprun.sh eclipse
```

will run eclipse. Curent available programs are: *eclipse and xterm - for using interactive jobs.*

5.4.4 Running a custom VM on the NCIT-Cluster

The NCIT Cluster has two virtualization stategies available: short-term virtual machines and long-term, internet connected VMs. The short-term Virtual Machines use KVM and the LustreFS Storage. They are meant for application scaling in cases where you need another type of operating system or root access. The long term VM domain is a Hyper-V R2 Cluster.

You must be part of the *kvm-users* group to run a KVM machine. Basically you use the SunGridEngine batch system to reserve resources (cpu and memory). This way the virtual machines will not overlap with normal jobs. All virtual machines are hosted on LustreFS, so you have a infiniband connection on the opteron nodes and 4x1Gb maximal total throughput if running on the other nodes.



This system is used for systems testing and scaling. You boot your machine once customize it, you shut it down and you boot several copy-on-write instances back up again. Copy-On-Write means the main disks are read-only and all the modified data is written to instance files. If you wish to revert to the initial machine, just delete the instance files and you're set. Additionally you can run as many instances you like without having to copy your master machine all over again.

The VM startup script also uses apprun.sh. For example:

```
##
## Master Copy (for the master copy)
#
#apprun.sh kvm --queue ibm-quad.q@quad-wn05.grid.pub.ro --vmname ABDLab --cpu 2 \
    --memory 2048M --hda db2-hda.qcow2  --status status.txt \
    --mac 80:54:00:01:34:01 --vncport 10 --master


##
## Slave Copy (for the copy-on-write instances)
#
apprun.sh kvm --queue ibm-quad.q@quad-wn01.grid.pub.ro --vmname ABDLab01 --cpu 2 \
    --memory 2048M --hda db2-hda.qcow2  --status status.txt --mac 80:54:01:01:34:01 \
    --vncport 11
apprun.sh kvm --queue ibm-quad.q@quad-wn02.grid.pub.ro --vmname ABDLab02 --cpu 2 \
    --memory 2048M --hda db2-hda.qcow2  --status status.txt --mac 80:54:02:02:34:02 \
    --vncport 11
```

See http://cluster.grid.pub.ro for a complete howto.

6 The Software Stack

Here's a shortlist of the software and middleware available on our cluster.

MPI

API	Flavor	Observations
OpenMPI v.1.5.1	Openmpi 1.5 Gcc 4.1.2 (if needed we can compile it for pgi , intel and sun)	The default mpi setup. On the opteron nodes, it uses the infiniband network by default. All TCP nodes use both ethernet cards to transmit MPI messages.
OpenMPI v.1.5	Openmpi 1.5 Gcc 4.1.2	No infiniband support compiled
OpenMPI v.1.3	Openmpi 1.3 Gcc 4.1.2 and 4.4.0, Intel, PGI and Sun Compiler supported	No infiniband support compiled

Environments: mpi/openmpi-1.3.2fill_gcc-4.1.2, mpi/openmpi-1.3.2_gcc-4.4.0, mpi/openmpi-1.3.2_pgi-7.0.7, mpi/openmpi-1.3.2_sunstudio12.1, mpi/openmpi-1.5.0_gcc-4.1.2, mpi/openmpi-1.5.1_gcc-4.1.2 (You need to load the compiler before the mpi environment. For example if you use mpi/openmpi-1.3.2_pgi-7.0.7 then a pgi 7.0.7 module load is required).

Website: http://www.open-mpi.org/
SDK Reference: http://www.open-mpi.org/doc/v1.5/
http://www.open-mpi.org/doc/v1.3/

API	Flavor	Observations
Sun-HPC8.2.1c	Openmpi 1.4 Gcc, Intel, PGI and Sun Compiler supported	Sun Cluster Tools 8.2.1c is based on Openmp 1.4. Infiniband support is provided.

Environments: mpi/Sun-HPC8.2.1c-gnu, mpi/Sun-HPC8.2.1c-intel, mpi/Sun-HPC8.2.1c-pgi, mpi/Sun-HPC8.2.1c-sun

Website: http://www.oracle.com/us/products/tools/message-passing-toolkit-070499.html
SDK Reference: http://download.oracle.com/docs/cd/E19708-01/821-1319-10/index.html

API	Flavor	Observations
Intel MPI 3.2.1	MPICH v2 flavor	Loose MPI implementation

Intel compiler There is no tight integration setup here. You must build your own mpi ring using mpdboot and mpiexec

Website:
SDK Reference: -
Sun Grid Engine uses a loose MPI integration for Intel MPI. You must start the mpd ring manually. For example:

```
cat $PE_HOSTFILE | cut -f 1 -d ' ' > hostfile
mpdboot --totalnum=$NSLOTS --file=hostfile --rsh=/usr/bin/ssh
mpdtrace -l
mpiexec -np $NSLOTS ./hello_world_intel
mpdallexit
```

When using a paralell environment, Sun Grid Engine exports the file containing the reserved nodes in the variable PEfill_HOSTFILE. The first line, parses that file and rewrites it in MPICH format. The communication is done using ssh public key authentication.

Development software

The current compiler suite is composed of GCC 4.1.2, GCC 4.4.0, Sun Studio 12 U1, Intel 11.0-83, PGI 7.0.7 and PGI 10. Additionaly on the CELL B.E platform there is a IBM XL compiler for C and Fortran available, but currently unaccesible.

Supported java version are Sun Java 1.6 U23 and OpenJDK 1.6. We provide Matlab 14 access by running it on VMWare Machines. The main profiler application is Sun Studio Analyser and Collector and Intel VTUNE. Other MPI profiling tools available are Scalasca (and its viewer cube3).

Current Math libraries available: Intel MKL, NAG (for C and Fortran currently unavailable) and ATLAS v.9.11 (compiled for gcc and sunstudio on the Xeon nodes).

Current debugging tools: TotalView 8.6.2, Eclipse Debugger (GDB), Valgrind We provide acces to a remote desktop to use all GUI enabled applications. We have a dedicated set of computers on witch you can run Eclipse, SunStudio, TotalView etc and export your display locally. We currently provide remote display capability through FreeNX, VNC or X11 Forwarding.