Hybrid Computer Cluster with High Flexibility

Shuo Liang
School of Computing and Engineering
University of Huddersfield
Huddersfield, United Kingdom
Email: shuo.liang@hud.ac.uk

Dr Violeta Holmes
School of Computing and Engineering
University of Huddersfield
Huddersfield, United Kingdom
Email: v.holmes@hud.ac.uk

Ibad Kureshi
School of Computing and Engineering
University of Huddersfield
Huddersfield, United Kingdom
Email: i.kureshi@hud.ac.uk

***Abstract*—In this paper we present a cluster middleware, designed to implement a Linux-Windows Hybrid HPC Cluster, which not only holds the characteristics of both operating system, but also accepts and schedules jobs in both environments. Beowulf Clusters have become an economical and practical choice for small-and-medium-sized institutions to provide High Performance Computing (HPC)resources. The HPC resources are required for running simulations, image rendering and other calculations, and to support the software requiring a specific operating system. To support the software, small-scale computer clusters would have to be divided in two or more clusters if they are to run on a single operating system. The x86 virtualisation technology would help running multiple operating systems on one computer, but only with the latest hardware which many legacy Beowulf clusters do not have. To aid the institutions, who rely on legacy non-virtualisation-supported facilities rather than high-end HPC resources, we have developed and deployed a bi-stable hybrid system built around Linux CentOS 5.5 with the improved OSCAR middleware; and Windows Server 2008 and Windows HPC 2008 R2. This hybrid cluster is utilised as part of the University of Huddersfield campus grid.**

***Keywords*-cluster middleware, computer cluster, job scheduler, resource manager**

I. Introduction

This paper is motivated by a growing need for High-Performance Computing (HPC) resources in Higher Education Institutions and the development of computer clusters built from Commercial off-the-shelf components (COTS) using open source cluster middleware. Building computer clusters with legacy x86 hardware is an effective and practical choice for providing affordable HPC resources. In order to cater for the variety of user demands, those Beowulf clusters are expected to support applications that require different hardware and software platforms.

Many universities in the UK started to establish their departmental and campus clusters in the last decade, e.g. a computer cluster at the University of Nottingham was built in 2004/5, and it had hit the June 2005 TOP500. [1]

In the University of Huddersfield there are a number of research groups requiring HPC resources for molecular and fluid dynamics simulation, 3D rendering, etc. using software such as DL-POLY, ANSYS, and Backburner. Some packages can be run on Linux or Windows operating systems while the others support multi-platform, supplying different user interfaces and document support.

One way of running these applications on different operating systems is to divide a computer cluster into smaller sub-clusters for each platform, which would lead to a duplication and poor utilisation of the resources. Another solution is to implement virtualised systems. Although the latest virtualisation technology could be a better alternative, it cannot be implemented effectively on legacy hardware. Hence, the aim of our research work is to devise and implement a system that will enable better utilisation of the HPC resources and support multi-platform applications used by our research community.

Software Name	Description	OS
Abaqus	Finite Element Analysis	L
Amber	Assisted Model Building with Energy Refinement aimed at biological systems	L
Backburner	Rendering software for 3ds Max	W
Blender	Open Source 3D Modeller and Renderer	L
CASTEP	CAmbridge Sequential Total Energy Package	L
COMSOL	Multiphysics Modelling, Finite Element Analysis, Engineering Simulation Software	W&L
DL_POLY	General purpose classical molecular dynamics (MD) simulation software	L
ANSYS FLUENT	Computational Fluid Dynamics (CFD)	W&L
GAMESS-UK	Molecular QM code	L
GULP	General Utility Lattice Program	L
LAMMPS	Large-scale Atomic/Molecular Massively Parallel Simulator	L
MATLAB	Numerical Computing Environment	W&L
METADISE	Minimum Energy Techniques Applied to Defects, Interfaces and Surface Energies	L
NWChem	Multi-purpose QM and MM code	L
Opera	Finite Element Analysis for Electromagnetics	W

Table I
Applications on the Huddersfield Campus Cluster
(W:Windows, L:Linux)

Our initial objective was to devise a simple dual-boot system to support both Windows and Linux platforms for the software in table I. A number of case studies were

conducted to identify the efficiency and ease of use of these software from the scientific and administrative users point of view. Based on the results from our case studies and the initial dual-boot system evaluation, the improved hybrid system design had been developed. This system provides more effective multi-platform middleware, which utilises existing hardware to provide the maximum performance in both Linux and Windows based environments.

The rest of the paper is organised as follows: Section II introduces our motivation; Section III describes the initial simple dual-boot system design and implementation; Section IV describes our proposed hybrid system design. Section V summarises the results and outlines possible future development.

II. Motivation

As a medium-sized Higher Education institution, the University of Huddersfield is still finding its place within the research community. The HPC Computing Resources centre was built to aid new research efforts requiring powerful computational resources. As part of the University Queensgate campus grid project, a number of computer clusters were built with Windows and Linux operating systems to cover the requirements of Linux Scientific Computing Software and Windows exclusive software [2]. While maintaining the Queensgate Clusters, we had identified that the Clusters need automated software to handle the specialisation for dual-boot, re-imaging, etc. The need for multiple operating systems is driven by both user needs and application requirements.

A lot of closed-source non-Java software, especially some large-scale Windows software, that use a large number of system exclusive API, have weaknesses in system platform migration. Since their source code is not published, the user can only run the program on the platform which the developer has offered. Although some of the companies offer multi-platform supports, others would not change a platform policy for the benefit of a small group of users.

In case of an open-source software, not every open-source project gives multi-platform support due to developers finite focus or small demand. An open-source project is capable to be forked from another new project in order to support new platforms or functions according to the developers need.

There are several solutions, which could enable deployment of both Windows and Linux OS on the same system. We have considered **virtualisation** and **multi-boot** technologies.

Virtualisation technology is not a new concept; it had been developed and used in mainframe computers. In recent years, the virtualisation has become applicable to PC and Workstation based machines since Intel (VT-x) and AMD (AMD-V) have started to support hardware-assisted virtualisation for x86 architecture. With hardware enhancements, the guest OS can achieve better performance and more compatibility. However, hardware support was not provided for their entire range of products.

A *multi-boot* computer could be implemented by various approaches, such as:

1) Changing active partition
2) Multi-system bootloader, e.g. GRUB (for Linux as main system), GRUB4DOS (for Windows as main system)
3) PXE also can enable multi-boot function

Pros: Multi-boot solution has wide range compatibility of hardware. It does not require new technology, and there is no performance reduction.

Cons: Reboot takes time, normally about 5 mins.

A Beowulf cluster at the University of Huddersfield was built from re-used laboratory computers with Intel CoreTM 2 Quad-core Q8200 processor that have no virtualisation support. Hence, we had to consider other ways to enable the multi-platform cluster.

A multi-boot approach is in our opinion, better suited for the legacy machines that have no hardware virtualisation support. This system could sit alongside High-end HPC resources making an efficient use of legacy machines.

III. The initial dual-boot OSCAR system

Because of native incompatibility between Windows system and UNIX-like system, and the above-mentioned solutions limitations, the multi-boot computer cluster system could be a viable solution for legacy hardware.

A. System architecture

In our initial approach, the dual-boot cluster was implemented on a small-scale cluster with 16 compute nodes (COTS computers) and 64 processors. Our dual-boot system is illustrated in Figure 1.

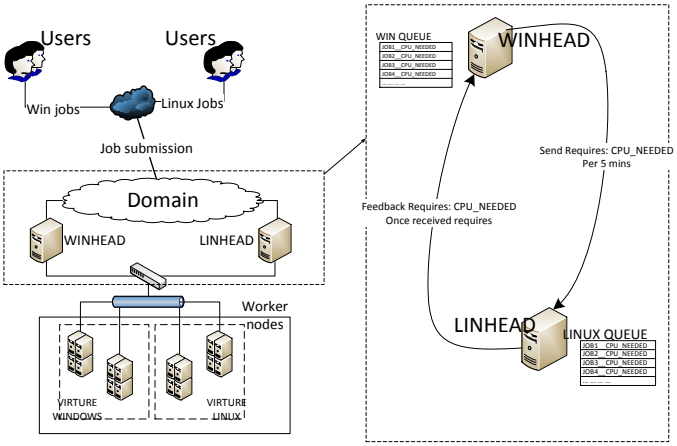

Figure 1. The initial dual-boot system

Windows and Linux head nodes had Windows HPC server 2008 R2 and CentOS 5.4 with OSCAR 5.1 beta 2

installed respectively and provided the complete access to the Windows and Linux clusters.

Windows HPC 2008 R2 offers a full GUI for system managers. Administrators can easily deploy a cluster by using several steps of configurations that involve network configuration, creation of node templates and deployment of nodes. All the rebooted compute nodes could be back executable without additional setup. An additional program was created and implemented on the Windows head node to enable the communication with the Linux head node.

On the Linux head node, we have used OSCAR, which is an open-source cluster middleware kit and has a friendly and clear interface for setting up a Linux cluster. It supports various Linux distributions, which gives more flexibility, easier maintenance and quicker updates. OSCAR wizard supports cluster head node installation, configuration of cluster packages and building of the worker nodes images, and complete cluster installation. In our clusters, CentOS 5.4 works well with OSCAR 5.1 beta 2. All our developments are based on these versions of software. Another program was created and implemented on the Linux head node (‘LINHEAD in figure) to communicate with the program in Windows head node (‘WINHEAD in figure).

Furthermore, we have developed a cluster middleware package **dualboot-oscar**.

dualboot-oscar can be divided by function into dual-boot /bf controller and dual-boot **deployment**.

B. Dual-boot Controller

1) A Implementation of dual-booting: The idea to implement automatic dual-boot system came from an article of an IBM engineer [3]. It offers a method to switch a single machine from current system to another system which is in another partition.

GRUB (GNU GRand Unified Bootloader) is a bootloader for Unix-like system and used in most Linux distributions. It could be installed to the MBR (Master boot record) section or head of a primary partition. When it is installed on MBR section, it boots any primary partition and logical partition. If GRUB is installed in MBR, it will ignore active partition. Instead, it reads its configuration file and follows its own logic.

By creating a shared partition, which is formatted to FAT (File Allocation Table) file system, a GRUB configuration file `controlmenu.lst` could be stored at this FAT partition. The default GRUB configuration file `menu.lst` in the Linux EXT3 partition is relocated to the file in FAT partition, by adding the command `configfile` in `menu.lst`. After these procedures, both Windows and Linux were given the read/write access to GRUB configuration file. Thus, they can both control default OS. With the OS switching script, which can edit or replace the file `controlmenu.lst` with administrator system permission, compute node can be easily switched between two different systems.

For switching in a multi-boot system a Perl script `bootcontrol.pl` written by Carter [3] is used to modify GRUB configuration file. Figure 2 shows the `menu.lst` in its original place, `/boot/grub/menu.lst`. `menu.lst` is set redirected to the file `controlmenu.lst` , which can be seen in Figure 3 and is located in the FAT partition for controlling default operating system.

To reduce the installations in Windows compute node, Carter's universal Perl script was replaced by our new Windows and Linux batch scripts, e.g. Windows .bat or Linux .sh file, which rename files from `controlmenu_to_linux.lst` or `controlmenu_to_windows.lst` into `controlmenu.lst`. The two files `controlmenu_to_linux.lst` and `controlmenu_to_windows.lst` are pre-configured and copied into FAT partition.

```
1  default=0
2  timeout=5
3  splashimage=(hd0,1)/grub/splash.xpm.gz
4  hiddenmenu
5
6  title changing to control file
7          root (hd0,5)
8          configfile /controlmenu.lst
```

Figure 2. An example of modified `menu.lst`

```
1  default 0
2  timeout=10
3  splashimage=(hd0,1)/grub/splash.xpm.gz
4
5  title CentOS-5.4_Oscar-5b2-linux
6  root (hd0,1)
7  kernel /vmlinuz-2.6.18-164.el5 ro root=/
       dev/sda7 enforcing=0
8  initrd /sc-initrd-2.6.18-164.el5.gz
9
10 title Win_Server_2K8_R2-windows
11 rootnoverify (hd0,0)
12 chainloader +1
```

Figure 3. An example of modified `controlmenu.lst`

2) Batch job triggered dual-boot system: The batch job triggered dual-boot system divides the entire dual-boot system to scheduling and executing of dual-boot actions.

The system switching action is packed as a PBS or Windows HPC job script, which locates a single node, modifies GRUBs configure file, and reboots the machine. The advantage of sending switch orders through job scheduler is that job scheduler can automatically locate free nodes, and all the running jobs can be protected from other accidental operations.

The PBS batch job, which is a BASH script as shown in Figure 4, books one full node (with 4 cores), changes the default boot OS, and reboot. The command `sleep 10` is to avoid rebooting action interrupting the OS changing action.

```
1
2  ####################################
3  ### Job Submission Script         ###
4  # Change items in section 1         #
5  # to suit your job needs            #
6  ####################################
7  # Section 1: User Parameters        #
8  ####################################
9  #
10 #!/bin/bash
11 #PBS -l nodes=1:ppn=4
12 #PBS -N release_1_node
13 #PBS -q default
14 #PBS -j oe
15 #PBS -o reboot_log.out
16 #PBS -r n
17 #
18 ####################################
19 # Section 3: Executing Commands     #
20 ####################################
21 echo \$PBS_JOBID >>/home/sliang/
       reboot_log/rebootjob.log #write logs
22 sudo /boot/swap/bootcontrol.pl /boot/swap
       /controlmenu.lst windows #changes
       default boot OS
23 sudo reboot #reboot node
24 sleep 10 #leave 10 seconds to avoid job
       be finished before reboot
```

Figure 4. An example of PBS job An OS switch job in torque

3) Daemon programs for queue monitoring : The key to make the dual-boot cluster switch idle resources automatically, are the daemon (background) programs. Two daemon programs are running at each head node, which are designed to determine the job queue state, judge the system switching actions and send the switching batch job.

In the OSCAR head node, PBS does not provide APIs (Application Programming Interface) for other programs. Several Perl programs had been written for parsing the output of PBS commands and submitting OS switching job. A C++ program was written for TCP/IP communication with Windows HPC 2008 R2 head node.

In the Windows HPC 2008 R2 head node, Microsoft provides a SDK (Software Development Kit) for programs to fetch the data and send the tasks, e.g. get the queue state and nodes state. In the *dualboot-oscar* v1.0, two programs are made for fetching queue state and communicating with OSCAR head node. To reduce the difficulty of programming, the communicating program is compiled in the Cygwin environment.

4) Queue state fetching programs (detector): To define the queue state, we define a scheduler is "stuck", when the scheduler has no job running and several jobs are queuing. The detector reads how many compute nodes the first queuing job needs.

The detector gives a text output of scheduler, by parsing the PBS command `pbsnodes` (full detail of nodes in PBS is shown in Figure 7) and `qstat -f` (for full detail queue status, see Figure 8). The first line is the information for the communicator, others are debug information. The format is explained in Figure 5, which is a character sting sent through network. The detector's outputs in the queue state of running, stuck and others is shown in Figure 5.

Position	Definition	Output
0	[Queue state]	Stuck=1
		Others=0
1-4	[Needed CPUs]	Default=0000
5-67	[Stuck job ID]	Default=none
68-	[Undefined]	

Figure 5. Output format of detectors

In Windows HPC 2008 R2 head node, the detector fetches data through the API it provided, and follows the same output format as in figure 5

```
1  [sliang@eridani pbs]$ /dualboot/
       checkqueue.pl
2  00000none
3  Other state
4  R=0 nR=0
5
6  [sliang@eridani pbs]$ /dualboot/
       checkqueue.pl
7  00000none
8  Job running, no queuing.
9  R=1 nR=0
10 1186.eridani.qgg.hud.ac.uk
11         Job_Name=sleep
12         Job_Ownner=sliang@eridani.qgg.hud
               .ac.uk
13         state=R
14         time=2010 04 17 20 11 12
15
16 [sliang@eridani pbs]$ /dualboot/
       checkqueue.pl
17 100041191.eridani.qgg.hud.ac.uk
18 Queue stuck
19 R=0 nR=1
```

Figure 6. Three kinds of outputs of PBS detector [4]

C. Dual-boot deployment

1) Modifying OSCAR deployment tool: The deployment of initial *dualboot-oscar* requires several manual changes in the deployment script generated by OSCAR. It has to be redone each time administrator rebuilds the node image, which brings huge inconvenience in system managing. The changes include inserting FAT partition, clearing the initial

```
enode01.eridani.qgg.hud.ac.uk
        state = free
        np = 4
        properties = all
        ntype = cluster
        status = opsys=linux, uname=Linux
             enode01.eridani.qgg.hud.ac.uk
             2.6.18  1 6 4 .el5 #1 SMP Fri
            Sep 9 03:28:30 EDT 2011 x86_64
            ,sessions=? 0,nsessions=? 0,
            nusers=0, idletime=257163,
            totmem=15881584kb, availmem
            =15825740kb, physmem=8069096kb
            , ncpus=4, loadave=0.00,
            netload=154924801596, state=
            free,jobs=? 0,rectime
            =1271497128
```

Figure 7. An example of `pbsnodes` output

```
Job Id: 1185.eridani.qgg.hud.ac.uk
        Job_Name = release_1_node
        Job_Owner = sliang@eridani.qgg.
            hud.ac.uk
        job_state = R
        queue = default
        server = eridani.qgg.hud.ac.uk
                 elease_1_node.e1185
        exec_host = node16.eridani.qgg.
            hud.ac.uk/3+node16.eridani.qgg
            .hud.ac.uk/2+node16.eridani.
            qgg.hud.ac.uk/1+node16.eridani
            .qgg.hud.ac.uk/0
        Priority = 0
        qtime = Fri Apr 16 17:55:40 2010
        Resource_List.nodes = 1:ppn=4
        Variable_List = PBS_O_HOME=/home/
            sliang,PBS_O_LANG=en_US.UTF-8,
                PBS_O_PATH=/usr/kerberos/
                    bin:/usr/local/bin:/
                    usr/bin:/bin:/usr/
                    X11R6/
```

Figure 8. An example of `qstat -f` output [4]

section of the disk for Windows installing, and adding files into the node image.

The standard node image is a pure EXT3 format located in local disk of head node. To add a FAT partition and an empty partition for Windows, the disk layout file `ide.disk` and the deployment script it generated; `oscarimage.master`, needs to be manually modified.

The main points to be edited are:

1) Reserved space in `ide.disk` by adding partitions of Windows and dual-boot FAT partition.
2) In `oscarimage.master`, replace `mkpart` by `mkpartfs`, to make FAT works proper.
3) Add `modify-window=1 size-only` argument to `rsync` commands, to support syncing FAT format partitions.
4) remove the lines of Windows partition in `fstab` and unmount commands to avoid errors.

2) Patching Windows HPC deployment tool : Windows HPC has stored its configure file in a clear-text file, which is `C:\Program Files\Microsoft HPC Pack 2008 R2\Data\InstallShare\Config\diskpart.txt` shown in Figure 9.

```
select disk 0
clean
create partition primary
assign letter=c
format FS=NTFS LABEL="Node" QUICK
    OVERRIDE
active
exit
```

Figure 9. original `diskpart.txt`

Because we already know our disk size, the modified version only uses a part of the disk. In our case, it is a 250GB hard disk, so we reserved 150GB for Windows. The modified version is shown in Figure 10.

```
select disk 0
clean
create partition primary size=150000
assign letter=c
format FS=NTFS LABEL="Node" QUICK
    OVERRIDE
active
exit
```

Figure 10. modified `diskpart.txt` in *dualboot-oscar* 1.0

Because this `diskpart.txt` script wipes out the whole disk, the Windows partition has to be installed first, and each time during reinstallation of Windows, Linux needs to be reinstalled as well.

Our initial dual-boot systems performed well in production trials, and it provided the dual-boot function between Windows and Linux dynamically. We have evaluated the time spends in booting from one OS to another takes no more than five minuets. Keeping two job schedulers and both Windows and Linux server in bi-stable mode gives flexibility and speed-up, compared with other one-Linux-schedular hybrid cluster in mono-stable mode. [5]

However, as evident from the above descriptions in deploying and managing, there were number of system limitations that needed to be overcome.

This system requires a substantial input from the administrators, that is not always practical and is time and labour consuming in the process of reinstallation and reconfiguration.

IV. Improved Easy-to-Deploy Multi-boot Cluster

The improved easy-to-deploy multi-boot systems was designed to overcome the shortcomings of the initial system. The new version can be divided into Controller and Deployment similarly to the initial version.

Figure 11 shows a flowchart of the Controller of *dualboot-oscar v2.0*. A control process begins with fetching a queue state on the Windows head node. This state will be forwarded to another daemon program on Linux head node. Once Windows scheduler state arrives, the Linux daemon fetches the state of Linux scheduler. With this information, the Linux daemon can make a decision if switching is required, and which operating system (OS) to be switched to, as well as how many node to be switched. The reboot order will be sent to the Windows or Linux daemon if one of them have been identified as resources required. Each daemon sends reboot batch jobs to its scheduler. The control process terminates when all reboot jobs are run.

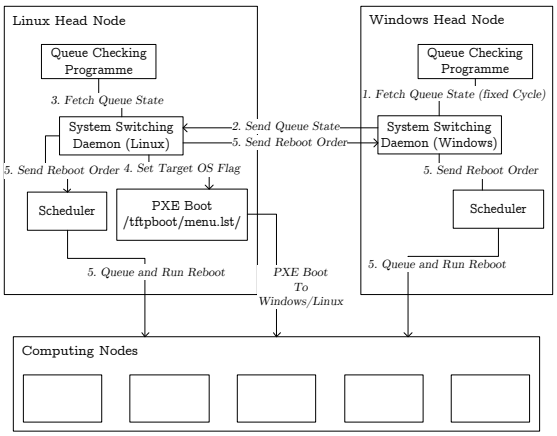

Figure 11. Flow chart of dual-boot OSCAR 2.0

A. Dual-boot Controller in Version 2

Because the local GRUB bootloader has to be correctly pointed by MBR section on a hard disk, the reimaging of Windows partitions always rewrites MBR and damages GRUB which boots Linux. Therefore, Windows has to be installed before Linux. This is a considerable inconvenience during the system maintenance. *dualboot-oscar* v2.0 only places the PXE (Preboot Execution Environment) network bootloader ROM in the head node. Thus, the MBR information in each computer node does not have to be fixed after either systems reimaging.

1) PXE boot ROM: PXELINUX is sub-project of SYSLINUX, whose aim of design is to help Linux booting from different storage formats, e.g. CD-ROM, PXE and FAT. PXELINUX could load our machines into PXE environment, and it is also the method that OSCAR uses to deploy compute nodes. However, PXELINUX has less ability in controlling local partitions booting. It only can quit PXE and lead to normal boot order.

The solution to let PXELINUX control compute nodes boot order, is to load a ROM to PXE first, such as PXEGRUB, then let PXEGRUB load remote node.

Initially, the ROM of PXEGRUB in GRUB 0.97 was chosen for the network bootloader.

PXEGRUB is an optional component of GNU GRUB. It could be obtained by compiling from source code with parameter `--enable-diskless` and `--enable-(suited NIC drivers)`. DHCP and TFTP services could specify individual boot ROM and configure file for each node.

The PXEGRUB also makes system switching simpler. The *dualboot-oscar* v1.0s method uses a FAT partition which stores configure file. As their configure files are stored at the head node, a compute node could be switched by any reboot action, including soft reboot and physically power reset. This is an improvement to the initial system.

A lot of tests have been done in virtual machine on a newer workstation, which does support virtualisation. The tests proved the practicality of this new method in the virtualised environment. Due to the discontinued development of GRUB 0.97, new models of LAN cards are not supported. Therefore, we needed to change our approach.

GRUB4DOS is an open-source fork of GRUB. GRUB4DOS supports a wider range of disk formats and load methods than GRUB. It also has a very easy-to-obtain PXE ROM.

The PXE ROM of GRUB4DOS reads different menu files, which are located in the directory of `menu.lst/` under the PXE directory (normally is `/tftpboot/`), named from compute nodes' LAN cards MAC address.

By modifying the menu files, *dualboot-oscar* can control any machine's boot order, as long as they are connected to their head node.

Initially, the OS loading method is designed to make `menu.lst` for each machine's MAC, then it could boot specific machines to specific operating system. However, the daemon program in OSCAR head node would not easily get information about which machine is scheduled to be rebooted. The flowchart of this approach is shown in Figure 12.

Eventually, the method is developed into a single "flag" control system. All the rebooting nodes will be led to the same operating system, because the whole dual-boot cluster will only need one system at one time. As in Figure 13, the current approach is more concise.

2) Fetching queue states: The former way to get the queue information can be kept. The output of queue state checking programs could be collected and used by new

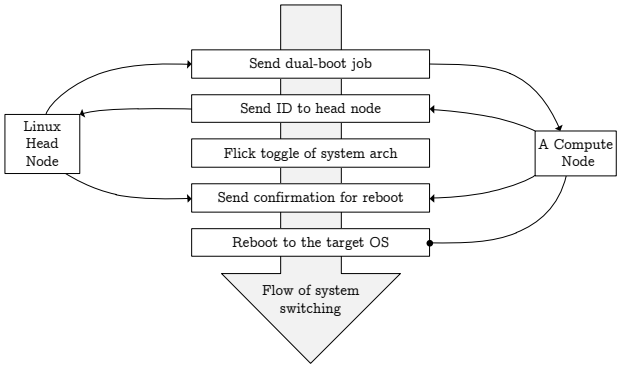

Figure 12. Initial approach of PXE OS Boot Control of version 2

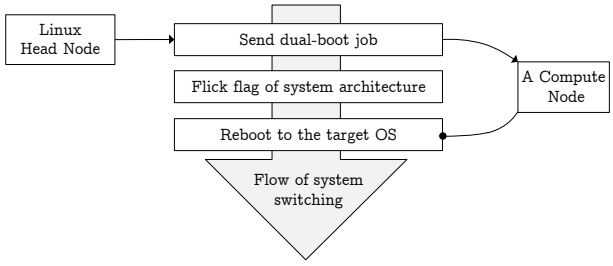

Figure 13. Current way of PXE OS boot control of v2.0

communicator programs.

3) Head node communicators between Windows head and OSCAR head: The two communicators are written in Perl. Strawberry Perl or Active Perl is required in Windows Head node. Its flow chart is in Figure 11. Windows queue status is submitted to Linux side by TCP/IP socket communication. Multi-boot service sends switch batch job (just reboot).

1) Windows communicator fetches queue state in fixed cycles (intervals), e.g. 10mins.
2) Windows communicator sends queue state to Linux communicator.
3) Linux communicator fetches PBS queue state and decides how many machines are needed to switch OS.
4) Set target OS flag.
5) Send reboot order to Windows HPC scheduler or PBS scheduler.
6) Machine will be rebooted when it is totally free, then loaded into another OS.

B. Deployment of Version 2

A significant improvement is that the deployment of v2.0 has become better integrated than v1.0. After patching several supporting packages, OSCAR's dual-boot deploying scripts can be generated automatically each time. Windows partition and OSCAR partition can be individually reimaged without corrupting each other. Since there is no requirement of creating FAT partition for dual-boot and fixing MBR, the deployment is greatly simplified.

1) Patching OSCAR deployment tool: OSCAR uses `systemimager`, `systeminstaller`, and `systemconfigurator` to build compute node image, install system packages on compute node, and configure the nodes which are installed by `systeminstaller`.

By patching `systemimager` and `systeminstaller`, a new disk format label `skip`, is enabled in OSCAR's disk image configure file, e.g. `ide.disk` shown in Figure fig:ide.disk. The first partition with label `skip` will be reserved for Windows.

```
1 /dev/sda1        16000  skip
2 /dev/sda2        100    ext3  /boot
     defaults bootable
3 /dev/sda5        512    swap
4 /dev/sda6        *      ext3  /
     defaults
5 /dev/shm         -      tmpfs /dev/shm
     defaults
6 nfs_oscar:/home  -      nfs   /home
     rw
```

Figure 14. `ide.disk` in v2.0

2) Patching Windows HPC deployment tool: Deployment for Windows HPC compute nodes in v2.0 is similar to v1.0. By modifying `diskpart.txt`, Windows HPC deploy tool will only make one primary partition with specified size using the script shown in Figure 10. In the case of a number of compute nodes needing reimaging, a different `diskpart.txt` file shown in Figure 15 can be used to replace the `diskpart.txt` in Figure 10. This script does not corrupt the Linux partition and only formats the Windows partition.

```
1 select disk 0
2 select partition 1
3 format FS=NTFS LABEL="Node" QUICK
     OVERRIDE
4 active
5 exit
```

Figure 15. `ide.disk` in v2.0 for reimaging

The *dualboot-oscar* was deployed on our cluster "Eridani". Our system was tested on an application requiring optimisation of Genetic Algorithms using the Distributed and Parallel MATLAB. To run calculations on a cluster, it was necessary to have Matlab Distributed Computing Server (MDCS). MATLAB and MDCS had been installed on a shared folder in the Windows head node of "Eridani". [6] The compute nodes, which this application used were

switched to Windows system by our *dualboot-oscar*. As load shifted between the two OS environment, the system seamlessly adjusted.

This case study confirmed that our *dualboot-oscar* provided better utilisation of "Eridani" and supported both Linux and Windows applications.

V. Conclusions and Future work

In this paper, we presented a new design and implementation of dual-boot cluster middleware – *dualboot-oscar*. We have identified the need for flexible multi-platform systems in High-Performance Computing, evaluated current solution in HPC hybrid resources management. As a result of our investigation of commercial and open-source software for multi-platform systems, we have developed a dual-boot cluster system with OSCAR and Windows HPC, which is suitable for virtualisation-incompatible systems.

Version 1.0 of *dualboot-oscar* had been fully deployed on the cluster "Eridani" in the QGG. The principle of version 2.0 is stable and has been tested on "Eridani". Version 2.0 preserves the performance advantages from version 1.0 and has achieved the improvement in the system maintenance and reduction of manual modification and installation in system setup. This dual-boot system benefits both users and administrators in our QGG campus grid. It is a flexible, multi-purpose cluster tool that enables better utilisation of our cluster resources. The daemon (background) programs were devised to enable dynamic switching of cluster nodes between Windows or Linux depending on users demand. Currently the daemons for queue monitoring are still following the rule "first-come first-serve". This could be improved to adapt the rules from diverse administration requirements.

A further version of this dual-boot package is being considered as either a package of tools and software patches for some particular version of OSCAR or a dual-boot enabled fork of OSCAR project. This project could be hosted online as an open-source project to be carried on by anyone who wants to contribute to it.

VI. acknowledgements

The authors would like to acknowledge the use of the University of Huddersfield computational grid known as the Queensgate Grid in carrying out this work.

References

[1] TOP500.Org, "TOP500 Supercomputing Sites," 2011. [Online]. Available: http://www.top500.org/

[2] V. Holmes and I. Kureshi, "Huddersfield university campus grid: Qgg of oscar clusters," *Journal of Physics: conference series*, vol. 256, no. 1, p. 012022, December 2010. [Online]. Available: http://eprints.hud.ac.uk/9896/

[3] M. Carter, "Automate OS switching on a dual-boot Linux system," Mar. 2006. [Online]. Available: http://www.ibm.com/developerworks/linux/library/l-osswitch/

[4] S. Liang, "A Dynamic OS Switching Solution for Dual-boot Clusters," Master's Thesis, 2010.

[5] I. Kureshi, V. Holmes, and S. Liang, "Hybrid HPC – Establishing a Bi-Stable Dual Boot Cluster for Linux with OSCAR middleware and Windows HPC 2008 R2," in *AHM2010*, Cardiff, 2008. [Online]. Available: http://www.allhands.org.uk/2010/sites/default/files/2010/TuesW2KureshiHybridPC.pdf

[6] D. Haupt, "Genetic algorithms parallelization," Master's Thesis, Brno University Of Technology, Brno, 2011.