MSc By Research

Implementation of A Condor Pool At The University Of Huddersfield That Conforms To A Green IT Policy

Author:
David Gubb

Supervisor:
Dr. Violeta Holmes

January 31, 2013

Abstract

The University of Huddersfield has a large number of computers laboratories on campus that are used to capacity during timetabled session but out of these sessions the machine can be unused for long periods of time. Whilst these machines have power management software installed to reduce the universities electricity bill. The idle machines could be used to perform complex calculations and simulations to benefit the research community through cycles stealing techniques and High Throughput Computing (HTC) middlewares.

In order to provide a suitable HTC service an investigation was undertaken into what middlewares are available and how they compared against each other. This study also looked at what is green IT and how the chosen HTC middleware has been adapted to conform. The investigation also involved looking into publication from other universities to see how it is used.

A survey was conducted into how useful a Condor HTC grid could fit in with other universities High Performance Computing (HPC) clusters and how beneficial Condor is. The survey also looked at how Condor is funded and how it is administered. Overall the results show that Condor is an extremely useful and low cost HTC solution.

Condor has been deployed within Canalside East within the University of Huddersfield as a test bed with plans to expand the Condor pool across campus. To help Condor fit within the green IT policy the compute nodes were configured to allow the machines to go into a low power state when required.

To be able to prevent the possibility of having a large job queue with very few nodes online a number of scripts were created that would collect the information required to remotely wake machines up using Wake on LAN (WoL). The scripts will wake machine when a number of jobs are idle and there are machine available that are offline.

In order to make Condor able to run programs that have been developed for Windows and Linux a dual Condor client system has be implemented. This has been achieved by using the standard Windows client and a virtualized Linux client with Condor on called Pools of Virtual Boxes (PoVB). These clients run as a Windows service that can be remotely switched on and off when required remotely within the same script that can wake machines when required.

Acknowledgements

I would like to express my gratitude to my supervisor Dr Violeta Holmes for her continued support and encouragement optimism.

I am truly indebted and thankful to my collages and friends Ibad Kureshi, Yvonne James and Shuo Liang. They have provided me with advice, technical help, support and for just been there provide prospective when all was lost.

Finally I cannot put into words how grateful I am for my parents help and support throughout my Master project because without them this could not have been possible.

Contents

List Of Abbreviations **6**

1 Introduction **7**
1.1 Introduction 7
1.2 Queens Gate Grid 7
1.3 Aims 9
1.4 Objectives 9
1.5 Summary of Chapters 9

2 Literature Review **11**
2.1 What is Green IT and How it is used? 11
2.1.1 Green IT 11
2.1.2 Green Data Centres 12
2.2 High Throughput Computing 13
2.3 High Throughput Parallel Computing 13
2.4 HTC Middlewares 14
2.4.1 Windows HPC (HTC options) 14
2.4.2 BOINC 14
2.4.3 Condor 15
2.5 Which Middleware? 16
2.6 How Other Universities Use Condor? 16
2.6.1 Liverpool University 16
2.6.2 Newcastle University 18
2.6.3 Reading University 18
2.6.4 Bristol University 19
2.6.5 Oxford University 19
2.6.6 Cambridge University 20
2.6.7 Cardiff University 20

3 Campus Grids Special Intrest Group Condor Survey Results 22
3.1 Survey methodology 22
3.2 Survey Results 23
3.2.1 Condor Pool Information 23
3.2.2 Alternative Computing Resources 25
3.2.3 Payment Schemes 26
3.2.4 Users of the Condor Pool 27
3.2.5 Support and Encouraging New Users 27
3.2.6 Monitoring Tools 29
3.2.7 Condor Pool Configuration 29
3.2.8 Institutional Views 31
3.3 General Comments About Condor 32

4 Condor Deployment at the University of Huddersfield 33
4.1 Available resources 33
4.2 University of Huddersfield Deployment 35
4.3 Configuration of QGGCondor 36
4.4 Executable Machine Configuration 38
4.4.1 Windows Condor Execute Machine Installation . . . 39
4.4.2 Pools of Virtual Boxes 39
4.4.3 Pools of Virtual Boxes Configuration 40
4.5 Automation of IP and MAC address collection for 41
4.6 Power Management 42
4.6.1 The University of Liverpool script 42
4.6.2 Extraction of Information 43
4.6.3 Controlling Services Locally 44
4.6.4 Controlling Services Remotely 44
4.6.5 Final Power Management Script With Service Control 45
4.7 High Throughput Parallel Computing Implementation 46
4.8 Creating an MSI 46

5 Conclusion 49
5.1 Further Work 50

Bibliography 51

A Modifications made to Windows condor_config 55

B [mac.pl] Automation of MAC and IP Collection 56

C Condor wake on script from Liverpool 58

D Source code condor power v1 **60**

E HTPC Settings **64**

F Final Script **67**

List of Figures

3.1 Number of Cores in Condor Pool 23
3.2 Type Operating Systems Condor Pool Operates On 24
3.3 Who Administers The Condor Pools 25
3.4 How Many Institutions Have a Payment Scheme For 26
3.5 How Condor Pool Are Advertised 28
3.6 Condor_View [1] 29
3.7 Network Speed for Condor Pools 30
3.8 Power Managment 31

4.1 QGG Utilisation for 2012 34
4.2 System layout 35
4.3 Changes made to *condor_config* 37
4.4 Changes made to *condor_config.local* 37
4.5 Changes made to */etc/profile* 38
4.6 *condor_status* output 38
4.7 *condor_q* output 38
4.8 Windows Power Saving Settings Made To *C:\condor_config* 39
4.9 PoVB Hard Drives[2] 40
4.10 Changes made to *condor_congig.policy* 40
4.11 Changes made to *povb_config* 41
4.12 *WOLinfo.txt* Output 42
4.13 Condor Power Script Version 1 43
4.14 Commands To Get The Hostnames To Compare 46
4.15 Installation of Condor using Batch Script[3] 47
4.16 Installation of Condor using Batch Script 48

List Of Abbreviations

BLCR Berkley Lab Checkpoint/Restart.

CALs Client Access Licenses.

CG-SIG Campus Grids Special Intrest Group.

HPC High Performance Computing.

HTC High Throughput Computing.

HTPC High Throughput Parallel Computing.

MRF Midnight Render Farm.

NAT Network Address Translation.

NGS National Grid Service.

NW-GRID North West Grid.

PoVB Pools of Virtual Boxes.

QGG Queens Gate Grid.

SETI Search for Extra Terrestrial Intelligence.

SLA Service Level Agreement.

WoL Wake on LAN.

Chapter 1

Introduction

1.1 Introduction

The University of Huddersfield has a large number of computer labs that are used to capacity during the courses delivery. Out of scheduled teaching and learning sessions the computer labs are not used at full capacity.

After a set period of time the power saving software that is installed (Energy Star EZ GPO)[4] on the laboratory computers recognises that the computer has become idle and it will force the computer to go into hibernation. When a user comes to use the machine that is hibernating, the user can easily and quickly wake the machine up by either moving the mouse or by pressing a key on the keyboard.

While having computer laboratories go into hibernation saves money on the University electricity bill. These idle computers could be used to benefit the research community by performing complex calculations or simulations. This can be achieved by using cycle stealing techniques used with High Throughput Computing (HTC) such as Condor[5].

1.2 Queens Gate Grid

Adding a HTC resource would be extremely valuable to the University of Huddersfield super computing grid the Queens Gate Grid (QGG).

QGG was established as part of a MSc project that evaluated the requirements of the research community at the university and the need to have a High Performance Computing (HPC) resource that can benefit the research community[6]. The research community itself has expanded further with the newly formed particle physics research group as well as other

researchers who have found they need to use HPC to further their research.

Originally the QGG had just the Eridani cluster, which is a 152 core beowulf Linux cluster, and the Tauceti cluster, inherited from the School of Applied Sciences which is a 30 Core AMD Cluster. These clusters then used a 16TB network storage device named Mimosa.

Now the HPC resources of the QGG still include the 152 core Intel Beowulf cluster but Tauceti is now the test platform for the administrators of the QGG and has been broken down to 16 Core cluster with the other nodes have been made available for student projects. The new jewel of the QGG is the 256 core AMD Sun cluster called SOL that is now hosted in the university data centre. The cluster is hosted in the data centre because it is connected directly to the university networking back bone so that users can connect and move data as fast as possible with the lowest amount of latency to the users. Another advantage of SOL been installed in the university data centre is that the cluster is powered directly from a UPS so it can operate 24 hours without external power and in case of power failure, there is a generator back up so the cluster can continue to operate.

The storage has been upgraded to a 24TB mirrored Gluster storage devices that allows us to have a back up of all the user data that has been synchronised. Gluster system itself automatically synchronises the data as it is changed across the network. The mirrored devices are put into the two of the QGG data centres. This allows each cluster to retrieve data across the network with as low a network latency as possible.

The QGG has also expanded into GP-GPU technology which uses specially designed graphics cards to perform parallel calculations. This system is the Vega cluster which at the moment uses 2 Nvidia M2050 GP-GPU cards. The Vega cluster is also the only Windows HPC the QGG has.

Since the writing of this thesis the computing resources have grown to have an offsite 216 core Linux cluster in an agreement with Daresbury laboratory[7].

Midnight Render Farm (MRF) which is used to render image or videos. This uses Autodesks Back Burner[8] software in one of the computer laboratories when it is closed on an evening.

The QGG has a wide range of users, such as Mechanical engineers, Chemists, Physicists, biologists, Software engineers and Graphics rendering.

1.3 Aims

The aim of this project is to implement a HTC computational grid that conforms to a green I.T. policy which allows long running serial calculation. This grid must be integrated into QGG so that existing users can log on and are able to use it to submit jobs to the HTC grid. This grid also need to be able to support as many different applications so that it could potentially run Windows or Linux jobs depending on the suitability.

1.4 Objectives

- Perform a literature review into the relevant technologies available and the guidelines available for green IT
- Analyse other universities Condor deployments
- To perform a survey to gain the opinion of other institutions regarding Condor and how it fits into the institutions HPC resources
- Create a script that works with Condor that enable control of the power state of the laboratory computers
- Create an executable that enables an easier deployment of Condor within the universities computer image
- Refine Condor so that it can conform to the universities Green IT policy
- Implement Condor which conforms to the green I.T. policy

1.5 Summary of Chapters

Chapter 1 (Introduction) Discusses why HTC computational grid may be useful to the University. The aims and objectives are also outlined.

Chapter 2 (Literature Review) In depth review of Green IT, the available HTC middlewares, how Condor is used and deployed at other institutions.

Chapter 3 (Condor Survey) This will present and discuss the results of a survey conducted to find out how Condor fits into the respondents HPC resources.

Chapter 4 (Condor Deployment) The actual deployment of Condor for the University of Huddersfield based on the research performed previously in the previous chapters.

Chapter 5 (Conclusions and Further Work) Conclusions are made on the implentation of Condor at the University of Huddersfield with any work that could be added to the project to improve it.

Chapter 2

Literature Review

This section will cover what is green IT, what is HTC, what HTC middlewares are available and how do other universities use Condor.

2.1 What is Green IT and How it is used?

In order to make a HTC computational grid that conforms to a green IT policy, green IT has to be defined and how it can be used within an IT infrastructure.

2.1.1 Green IT

Green IT first appeared in 1992 as part of a joint project between the U.S Environmental Protection Agency and the U.S Department of Energy who formed ENERGY STAR. The aim for ENERGY STAR was to help save money and protect the environment through energy efficient products and practices.[[9]

ENERGY STAR initially introduced a voluntary labeling system to identify products are energy-efficient and can reduce the production of greenhouse gases[9].

Green IT for most companies is about reducing the amount of time that a computer is using power and the amount of time a computer sits idle thus saving power and money. In this project the aim is to introduce a HTC grid that uses the computers to their potential without adding anymore time where the computer may be idle.

2.1.2 Green Data Centres

In most data centres the majority of the running costs are from cooling the servers and from powering the servers. Reducing these two aspects can make the data centres greener whilst reducing overall running costs.

The easiest way to reduce the amount of power used and the heat created is by using the servers built in power saving features and CPU throttling. For example by enabling the power saving features on an AMD Opteron could reduce the power consumption by up to 65% if the CPU utilization is less than 50% [10]. This option requires no extra investment and it will have very little impact on the performance of the server.

The best way to reduce power consumption is to replace old service with newer more energy efficient servers. Another advantage in having newer servers is that they will perform better than the old ones because there can be more CPU cores within the server. The University of California at San Diego did this and saved 6.5KW with the same computing power"[11]. By upgrading the servers reduced power consumption by 81.25%, which will reduce the cost of running the servers as well as reducing the amount of cooling required for the servers.

As the number of cores increase per CPU it has become more economical to combine servers that may have had one service running to consolidate into fewer servers. This is achieved by virtualizing these services into individual virtual machines that can share the computing resources of these powerful servers. By migrating the service into virtual machines it keeps the reliability of having multiple machines running with a lower power requirement and footprint[11]. This is achievable because quite a few services require low amounts of CPU utilization for long periods of time making it more cost effective[11].

The ideal data centre belongs to AISO.net who run their data centre completely from solar power[12]. This is achieved by using 120 solar panels that produce 12KW of power that runs the servers and charges batteries for use on rainy days[12]. They have invested into VMware and new hardware so that they could consolidate 120 servers into 4![12] Unfortunately this is not an option in the U.K. even though we might want to.

The standard temperature the air-conditioning units in data centres has changed from been 13°C to 27°C[13]. This has been found to be the perfect balance between keeping the servers cool enough without the fan throttling too high which consumes a larger amount of power[13]. As a result the air conditioning units uses less energy to cool the data centre[13].

2.2 High Throughput Computing

HTC systems concentrate on how many compute jobs that can be performed over large periods time such as weeks, months and even years rather than with HPC systems that concentrate on how many calculations per second[14].

The hardware for HTC can be dedicated hardware such as servers or even some computer clusters although it usually makes more financial sense to use HTC middlewares that allow for cycle stealing techniques. The ability to cycle steal allows for a better use of existing hardware. Using existing hardware in an institution such as a University where there are large amounts of computing hardware that can remain idle for hours.

HTC systems excels at running embarrassingly parallel tasks as well as long running serial jobs. Embarrassingly parallel task is when each task is so independent that it can be performed without needing any of the other tasks. Most parallel application have parts that are dependant on others tasks to be done before all tasks can be complete. This can be taken into account using Condor DAGMAN scripts[15] that allows for dependence to be defined but this is not as easy implement with closed source applications or precompiled MPI applications.

2.3 High Throughput Parallel Computing

High Throughput Parallel Computing (HTPC) takes the advantage of been able to use machines that already configured to run HTC tasks but adds the ability to run parallel jobs on the machine itself[16]. This has become possible because CPUs can have multiple cores on a single processor and the amount of RAM a machine can use has increased.

While users could argue that they could perform these small parallel jobs on their own machines. The problem with this idea is if they have a large number of parallel jobs to run then they loose the use of the machine until complete which could be days or weeks.

The user could run these parallel tasks on a HPC cluster but if their parallel job can fit onto a desktop machine then it would be a waste of a node that could be used for larger parallel tasks. If there are a large number of small parallel job to run, this could create a long backlog of jobs that would disadvantage other users of the cluster. In many universities there are more desktop machines than HPC dedicated machines available for running HTPC jobs. Therefor this resource will enable the large number of parallel jobs to run quicker.

2.4 HTC Middlewares

In order to develop a suitable HTC grid for the QGG, an analysis of the different middlewares need to be evaluated to determine if there are any better alternatives to Condor.

2.4.1 Windows HPC (HTC options)

Windows HPC Server 2008 R2 is primarily used to create and manage Windows HPC Clusters. Windows HPC Server 2008 R2 can also work as a HTC middleware[17] using workstations and dedicated servers.

The workstations can be configured to cycle steal which can monitor the CPU usage, keyboard activity and mouse usage to determine if a use has come back to the computer[17].

Workstations can be scheduled to become a dedicated HTC node between certain times of the day, such as out of office hours and bank holidays. If additional resources are required then the administrators from the head node can manually switch on any machines that have been switched off. Windows HPC Server 2008 R2 can only run software developed for a Windows environment[17].

Windows HPC Server 2008 R2 has a high start up cost associated with deployment because each compute node requires its own licence for Windows 7, Client Access Licenses (CALs) and HPC Server 2008 R2. For Organisations with an existing pool of computers that run Windows 7 can easily start cycle stealing with the majority of the cost been the HPC Server 2008 R2 Licence and the CALs[17].

2.4.2 BOINC

BOINC is a piece of software designed to sit on a computer and cycle steal which is developed at the University of California [18].BOINC will allocate jobs to machines that have been installed with BONIC and is registered to a specific project[19].

One of the biggest BOINC projects is the SETI@home whose aim is to Search for Extra Terrestrial Intelligence (SETI). SETI works by sending a small sample of data from a radio telescope which is 0.25MB[20] in size. The data is then processed on the compute node by a program called splitter [20]. Once complete the results are sent back to the SETI servers. Each piece of data is processed three times to verify if there have been any errors in processing the data [21].

The advantage of running BOINC for types of project like SETI is that people turn processing the data into competitions because each processed data set is allocated a certain number of points. This is one of the ways that the projects entice people into performing their calculations for free.

An example of how powerful BOINC could be if it was all in one data centre is that the SETI project has 3,281,277 hosts registered to process data. If each host had only 1 processing core then SETI has almost double to processing cores than the top supercomputer in the world Sequoia with 1572864 cores [22]. Admittedly Sequoia would consume less power and each core will most likely be more powerful as well as having access to more RAM.

2.4.3 Condor

Condor is an open source HTC middleware developed by the University of Wisconsin-Madison in 1986[23]. Condor is a grid middleware that allows for computing jobs to be run on distributed computing resources. Condor is capable of running parallel jobs as well as serial jobs[24]. The great thing about Condor is that it is compatible with Mac OSX, Linux and Windows meaning that no matter what hardware or operating system is available to use Condor can be configured to run[24].

Condor is highly effective at cycle stealing from desktop machines. This is achieved by monitoring the machine CPU, keyboard and mouse activity to determine if it is idle. Condor can also be configured to use dedicated machines which is the best way to make a computer cluster and run parallel applications[24].

When code is compiled using *condor_compile* the code can stop on that machine and start up again on another available machine from the last time it save its progress, this is called checkpointing[24]. Checkpoint is only available when running Condor on Linux or Mac OSX. Condor can be configured when a computer changes into an owner state to either stop running the job and keep the data in its RAM or to drop the job completely and start it again on another available machine[24]. Whilst keeping the job in the RAM so the job doesnt have to start again would be convenient to the person running the job it could severely interfere with the user experience. The user would probably restart the machine that would force the job to start again on another machine defeating the whole purpose.

The data for the jobs can be transferred using Condors own data transfer mechanisms so that network drives dont have to mounted on every single computer.[24].

Condor can be used to submit jobs to other systems using Condor-G, which can submit to Globus job managers[24]. Collaboration can be achieved with other Condor pools by using a mechanism called Flocking which will run the job on another machine in the linked pool when the current pool has too many jobs are queued[24].

2.5 Which Middleware?

Condor will be used as the HTC middleware because it can be used on almost any operating system. A Linux head node for Condor con allocate jobs to any nodes regardless of the compute nodes operating system which allows it to be integrated with the existing QGG infrastructure.

Condor is also used by a large number of universities around world so it must be effective and easily configurable to work in different IT infrastructures.

Condor is open source, which is useful if any modifications need to be made and its free to own and develop making it more cost effective. The Condor project is actively been developed with a new updated version is released nearly every month that can add new features as well as fixing bugs and security issues.

2.6 How Other Universities Use Condor?

2.6.1 Liverpool University

At the time of the article (2009) the Condor compute nodes were Intel Core 2 Duo machines with 2GB of RAM and they were running Windows XP [25].

The Condor pool had around 300 laboratory machines to perform calculations but there are around 2000 computers at the University of Liverpool campus. They chose to only use the most powerful computers. This was done because the most powerful machines will be newer more energy efficient machines as well as been able to complete the jobs quicker. The main problem with this approach is if the Condor queue is extremely saturated then having a few more slots would help clear the queue quicker. [25].

Condor has been configured so that it will drop a job when a user comes to use the machine and then it will wait 10 minutes before it powers

down. Each machine has been configured to use a job slot for each core (2 job slots) to maximise the amount of jobs that can run at a time [25].

The Condor head node is configured as a combined job manager and scheduler as well as been able to submit jobs via Condor-G. They monitor the long term usage of Condor using Condor View [25].

Initially the computers we left on most of the time, which would waste a lot of energy. Then a policy came in to introduce power saving by sending the machines into standby. The problem with leaving a machine in standby is that it still uses a lot of the power because the system data is stored on the RAM of the machine that requires power to keep the data [25].

It was decided that putting the machine into hibernation instead of standby is a better idea because the system data is written temporarily to the hard disk. The advantage of doing this is that most of the system can power down only leaving enough to check for an input from a keyboard or mouse. It is also better for the user because a machine takes less time to power back up from hibernation than it does standby [25].

User of GAMESS and PC-GAMMESS can submit jobs through a web portal created by Liverpool to make it easier for the user to submit jobs[25].

Liverpools first attempt at power saving for Condor was to use a DOS batch script to check if a user was logged in but Condor itself does not run as a user. The machine would turn itself off even if a job was running[25].

The solution was to make the batch file look to see if a *condor_exec.bat* existed in the working directory for Condor. When Condor transfers an executable to an node it calls it *condor_exec* which stays there while the job runs. Sometime the files would stay there when the job had finished because Condor had not run its scheduled job *condor_preen* which is supposed to remove all the contents of the working directory when a job finishes. This resulted in machines staying on longer than required[25].

The final attempt uses Data Synergys PowerMan Software that monitors the power usage of computers and records the data. PowerMan has the feature to allow administrators to wake and hibernate machine when required. PowerMan can be configured to wait for a specific running process to finish before hibernating a machine[25].

All of the execute nodes have Wake on LAN (WoL) enabled so that the machines can be woken up on demand. These machines can be woken up when required by a Perl script that runs every 15 minutes using Crontabs on the head node. When required machines are woken up a computer laboratory at a time until sufficient machines are awake. This has proved to have an unexpected flaw of been very noisy to the point some students were unplugging machine to prevent it from happening [25].

Adjustments had to be made to the Condor configuration on the ex-

ecute machines when the power options were changed from hibernation after 30 minutes to hibernation after 15 minutes. This had to be done because if a machine was idle for 15 minutes then PowerMan would hibernate the machine even though it was been picked up as idle. Condor has been changed to wait for 5 minutes of no user activity before it will start to run jobs [25].

Liverpool have tried to implement the built in power saving policy for Condor but found it complicated and they had a perfectly good solution so they stuck to the home grown approach [25].

2.6.2 Newcastle University

Newcastle University has a Condor pool that uses idle Information System and Services computer laboratories. These machines run Windows 7 and they have Intel Core2Duo or quad core PC's with 8GB RAM. The Computers are restarted every day at 5 am so that any software updates can be installed[26].

These computers run on the policy of powering down after a certain amount of time of been idle. The Arjuna Agility Cloud Platform controls the waking of machines. This is used because it allows a Service Level Agreement (SLA) to be implemented and enforce within a Condor pool. These policies are programmed into the configurations of the Arjuna Cloud[27].

The cloud works by having two domains. The first one is the Client domain that represents the users privileges and the second is the Condor domain that represents the Condor head nodes. Depending on the users privileges users can specify if a job is a low priory so that Condor will only use machines that are online and it will not wake any machines up. If there are a large number of low priority jobs in the Condor queue then the cloud can make the decision to wake up some machines to clear the queue out[27].

2.6.3 Reading University

Reading University has a Condor pool of around 500 machines provided by the Computing Services and the School of Systems Engineering computer laboratories. These machines have various amounts of RAM that can be important in requesting a suitable machine to run a job[28]. These machines use a form virtualization called Cooperative Linux or more commonly known as CoLinux. [29]

Having a virtualized Linux Condor pool allows for checkpointing of jobs provided the code has been compiled with the Condor headers[29].

Each user of the grid has their home folders mounted automatically to the execute machines so that it doesn't need to use the Condor data transfer methods[29]. Each compute node has a 20GB temporary storage space allocated as well been able to use a mounted scratch space of up to 600GB[28]. This is a cost effective way of running jobs that have large data sets or large outputs the only problem is this introduces a bottleneck in the speed of data to be transferred.

The head node runs Centos 5.2 with Globus so that users from the National Grid Service (NGS) can submit jobs to the Reading Condor pool[29].

There is also negotiation with Oxford University so that users of the Reading University Condor Pool[29] could flock to the OXGRID.

Users can also use Berkley Lab Checkpoint/Restart (BLCR) which is on the login node which is a wrapper that a user uses *checkpoint_submit* command to allow code to use checkpointing without been compiled using the Condor Headers[28].

2.6.4 Bristol University

The University of Bristol has Condor installed on over 720+ machines across 6 departments within the university. Each department can flock jobs between the different pools when there are enough jobs sat idle.

By default Condor will drop the compute job when a user comes along and these are running Windows XP. Each compute node is at least 2GB of RAM and has a multicore Intel Processors.

They share their Condor pool on the NGS[30].

2.6.5 Oxford University

Oxford University is installed on the computer laboratories within 38 of the universities collages. Each collage/department has a head node to control the nodes connected. The Universities Computing Services hosts the Condor Master where users need to login to in order to submit jobs and this allows flocking between the different Condor pools[31].

The Condor pool supports Windows and Linux through virtualization using Co-Linux. This is controlled remotely by using Net RPC Commands to start and stop the Co-Linux Service. The Linux flavour used by Co-Linux is Debian.[31].

In order to easily deploy this setup a MSI installer is created that can deploy and configure the Linux and the Windows Condor services.[31].

Condor will run jobs when there is at least one processor free so that jobs can run in the background. The Windows Client controls the Co-Linux.[31].

Uses WoL to wake machines when nodes are required for the jobs across the university.[31]

2.6.6 Cambridge University

Cambridge's University Condor pools are setup and created by research groups and departments, which can then flock across different pools creating the CAMGRID. These machines are either desktop or servers. The universities Computing Services department coordinates the CAMGRID. Each group can decide on the overall cycle stealing policy that is implemented within their own pools[32].

Overall the CAMGRID is around 1150+ cores. The grid itself is designed to run large numbers of serial jobs and is also capable of running parallel jobs using the MPI universe on a single machine.

They have created a web GUI that allows users to their files anywhere on campus. There is also a GUI that creates DAGMAN Scripts called PyDAG. This is useful for staging jobs that may be dependent on other parts of the job. [33]

2.6.7 Cardiff University

The University of Cardiff has a Condor pool of around 1600 machines with a theoretical performance of 1 TFlops [34]. These machine operate within computer laboratories that run Windows XP[35].

The University provides a one on one service for developing bespoke application development to encourage users to use the Condor pool[35].

200 of the machines are available for use through the NGS[36].

To fully understand Condor deployment and management it was decided to conduct a survey across many universities in the UK. The results of this survey will be considered in the following chapter.

Chapter 3

Campus Grids Special Intrest Group Condor Survey Results

This survey was commissioned by the Campus Grids Special Intrest Group (CG-SIG) to determine how Condor is used and how it fits in alongside HPC resources to help improve computational research that is carried out.

A benefit of carrying out this survey that it also produces some primary information into how Condor is currently been used and if there are any key issues that may need to be considered later on.

3.1 Survey methodology

The survey was split into 8 major sections that would group the survey question into logical groups.

- Condor Pool Information
- Alternative Computing Resources
- Payment Schemes
- Users of the Condor Pool
- Support and Encouraging new users
- Monitoring Tools
- Condor Pool Configuration
- Institutional Views

The information from these sections could determine if owning a Condor pool for the University of Huddersfield would be a useful asset. The members of the GC-SIG will be able to create a business case for using Condor from this information. These business cases will help them to a case for expanding existing Condor Pools or even to establish a Condor Pools.

3.2 Survey Results

3.2.1 Condor Pool Information

This questions asked in this section of the survey was to determine the size, Operating System, who administers the Condor pool and if there is any charge for the use of the pool.

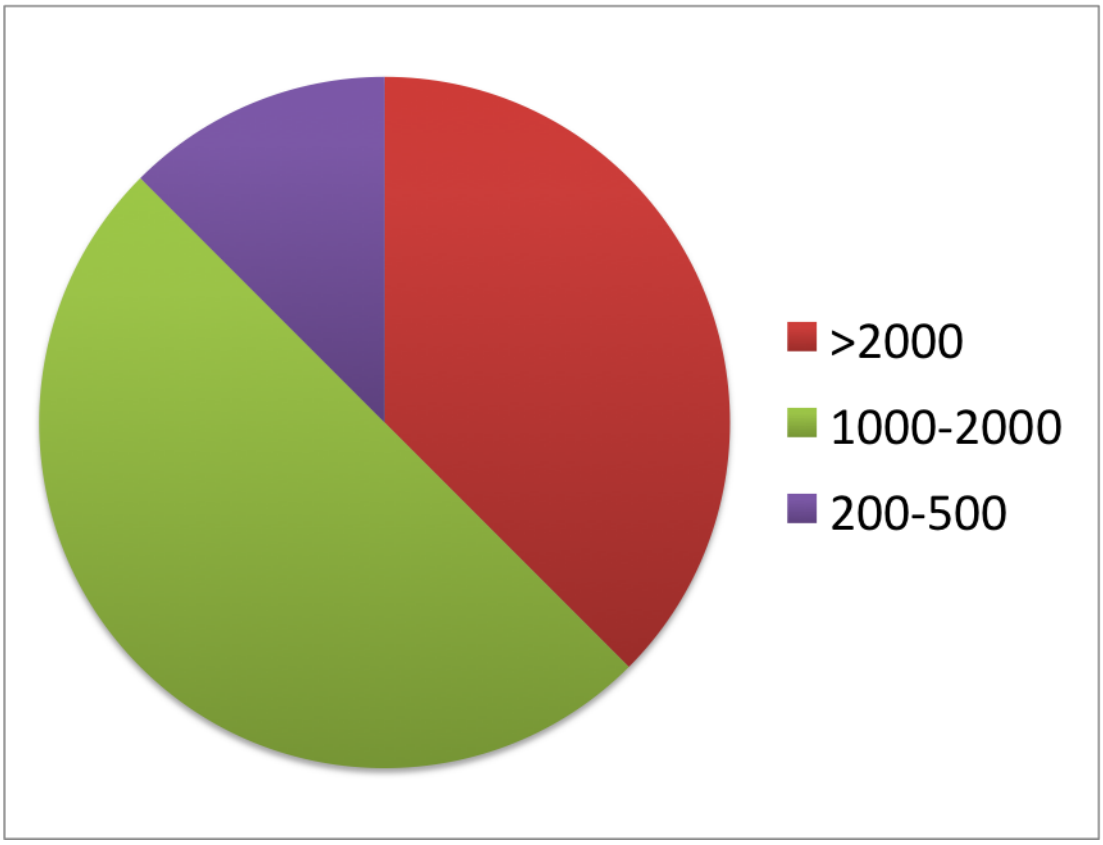


Figure 3.1: Number of Cores in Condor Pool

Typically most Condor pools have around 1000-2000 Cores and above. This is easily done with smaller numbers of computers due to most standard desktop machine available are quad core processors or a few just dual cores.

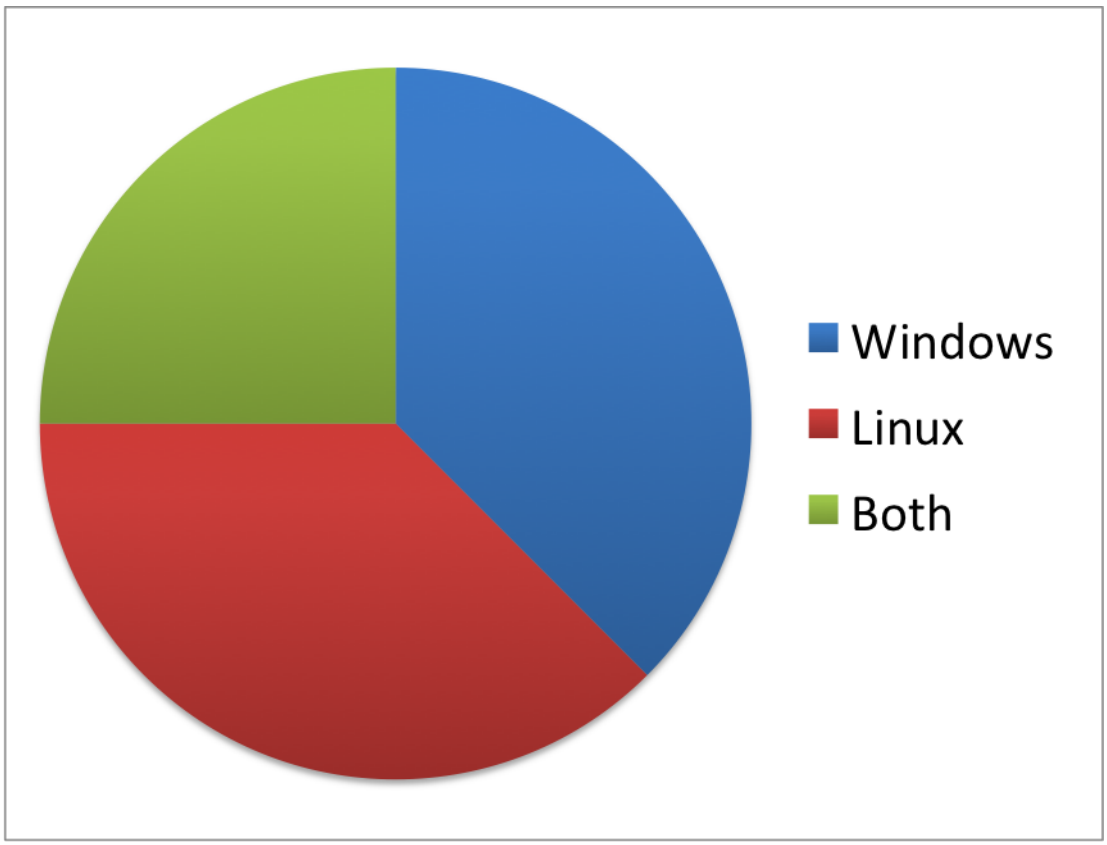


Figure 3.2: Type Operating Systems Condor Pool Operates On

TThis chart Figure 3.2 shows some extremely interesting information. The fact that there are equal number of pool using either Linux or Windows Condor Clients showing that it effective on either operating system.

More interestingly is the information gained by using it on both Windows and Linux. These users either rebooted lab machines out of hours into a flavour of Linux or ran the Linux Condor client or they were flocking between research pools or virtualised Linux.

Most of the users of the QGG are Linux users but most of the PC's in the University of Huddersfield user Windows so a suitable solution will need to be found.

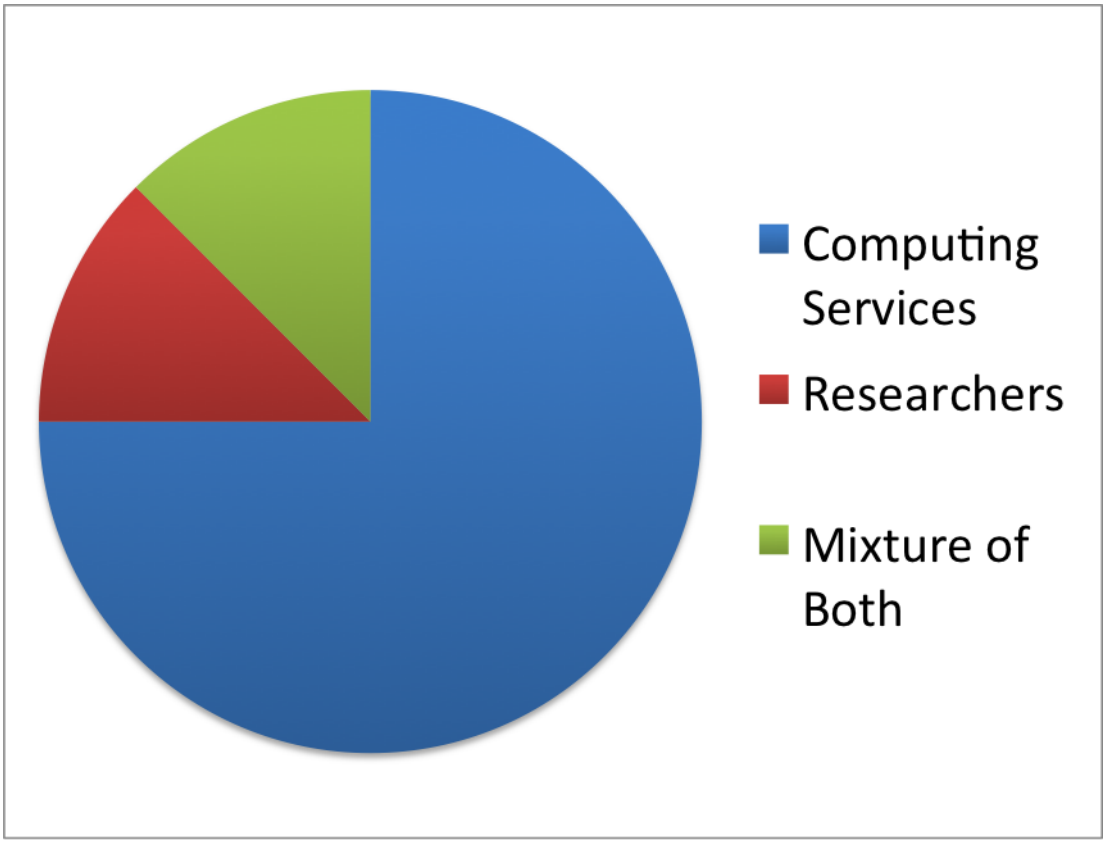


Figure 3.3: Who Administers The Condor Pools

Most of the Condor pools are administered by Computing Service staff at the respected institutions as shown in 3.3. This makes getting permission to deploy the pool across a larger number of computers easier especially if the computer laboratories are run by Computing Services. There is no surprise that researchers administer some of the Condor pools because if there is a need of extra computing power then researchers will try to find a way.

None of the institutions charge for use of the Condor pools but one asks researchers to make a contribution when they apply for research grants and funding.

3.2.2 Alternative Computing Resources

This section of the surveys aim was to find out what sized HPC/GPU clusters the institutions have access to so that it can be determined that it is useful to have Condor operating along side the HPC Clusters.

All but one respondent has some form of local HPC Cluster available to their users. 62.5% has access to external resources such as HECToR,HPC Wales, NGS and North West Grid (NW-GRID).

50% respondents have access to a local GP-GPU cluster and 25% use remote GP-GPU at HECToR.

3.2.3 Payment Schemes

We were also interested how other institutions funded their HPC cluster.

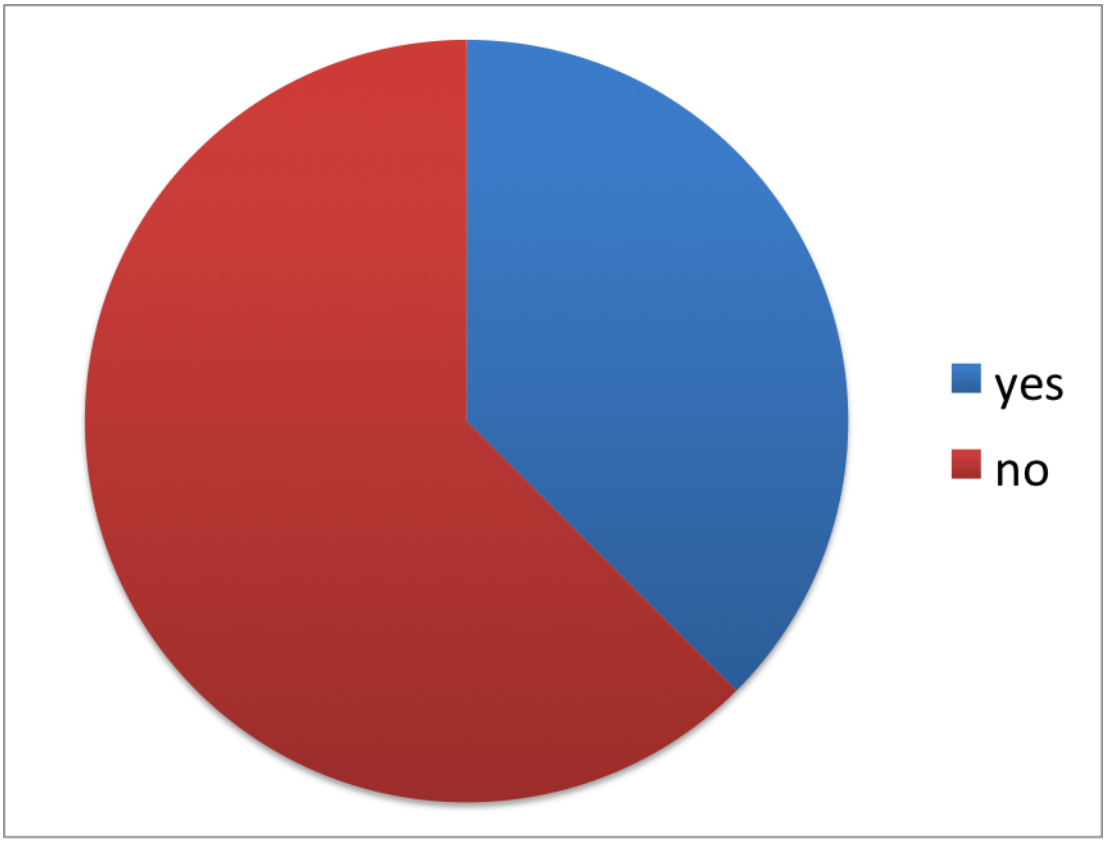


Figure 3.4: How Many Institutions Have a Payment Scheme For HPC

Most institutions do not have a payment scheme. The three that do charge have the following methods.

- By the CPU hour depending on the cluster
- Researcher by a blade to add to the cluster where they have dedicated access plus any that are idle

- CPU time is sold via the use of enCore cluster via on agreement with OCF. The University of Huddersfield has paid for time on the cluster. Some individual projects pay for resource to be hosted.

The contribution model seems to be the most sustainable model for universities because then HPC departments can then use the money to expand/maintain equipment that they have when required.

3.2.4 Users of the Condor Pool

This sections aim is to determine the subject areas that are able to use Condor as well as how many and whether it is used purely for research.

From the responses Condor pools have anywhere between 10 to 120 users.

In general Condor pools are used for research but there are some modules that use the pool for projects. Such as one of the respondents use it for 3rd year projects in biology and earth science. Interestingly one of the pools lets students use it freely without permission.

3.2.5 Support and Encouraging New Users

This section is determining how users are recruited and how are current users supported.

So all of the institutions offer a general Wiki page where users can learn the basics in order to connect and to run certain types of code. This feature reduces the amount of time administrators need to put into getting a new user who is computer literate onto Condor and running basic jobs. The downside to this is that it can either put off or confuse users who are not very computer literate or don't have the time to disseminate the information.

It is a very good sign that 5 out of 8 of the respondents run training sessions to help people use Condor. Training sessions are a good way of getting everyone who wants to use Condor because any problems or lack of understanding can rectified quickly. Running training sessions also help administrators see what types of users there are on the pool but it also allows users to be able to get to know who they should go to if they have any problems.

Only two of the institutions offered code support session. This could be very effective in getting users codes to run more efficiently on the Condor pool as well as on HPC clusters. In order to be able to run sessions there

need to be a member of staff who understands what good code is which can be hard to find.

Figure 3.5: How Condor Pool Are Advertised

Surprisingly two of the responded do not advertise their Condor pools!

So the general opinion is that word of mouth is seen to be the most effective. This could be because as user comes to sign up to use the HPC resources it could be easier to direct people into using Condor for the applications that are more suitable. One of the users introduced it in their introductory workshops for HPC; presumably it is easier to get people to use it if they have a suitable application.

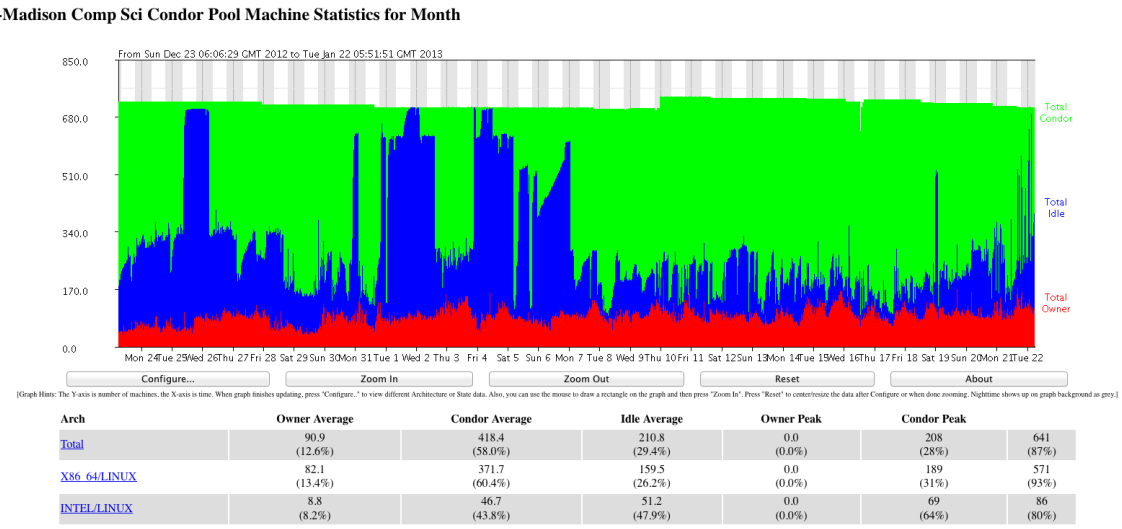
It is good to see that one institution uses a newsletter for Condor that can be used to keep users up to date on the state and usage of the Condor pool. While this is effective at keeping users up to date it would be difficult trying to get new users without sending out to the entire university, which could annoy people who do not need or intend to use Condor.

3.2.6 Monitoring Tools

All but one of the respondents use some form of monitoring tool. More users use a homemade monitoring solution rather than one that is available widely. This is due to either what is available is not good enough or that certain administrators want specific information from the Condor pool that is easier to extract using scripts that call upon certain Condor commands.

One respondent uses the Condor Quill add on that collates all the log files that Condor creates and puts them into a central database[37]. Condor Quill could be used to determine how effectively a Condor pool has been used so that any significant bad put can be identified.

Another respondent uses Condor View which reads the data in the Condor logs looking for how much time the execute machine have been idle, used by owner or executing Condor jobs and then generates graphical output displaying the data as shown in Figure 3.6.



Arch	Owner Average	Condor Average	Idle Average	Owner Peak	Condor Peak	
Total	90.9 (12.6%)	418.4 (58.0%)	210.8 (29.4%)	0.0 (0.0%)	208 (28%)	641 (87%)
X86_64/LINUX	82.1 (13.4%)	371.7 (60.4%)	159.5 (26.2%)	0.0 (0.0%)	189 (31%)	571 (93%)
INTEL/LINUX	8.8 (8.2%)	46.7 (43.8%)	51.2 (47.9%)	0.0 (0.0%)	69 (64%)	86 (80%)

Figure 3.6: Condor_View [1]

3.2.7 Condor Pool Configuration

This section will look at how Condor is generally configured.

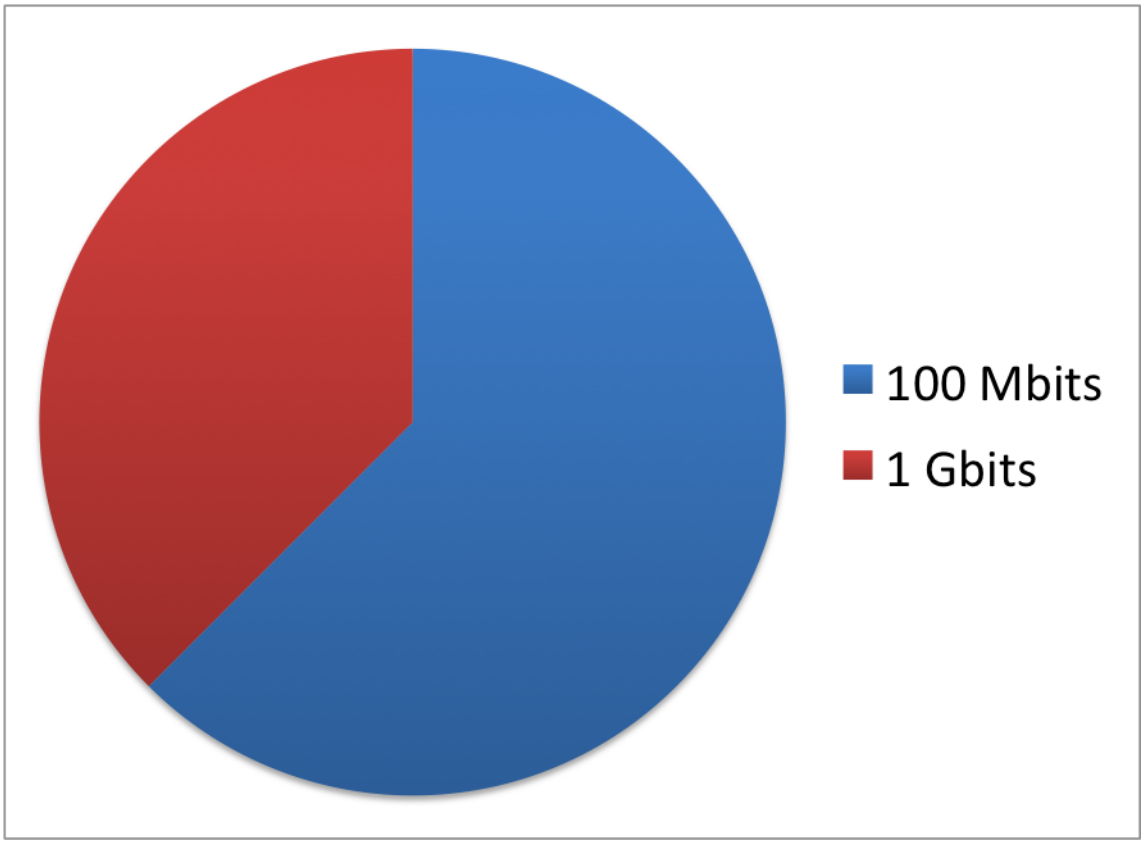


Figure 3.7: Network Speed for Condor Pools

As you can see in Figure 3.7 the typical network speed is 100 MBits/s this is due to most execute nodes are cycle stealing from a large number of computer laboratories so a standard low cost network is more suitable. To be able to run Condor on a 1 Gbit link would not normally be economically viable over large networks with lots of interconnects.

Security used is limited to some standard security such as using an LDAP for user authentication as well as the use of SSH and GSI-SSH keys, Kerberos, limiting which machines can submit to Condor, standard Linux authentication PAM.

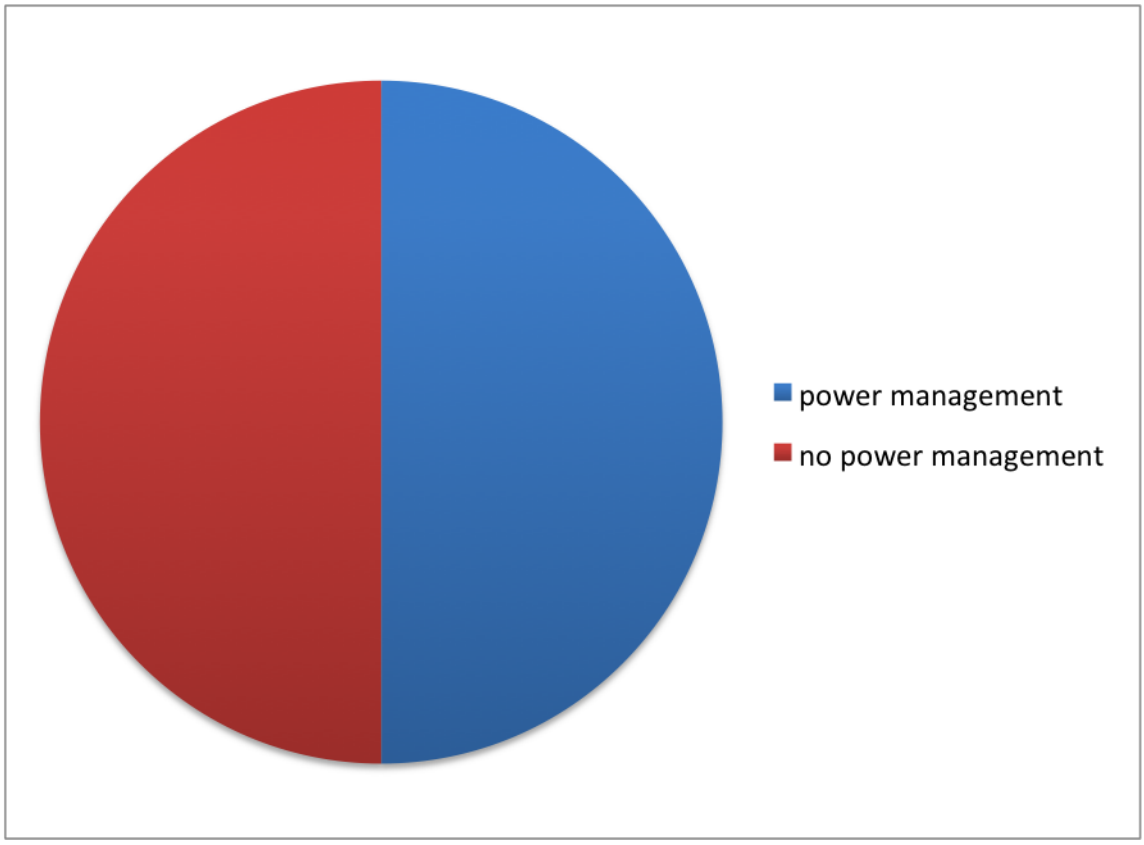


Figure 3.8: Power Managment

It is unexpected that over half of the respondents did not use any power management on their Condor pools when there is such a push for green IT and money saving. All who do use power management allow the compute nodes to power down/put to sleep after it has become idle without jobs. 3 out of the 4 do use WoL when required. To note one of the respondents tried to use the Condors built in power management but had too many difficulties so they when for a home made solution.

3.2.8 Institutional Views

So the general consensus of the users of the laboratory machines that are setup as execute nodes is that the users do not know Condor is installed or running in the background. The only complaint that there has been is one of the jobs was too memory intensive and it ended up locking the machine up. One institution that ran Condor out of hours had to be wary of update or virus scans running too long.

The institutions computing service generally allow the use of Condor over the networks and on the computer laboratory machines. One institution hosts the Condor Master which is a good option providing it is in their data centre so that it will have the best connection to the university network backbone as well as having a good UPS/Cooling for the node. Another has a member of staff allocated which looks after the Condor service. Also Condor execute software at another institution is updated annually with the new system image.

The benefits of Condor to the institution is that it is a low cost, low capital resource that is extremely powerful that can be used to further research depending on the applications. Also it can be used to alleviate the stress from computer clusters for serial and embarrassingly parallel tasks.

Some of the disadvantages of Condor in the view of institution it is run at is that Condor can be difficult to program for and it is suitable for certain types of jobs. As a side effect of Condor using idle laboratory machine is that the number of idle machine is very low making the Condor pool also very small so there will be less jobs run during the day. One institution has had complaints from the University energy officer about the computing cycles Condor uses was seen as ”wasted energy”. Another was that Condor was undercutting other paid HPC resource so there were complaints about under cutting these resources.

3.3 General Comments About Condor

One important comment for UK universities is that there is a heavy bias towards Linux Condor pools within the Condor documentation and more support provided by Condor. When most computers those are available to run Condor as compute nodes tend to be Windows based but would make more sense to have a bit more support.

While Condor View can be a useful monitoring tool it can be too basic, so it would be better to have something similar to Ganglia with a better interface and a greater detail of information when needed.

One of the respondents that used Condor purely on Linux machines praises Condor but they have not seen a demand for Windows Condor machines so they will not be expanding the pool in that direction.

Chapter 4

Condor Deployment at the University of Huddersfield

Originally the Condor deployment concentrated on developing a green way of deploying it, but this developed further into allowing users to be able to use a Linux version when required.

4.1 Available resources

The University of Huddersfield has a large amount of computer laboratories for the use of students to use for assignments and tutorial work. During the day there may be times where the computers laboratories are extremely busy and there may be times of the day where the scheduled teaching session may not have enough students to fill the computer. This means there could be a large number of machines available to run Condor even during busy times of the day.

These computers have power saving policies implemented on them so they will go into hibernation to save power so its not a big problem that all computers are not been used 100% of the day.

The University of Huddersfield HPC resources are current at used on average at 55% throughout the year of 2012 but during peak times the usage can increase up to 92% making queue time longer than expected show in Figure 4.1. The main problem is that the user base is expanding as well as users simulation increasing in the amount of time to run faster than new resource can come online. With a lack of funding available to expand local HPC resources it is essential to make use of any other available resources. Moving serial applications to a HTC resource would free HPC cluster to run more parallel application.

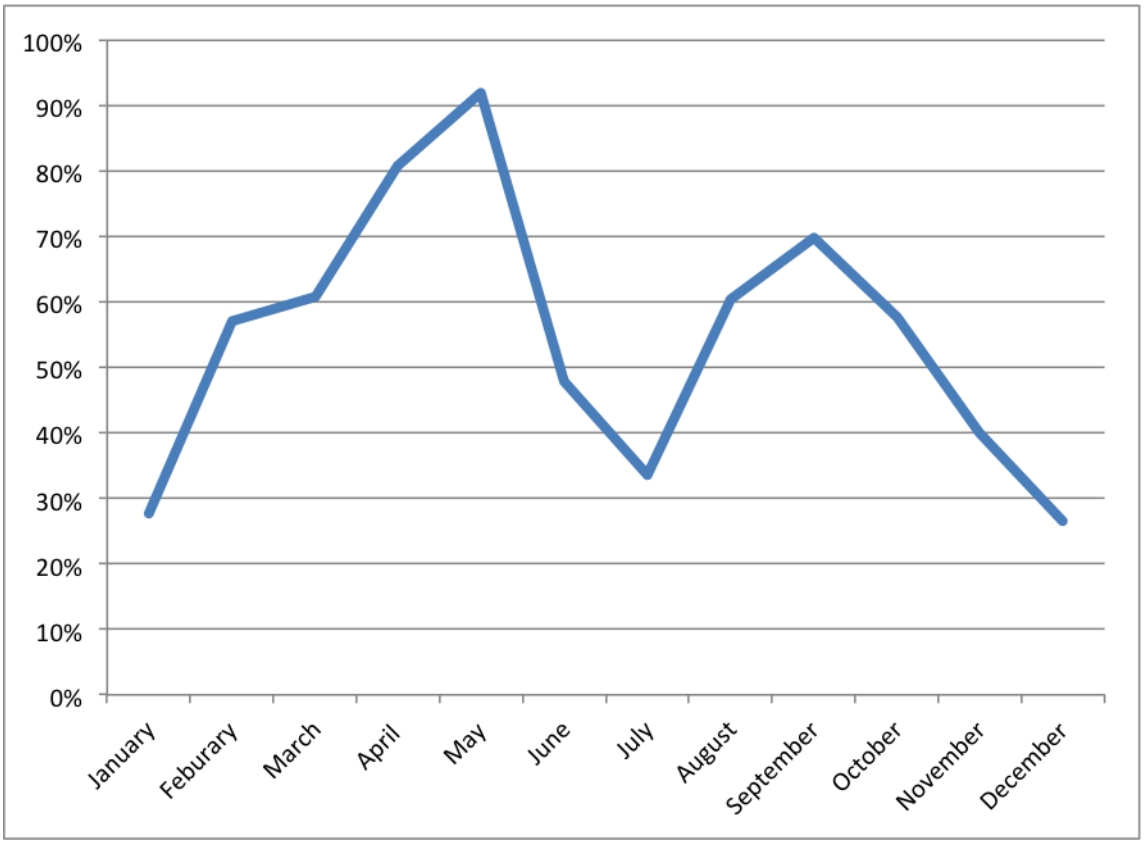


Figure 4.1: QGG Utilisation for 2012

The University has introduced a minimum specification policy for computers in teaching laboratories of a minimum of a quad core processor that benefits the Condor pool because of the extra number of cores that is on the processor increases the number of possible job slots for the Condor pool. Having computers with more cores brings in the possibility of been able to run parallel tasks as described later on in HTPC section. The newer processors have a greater support for Virtualization such as Intel VT-d and the AMD AMD-V modules that make virtual machines run more efficiently. The machines also run Windows 7 64-bit images.

In the department of Engineering alone there are approximately 500 cores available in computer laboratories. That currently provide more compute cores that is available on any of the QGG clusters.

4.2 University of Huddersfield Deployment

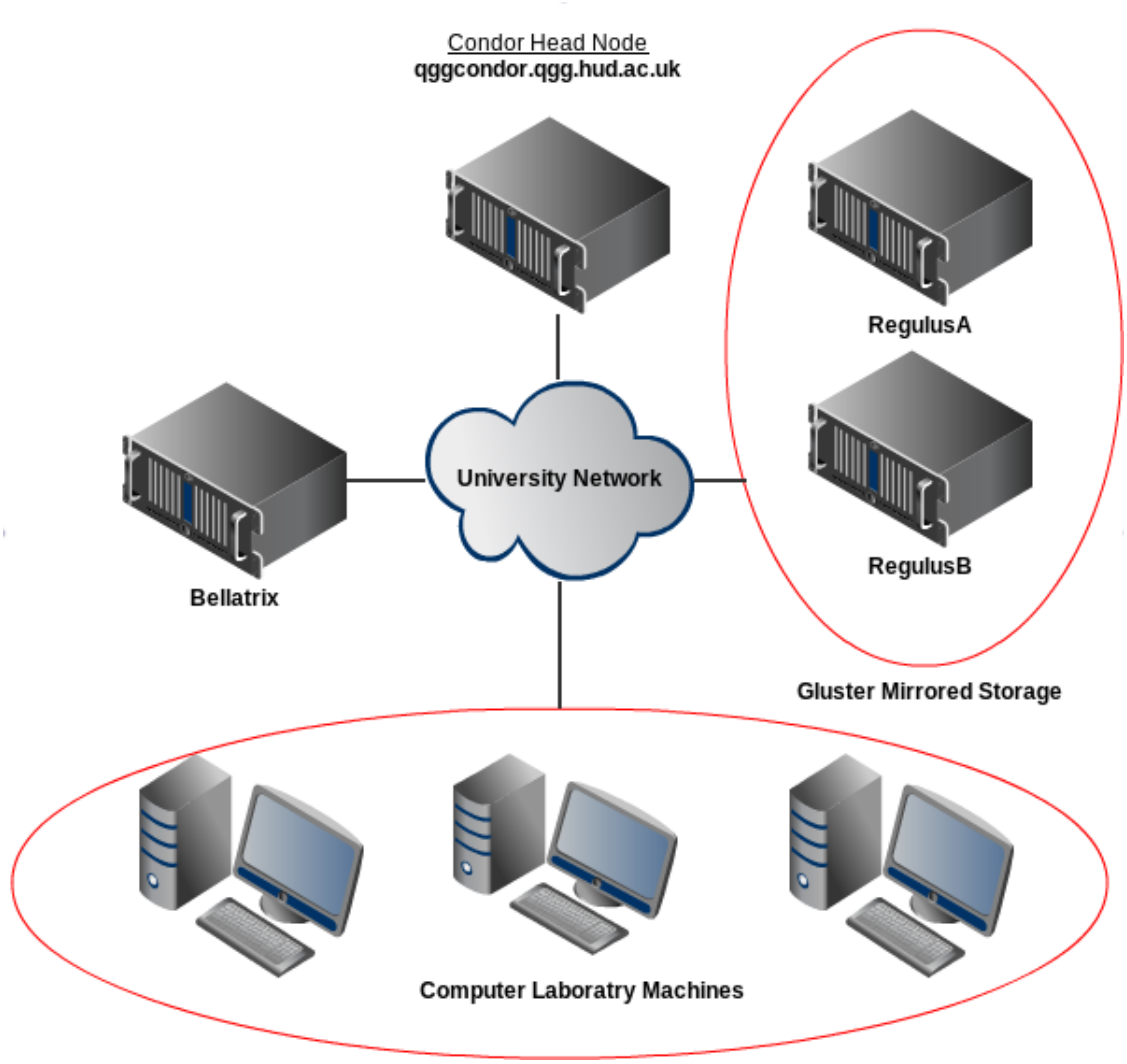


Figure 4.2: System layout

Condor will be deployed so existing users of the QGG can easily use this extra service. This will be achieved by using the existing login node Bellatrix using SSH key authentication which queries the QGG LDAP. The users will then SSH over to Qggcondor to be able to submit jobs. Users will also be able to submit jobs using Condor-G to other offsite resources.

The users home folders will be mounted on the head node so that they have access to their data when they submit a job.

Application that have been tested by administrators will be mounted as the *condor_apps* folder with example submission scripts for these application so that the user doesn't have to waste time trying to get the job to run. Also it means that the executable can be kept for a local repository.

The power management scripts will run on the head nodes using Crontabs every 15 minutes.

The following steps outline are describe what steps are needed to be completed in order to have a working Condor pool;

- Configure a working Condor Master
- Configue Windows Condor Executable nodes are implemented
- Configure the Pools of Virtual Boxes (PoVB) image so that it is secure and Checkpointing is implemented.
- Automate the way that IP and MAC addresses are collected for use with WoL
- Modify the existing power script from University of Liverpool so that it can also control services
- Modify the PoVB and Windows Condor services so that HTPC is possible
- Describe the steps taken to create an easily deployable MSI installer

4.3 Configuration of QGGCondor

Condor will be installed on dual socket Quad Core Opteron 2350 2GHz processor with 16GB of RAM. Using Condor Version condor-7.6.7, which was the current stable release at the time of deployment.

Once downloaded from the Condor website the file should be extracted and then moved to the desired location which in this case is */opt/condor-7.6.7*

Then use the following command from the location of the Condor installer.

./condor_config --install=/opt/condor-7.6.7 --local-dir=/opt/condor-7.6.7/local.qggcondor --type=submit,manager --central-manager=10.71.88.84 --owner=root

This command configures Condor to be installed entirely from */etc/condor-7.6.7* so that when a newer version of Condor is installed then it can be

tested from another directory before it can go production. This head node has been configured to be the Scheduler and the Resource Manager. This has been done because it reduces the complexity of the system.

Once Condor has finished installing the following changes need to be checked, modified or added to the */opt/condor-7.6.7/etc/condor_config*

RELEASE_DIR	= /opt/condor-7.6.7
LOCAL_DIR	= /opt/condor-7.6.7/local.qggcondor
LOCAL_CONFIG_FILE	= /opt/condor-7.6.7/local.qggcondor/ condor_config.local
ALLOW_WRITE	= *
DAGMAN_LOG_ON_NFS_IS_ERROR	= False

Figure 4.3: Changes made to *condor_config*

The first 3 are to define where Condor will find these locations.

The ALLOW_WRITE is very important or otherwise Condor will not run jobs.

There is a common issue with using the DAGMAN scripts from a network file share and Condor so the solution is to add the line at the bottom of the figure 4.3.

Check that the */opt/condor-7.6.7/local.qggcondor/condor_config.local* has the following values.

CONDOR_HOST	= 10.71.88.84
RELEASE_DIR	= /opt/condor-7.6.7
LOCAL_DIR	= /opt/condor-7.6.7/local.qggcondor
DAEMON_LIST	= COLLECTOR, MASTER, NEGOTIATOR, SCHEDD

Figure 4.4: Changes made to *condor_config.local*

The first three needs to be check so Condor is looking at the correct directories and it is important that the DAEMON_LIST has the daemons mentioned in figure 4.4 otherwise it will not schedule jobs and run them.

Once complete add the following to either just the individual users *.bashrc* or to the */etc/profile* so that the Condor commands can be used by all of the users of the system.

```
export  CONDOR_CONFIG=/opt/condor-7.6.7/etc/condor_config
export  PATH=/opt/condor-7.6.7/bin:${PATH}
export  PATH=/opt/condor-7.6.7/sbin${PATH}
```

Figure 4.5: Changes made to */etc/profile*

The results shown in Figure 4.6 show a working Condor Master that has a single execute machine.

```
Name               OpSys      Arch   State     Activity LoadAv Mem   ActvtyTime

slot1@DG965.AD.HUD WINNT51    INTEL  Unclaimed Idle     0.000  1014  0+05:30:09
slot2@DG965.AD.HUD WINNT51    INTEL  Unclaimed Idle     0.140  1014  0+01:50:05
                     Total Owner Claimed Unclaimed Matched Preempting Backfill

       INTEL/WINNT51     2     0       0         2       0          0        0

              Total      2     0       0         2       0          0        0
```

Figure 4.6: *condor_status* output

Also run *condor_q* and it will show an empty queue such as shown in figure 4.7.

```
-- Submitter: qggcondor.qgg.hud.ac.uk : <10.71.88.84:59649> : qggcondor.qgg.hud.ac.uk
 ID      OWNER            SUBMITTED     RUN_TIME ST PRI SIZE CMD

0 jobs; 0 idle, 0 running, 0 held
```

Figure 4.7: *condor_q* output

ENLARGE

Checkpointing is configured by default to periodically perform once every 3 hours $\pm$ 30mins and this will be saved from the location where the job is submitted i.e. qggcondor

4.4 Executable Machine Configuration

The University of Huddersfield has a large number of idle Windows machines so to make the most of this untapped resource these machines will run the Windows Condor executable as well as having the option of running PoVB for Linux Applications.

4.4.1 Windows Condor Execute Machine Installation

This is easily done by downloading the MSI installer for the latest stable version of Condor from the website. When running the installer make it add an existing Condor pool and direct it to QGGCondor. Then select the option for ”Wait for 15 minutes of keyboard and CPU activity” and to drop the job when a user comes back. Then change the permission to write to ”*”. Then click next till the install location and keep it to the C:\condor

Once installed restart the machine and then copy over the Windows Condor *condor_config* into the C:\condor folder.

The additions made to the configuration are the following:-

```
TimeToWait      = (30 * $(MINUTE)
Should Hibernate = ( (KeyboardIdle >$(StartIdleTime)) \
                 && $(CPUIdle) \
                 && ($StateTimer) >$TimeToWait)) )
```

Figure 4.8: Windows Power Saving Settings Made To *C:\condor_config*

This allows Condor to stop waiting and it lets the system power settings to then take over. The Windows client will then wait for 15 minutes to see if the system is idle and then after another 15 minutes it will then allow the machine to go into the planned hibernation.

The modifications made to the Windows condor_config is in Appendix A

4.4.2 Pools of Virtual Boxes

PoVB was created by the University of Marquette to allow Linux Condor jobs to be run on Windows computers. These Windows machines virtualise Linux using Sun Virtual Box which has Condor pre-installed its is easily deployed. PoVB is capable of detects the activity of the Windows machine so that if a user comes along then the job is dropped and started when a machine becomes available.

The configuration files for PoVB is located in the *C:\povb\shared* and from these each time the PoVB service starts it regenerates the snapshots that virtual box uses. The advantage of regenerating the snap shot each time is that it stops the virtual machine from using too much storage space. Another advantage is its easier to change the configuration for Condor without having to boot the virtual machine and make the changes.

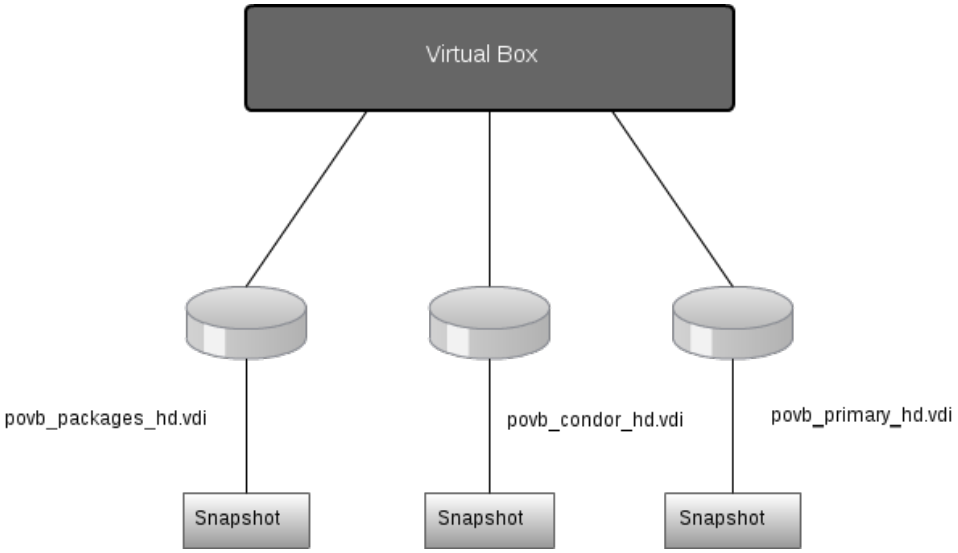


Figure 4.9: PoVB Hard Drives[2]

The primary hard drive has the operating system and it is where the shared folder is mounted for the Condor configuration.

The Condor hard drive has the Condor files

The Packages hard drive allows for deployment of software so that it doesn't have to be transferred to every time that it is run.

4.4.3 Pools of Virtual Boxes Configuration

Installation of PoVB is done by downloading the installation folder from [38]. Then extract the folder and then edit the following files with these changes in table 4.10

```
KeyboardBusy      = (POVB_HostOsKeyboardIdle <15 * $(MINUTE))
ConsoleBusy       = (POVB_HostOsKeyboardIdle <15 * $(MINUTE))
StartIdleTime     = (15 * $(MINUTE))
ContinueIdleTime  = (15 * $(MINUTE))
TimeToWait        = (30 * $(MINUTE))
ShouldHibernate   = ((POVB_HostOsKeyboardIdle >$(StartIdleTime)) \
                    && $(CPUIdle) \
                    && ($StateTimer) >$TimeToWait)) )
```

Figure 4.10: Changes made to *condor_congig.policy*

The First four lines make sure that PoVB appears as idle while the last two line are a slightly modified version of the power saving settings made to the Windows Clients but adjust for the different variables.

The next changes are to configure the PoVB installation, which simply points PoVB to the Condor Master and then configures it to use Network Address Translation (NAT). NAT is used because it does not request a new IP address from the DHCP server that would double the effective IP addresses, which could cause trouble in the future.

```
CONDOR_MASTER      = 10.71.88.84
NETWORKING_TYPE    NAT
```

Figure 4.11: Changes made to *povb_config*

To install PoVB navigate to its unpacked location from a command prompt and run the following command *"povb-x86_64-2.0.1\povb_installer.exe --vbox-download http://download.virtualbox.org/virtualbox/3.1.8/VirtualBox-3.1.8-61349-Win.exe"*. This command will install PoVB to *C:/povb* and it will also download and install Virtual Box 3.1.8 which is the maximum version supported. Once installed from Services change the startup of PoVB service to manual start-up. Copy over a modified *povb_packages_hd.vid* if required and then restart the machine.

4.5 Automation of IP and MAC address collection for WoL

With the power saving for Condor and PoVB so that machines will go into a lower power state it means that there could be the possibility of not enough machine will be awake to run the jobs that are in the queue so having the ability to wake machine remotely using WoL Protocols.

WoL requires the MAC and IP of the machine that is going to be woken so that it can wake machines up in any subnet. Collecting this information whilst installing Condor and PoVB is a solution it becomes impractical when large numbers of machines are been imaged so a better solution need to be created.

The solution that has been created allows for autonomous collection of the information, which can also serve as a record of the size of the Condor pool.

Extracting the information from the *condor_status* and then populating this information into a file called *WOLinfo.txt* that achieves the information. This file is then sorted and any duplicated information is then removed.

```
CE113-01 10.4.84.11 00:1C:C0:EF:97:61
CE113-02 10.4.84.12 00:1C:C0:EF:97:75
CE113-03 10.4.84.13 00:1C:C0:E2:91:01
CE113-04 10.4.84.14 00:1C:C0:EF:97:7F
CE113-05 10.4.84.15 00:1C:C0:EF:97:6B
CE113-07 10.4.84.17 00:1C:C0:EF:97:6C
CE113-08 10.4.84.18 00:1C:C0:EF:97:7B
CE113-09 10.4.84.19 00:1C:C0:EF:97:1E
CE113-11 10.4.84.21 00:1C:C0:EF:97:9E
CE113-12 10.4.84.22 00:1C:C0:EF:97:86
```

Figure 4.12: *WOLinfo.txt* Output

The output looks like Figure:4.12 above. The first coulomb shows the hostname for the corresponding machine and the second is the IP address and the last is the corresponding MAC address.

The script that generates the file is found in Appendix B and the script is called *mac.pl*. It runs once every hour because as long as the machine is on and has Condor installed then the information can be extracted and updated.

4.6 Power Management

4.6.1 The University of Liverpool WoL script

The original script came from a post from Ian Smith on the Condor forums[39] that is shown in Appendix C. The script was written in Perl and is designed to run as a Cron job. The script is designed to be able to wake powered down PC's using WoL. The MAC addresses that are required for WoL are collected and then kept in files for each computing laboratory.

When there are more than 10 idle Condor jobs then the script will try to wake up all of the machines that are in a low power state. When machines are woken up an email is sent detailing the amount of machines woken for the number of idle jobs.

While this is effective the Waking of the machines is too much of a blanket approach to waking the machines up, which could cause stress to the components. This approach could also annoy users of quiet computer labs due to the noise of all the machines waking up as concluded in Ian Smiths paper[40] that has been used in the Literature Review.

4.6.2 Extraction of Information

For the university of Huddersfield it is important to be able to run dual operating systems for the users. This required the investigation of been able to control services in Windows 7 remotely as well as how to provide a dual operating system Condor pool.

The first task is to modify the script so that it can extract all of the details so that the number of machine awake and the types of machine online such as Windows 7, Windows XP or Linux. This data is extracted from the *condor_status* and *condor_q* commands.

```
-------------------------------------------
Number of busy machines = 0
Number of idle machines = 329

Number of idle Win7 = 329
Number of idle WinXP = 0
Number of required machines = -329

Number of idle jobs = 0

Number of idle Windows jobs = 0
Number of idle Windows machines = 329
Number of Windows machines required = -329

Number of idle Linux jobs = 0
Number of idle Linux machines = 0
Number of Linux machines required = 0
-------------------------------------------

################################
#                              #
# Enough machines are online!!! #
#                              #
################################
```

Figure 4.13: Condor Power Script Version 1

The Figure 4.13 shows the output required to check how many machines are idle/busy and the types of machines that are idle. The number

of machines that are required for the jobs are identified and then a message indicates if there are not enough machines and how many jobs slots are required. The message appears at the bottom. The code is displayed in the Appendix D.

The information gathered from this script is used as the basis for the further work carried out. The next step would be to introduce the basic step of using WoL to wake machines to meet the demand required by excessive number of jobs.

4.6.3 Controlling Services Locally

So initially it was worth investigating a back up plan in case remote control of the service is not possible. The back up plan was to schedule the PoVB service to start when the computer laboratory shut and then stop the service when the labs opened again. This involve in creating a simple text file that turns the PoVB service on and one to switch it off again. So to turn off the Windows Condor Client it would look like *"net stop condor"* and to start PoVB *"net start povb"*

The next step is to configure the Windows 7 scheduler to turn the service on and off. This requires the user to be an administrator to create the scheduled task.

The issue with scheduling tasks is that the job queue may be overloaded but the main issue that when the lab shuts only a few computers may be online at that time which would either mean that the queued jobs would have to run through on a few machines.

Alternatively WoL could wake all the machine when the lab shuts but it means that extra machines would be online when they are not required. The main issue is if there was a queue of jobs that have not finished overnight they would end up queued for the next day, which would not be efficient.

It would me more efficient if individual machines could have been woken up and have the PoVB service started so that jobs can be dealt with when required with machines that are already powered up.

Not all labs shutdown at the same time so it causes initial issues customising the scheduled tasks according to each computer lab.

4.6.4 Controlling Services Remotely

Ideally it would be more efficient to run Linux jobs when required. This required working out a way of turning Windows services on and off from

a Linux server (Condor Master Node) so that the power script could be modified to perform this task autonomously. Which can be done by using the following command: *"net rpc service stop/start -I $IPADDRESS -U $USERNAME%$PASSWORD $SERVICE_NAME"*[41]. So this command can effectively control any service providing the user has permission to and providing that "Remote Service Management (NP-In) Domain" is allowed within the firewall of the machine giving the option to automate which Condor version needs to run depending on the jobs in the queue.

4.6.5 Final Power Management Script With Service Control

The final script uses a combination of the scripts and information from the previous sections that when combined allows for the control of waking up machines when required and controlling services remotely.

Using the first script which extracts the types of jobs are in the queue and then adding conditional statement that will either wake machines up or change the service that is been run on the execute machines.

The additions to the script will do the following

- Determining which machines are offline
- Determining which machines are running the Condor Client
- Determining which machines are running the PoVB Client
- Waking machines from the offline list
- Changing Windows clients to PoVB Clients
- Changing PoVB clients to Windows Clients

In order to determine the state of a machine the output from the MAC and IP collection script is read in and then compared to the command so that it can be known which machines are offline, running Windows Condor Client or PoVB Each one of the commands fills an array with the hostnames of the machines that meet the criteria and it will be compared with the *WoLinfo.txt*.

A while loop is initiated that for every line will check each array to see if the hostname exists within. If it is true for any of the arrays then the hostname populates that array. If it doesn't match any case then this means that the machine is offline and it populates that array.

Using these arrays can be used to wake machines using the offline array or using the Windows array machines can change services to PoVB.

```
my @WINDOWS=scalar(‘condor_status |grep ’WINNT’|grep ’Idle’|cut -c 7-
    18|uniq|cut -f1 -d”.”‘);

my @LINUX=scalar(‘condor_status |grep ’LINUX’|grep ’Idle’|cut -c 7-
    19|uniq|cut -f1 -d”.”‘);

my @BUSY= scalar(‘condor_status |grep ’Claimed’|Owner’|cut -c 7-
    19|uniq|cut -f1 -d”.”‘);
```

Figure 4.14: Commands To Get The Hostnames To Compare

4.7 High Throughput Parallel Computing Implementation

In order to allow Condor compute node to perform HTPC a few modifications need to be made to the *condor_config* for the Windows client or *condor_config.policy* for the PoVB. Using the information found here [42] or the amended version in Appendix E, this allows for the whole machine to be booked. Booking is required so that Condor can run the parallel code across the machine without a risk of another job appearing which could crash the machine or make it extremely slow. Been able to book an entire machine may be extremely useful for jobs that require large amounts of RAM.

In order to be able to run the parallel job Condor requires a local compilation of an MPI library such as OpenMPI or for the drive to be mounted such as in the PoVB setup the MPI library is located in the *povb_packages_hd.vdi*.

4.8 Creating an MSI

In order to be able to deploy Condor and PoVB easily across campus there needs to be a way of installing the program. This can be achieved by using Group Policies that require an MSI in order to install programs.

The MSI for Condor has been created by creating a batch script with the required variables and then converting from a .bat to an .exe with a free piece of software, then from an .exe to a .msi using some more free open source software. Once this is done test the MSI to check that it performs as required.

The batch script for the installation of Condor is shown in Figure 4.15.

```
@echo on
set ARGS=NEWPOOL="N"
set ARGS=%ARGS% POOLNAME=""
set ARGS=%ARGS% RUNJOBS="C"
set ARGS=%ARGS% VACATEJOBS="Y"
set ARGS=%ARGS% SUBMITJOBS="N"
set ARGS=%ARGS% CONDOREMAIL="hpc-rc@hud.ac.uk"
set ARGS=%ARGS% SMTPSERVER="smtp.localhost"
set ARGS=%ARGS% HOSTALLOWREAD="*"
set ARGS=%ARGS% HOSTALLOWWRITE="*"
set ARGS=%ARGS% HOSTALLOWADMINISTRATOR \
    = "qggcondor.qgg.hud.ac.uk"
set ARGS=%ARGS% INSTALLDIR="C:\Condor"
set ARGS=%ARGS% POOLHOSTNAME="qggcondor.qgg.hud.ac.uk"
set ARGS=%ARGS% ACCOUNTINGDOMAIN="ad.hud.ac.uk"
set ARGS=%ARGS% JVMLOCATION="C:\Windows\system32\java.exe"
set ARGS=%ARGS% USEVMUNIVERSE="N"
set ARGS=%ARGS% USEHDFS="N"
msiexec /qb- /i condor-7.8.6-winnt-x86.msi %ARGS%
```

Figure 4.15: Installation of Condor using Batch Script[3]

The batch script to install PoVB simply contains *"povb-x86_64-2.0.1\povb_installer.exe --vbox-download http://download.virtualbox.org/virtualbox/3.1.8/VirtualBox-3.1.8-61349-Win.exe"* from the PoVB installation chapter.

Then to finish the full configuration a final batch script is run that copies over any configuration files, from a server called Vega, which cannot be automatically during installation, generated which looks like Figure 4.16

and then restart the machine.

```
xcopy \\vega\Installer\condor\condor_config C:\Condor /Y

xcopy \\vega\Intsaller\povb\shared\condor_config.policy \
    C:\povb\shared\condor_config /Y

net stop povb

sc config povb names start= demand

shutdown -r -t 01
```

Figure 4.16: Installation of Condor using Batch Script

When these scripts have been used then the script should be converted into an MSI at the beginning of the chapter and then applying the group policy to a computer laboratory that will then install Condor and PoVB as desired.

Chapter 5

Conclusion

The literature review that was carried out identified that green IT is about reducing the amount of energy used within IT, which in tern reduces the production of carbon emissions.

By researching into the different HTC middleware option showed that Condor was the most adaptable system available because it has been used in a number of different UK universities. By analysing how Condor has been used revealed some of the useful tools and techniques that can be used by Condor to make it greener

The survey was an incredibly useful tool because it revealed how other UK universities run Condor alongside existing HPC resources. This survey also showed how Condor has been configured so that users can get the most use out of it for research.

The power management scripts that were developed are not only able to power up extra machines when the queue status requires it. The additional option of been able to switch between the different Condor clients will be a useful addition. This makes the Condor pool able to cater to far more of the research community who use Windows or Linux applications.

The creation of a MSI that can deploy the Condor solution developed makes deploying the Condor pool a easier and quicker task.

Condor on the execute machines conform to the green IT policy because after 15 minutes of been idle it will start a job but if it has been idle for a further 15 minutes it will go into a low power state. Condor will also not interfere with the user experience because it will drop the job as soon as a user comes to the computer and it will start on another computer that is idle.

It can be concluded that the original project aims and objectives have been achieved. This work on Condor has resulted in a publication at Digital Research 2012[43] where the reviewers gave very positive feedback.

5.1 Further Work

Many machines in the Condor pool have a graphics card installed that could also be utilised if configured correctly. This could be an extremely useful addition because it makes Condor able to cater to even more researchers who may be programming code in CUDA.

A refinement that could be made is to modify the IP and MAC collection script to populate a database that would also contain benchmark information so that the most powerful machines could be woken more often. This would be improvement because the more powerful machine are often newer more green machines that would save energy. These machines would complete jobs quicker so that fewer machines will need to be woken up.

References

[1] "UW-Madison Comp Sci Condor Machine Statistics for Month." http://condor-view.cs.wisc.edu/condor-view-applet/Month.html.

[2] C. A. Struble, "Pools of Virtual Boxes A Year Later." http://research.cs.wisc.edu/htcondor/CondorWeek2010/condor-presentations/struble-pools-virtual-boxes.pdf, Apr. 2010.

[3] "3.2 installation." http://research.cs.wisc.edu/htcondor/manual/v7.9/3_2Installation.html.

[4] "EZ GPO:ENERGY STAR." http://www.energystar.gov/index.cfm?c=power_mgt.pr_power_

[5] "Condor Project Homepage." http://research.cs.wisc.edu/condor.

[6] I. Kureshi, "Establishing a university grid for hpc applications." September 2010.

[7] "Daresbury laboratory: Facilities and services | STFC." http://www.stfc.ac.uk/About+STFC/342.aspx.

[8] "Autodesk - backburner 2012.1.1." http://usa.autodesk.com/adsk/servlet/ps/dl/item?siteID

[9] "History : ENERGY STAR." http://www.energystar.gov/index.cfm?c=about.ab$0history.

[10] R. L. Mitchell, "SEVEN STEPS TO a green data center," *Computerworld*, vol. 41, pp. 23–26, June 2007.

[11] C. Garretson, "Inside a green data center," *Network World*, vol. 24, pp. 49–52, Nov. 2007.

[12] G. Anthes, "Green grows the data center," *Computerworld*, vol. 41, p. 38, Sept. 2007.

[13] R. McFarlane, "Cooling the green data center," *EC&M Electrical Construction & Maintenance*, vol. 111, pp. 14–16, Apr. 2012.

[14] A. Beck, “High Throughput Computing: An Interview With Miron Livny.” http://research.cs.wisc.edu/condor/HPCwire.1, June 1997.

[15] U. o. W. Center for High Throughput Computing, “2.10 DAGMan Applications.” http://research.cs.wisc.edu/htcondor/manual/v7.6/2$010DAGMan$0Applications.html, 2012.

[16] “High Throughput Parallel Computing(HTPC).” http://research.cs.wisc.edu/condor/CondorWeek2010/condor-presentations/thain-fraser-hptc.pdf, Apr. 2010.

[17] Microsoft, “Cluster of workstations.” http://download.microsoft.com/download/3/C/9/3C9BE860-51F4-4A7A-BEE1-97F4DDA54FBC/Cluster_of_Workstations_SP2.docx, June 2011.

[18] University of California, “BOINC.” http://boinc.berkeley.edu/, Nov. 2012.

[19] University of California, “Choosing BOINC projects.” http://boinc.berkeley.edu/projects.php, Oct. 2012.

[20] R. Hipschman, “About SETI@home page 2.” http://seticlassic.ssl.berkeley.edu/about_seti/about_seti_at_home_2.html, 2003.

[21] R. Hipschman, “About SETI@home page 3.” http://seticlassic.ssl.berkeley.edu/about_seti/about_seti_at_home_3.html, 2003.

[22] “TOP500 List - June 2012 (1-100) | TOP500 Supercomputing Sites.” http://top500.org/list/2012/06/100, June 2012.

[23] “How did the Condor project start?.” http://research.cs.wisc.edu/condor/background.html, Jan. 2013.

[24] “What is Condor?.” http://research.cs.wisc.edu/condor/description.html, Jan. 2013.

[25] I. C. Smith, *Experiences with Running MATLAB Applications on a Power-Saving Condor Pool*. Sept. 2009.

[26] “IT Service-Newcastle University.” http://www.ncl.ac.uk/itservice/condor/aboutcondor, 2012.

[27] C. Gerrard, P. Haldane, S. Hamlander, S. McGough, P. Robinson, D. Sharples, D. Swan, P. Watson, and S. Wheater, “Intelligent Power Management over large Clusters,” 2010.

[28] D. Spence, “Reading Campus Grid User Guide.” http://www.reading.ac.uk/nmsruntime/saveasdialog.aspx?lID=36146&sID=89514, Oct. 2009.

[29] “Campus Grid-University of Reading.” http://www.reading.ac.uk/internal/its/e-research/its-eresearch-campusgrid.aspx.

[30] B. Cregan, “Condor.” https://www.acrc.bris.ac.uk/condor.htm, Dec. 2009.

[31] D. Spence, M. Tiejun, X. Xin, and W. David, “12_00_Spence-OxGrid-EU-Condor.ppt,” Oct. 2008.

[32] “CamGrid.” http://www.ucs.cam.ac.uk/scientific/camgrid, 2012.

[33] “Technical Details.” http://www.ucs.cam.ac.uk/scientific/camgrid/technical, 2012.

[34] “Condor.” http://www.cardiff.ac.uk/insrv/it/condor/index.html.

[35] J. Osborne and A. Hardisty, “Cardiff universitys condor pool: Background, case studies, and fEC,” in *Proc. AHM*, p. 361364, 2006.

[36] “Cardiff university | NGS.” http://www.ngs.ac.uk/institutes/cardiff.

[37] J. Huang, A. Kini, E. Paulson, C. Reilly, E. Robinson, S. Shankar, L. Shrinivas, D. DeWitt, and J. Naughton, “An overview of quill: A passive operational data logging system for condor,” *Computer Sciences Technical Report, University of Wisconsin*, 2007.

[38] “Pools of virtual boxes - browse /povb-2.0.1 at SourceForge.net.” http://sourceforge.net/projects/poolsofvirtualb/files/povb-2.0.1/.

[39] I. Smith, “Re: [Condor-users] condor with power saving PCs,” July 2007.

[40] I. C. Smith, “Towards a greener Condor pool: adapting Condor for use with energy-efficient PCs,” 2010.

[41] Microsoft, “Net start.” http://technet.microsoft.com/en-us/library/bb490713.aspx, 2013.

[42] “HTCondorWiki: whole machine slots.” https://htcondor-wiki.cs.wisc.edu/index.cgi/wiki?p=WholeMachineSlots, Nov. 2012.

[43] D. Gubb, H. Violeta, and I. Kureshi, “Implementing a condor pool using a green-IT policy.” http://digital-research.oerc.ox.ac.uk/papers/implementing-a-condor-pool-using-a-green-it-policy/view, Sept. 2012.

Appendix A

Modifications made to Windows condor_config

```
##Part 1

##  What machine is your central manager?
CONDOR_HOST=10.71.88.84

##Part 2
ALLOW_READ=*

ALLOW_WRITE=*

##Part 3

## Enable Condor to go into low power states
TimeToWait = (30 * $(MINUTE)
Should Hibernate = ( (KeyboardIdle >$(StartIdleTime))
    \
&& $(CPUIdle) \
&& ($StateTimer) >$TimeToWait)) )
```

Appendix B

[mac.pl] Automation of MAC and IP Collection

```
#!/usr/bin/perl

#get the Hostnames First
@host=‘condor_status -long |grep 'Machine =' |uniq |cut
   -c 12-40|cut -f1 -d"."‘;

#find the ip and mac addresses
@ipadd=(‘condor_status -long|grep 'StartdIpAddr =' |
   uniq| cut -f1 -d":"|cut -c 18-40‘);
@macadd=(‘condor_status -long|grep HardwareAddress|
   uniq|cut -c 20-36‘);

#Find number of machines
my $end=‘condor_status -long |grep 'Machine =' |uniq |wc
    -l ‘;

#i=$RANDOM;
#RANGE=$end;

#let "i %= $RANGE";

#open the file for WOL info
open (MYFILE, '>>WOLinfo.txt');

#removes next line from each array element
```

```
chomp @host;
chomp @ipadd;
chomp @macadd;

#fills the file with the required informatio
for ($count = $end; $count >= 0; $count--)
{
print MYFILE "$host[$count] $ipadd[$count] $macadd[
   $count]\n";
}

#closes file
close(MYFILE);

#checks that there are not any duplicate ip's or mac
   addresses
system('cat WOLinfo.txt|sort|uniq>WOLinfo.txt');
```

Appendix C

Condor wake on script from Liverpool

```
_____ cron job perl script
    _____

#!/usr/local/bin/perl

use strict;

my $condor_bin = '/opt1/condor/bin';
my $queue_args = '-constraint"JobStatus==1"-f"%d\\n"
    clusterid';
my $condor_q = "$condor_bin/condor_q␣$queue_args";
my $status_args = '-constraint\'State=="Unclaimed"\'-f
    "%s\\n"Name';
my $condor_status = "$condor_bin/
    condor_status$status_args";

my $get_idle_jobs      = "$condor_q|wc-l|tr-d''";
my $get_idle_machines = "$condor_status|wc-l|tr-d''";

my $no_of_idle_jobs;
my $IP_address;
my $MAC_address_file;
my $no_of_idle_machines;
my $email_file;
my $centre;
my $broadcast_address;
```

```
# actual broadcast addresses removed for security
   reasons

my %all_centres = ( "ROTC"=>"138.xxx.xxx.255",
                    "ARC2"=>"138.xxx.xxx.255",
                    "CDTC"=>"138.xxx.xxx.255",
                    "ERTC"=>"138.xxx.xxx.255" );

my $wakeup_root =  "/opt1/condor_wakeup";
my $wakeup = "$wakeup_root/wakeonlan ";
$email_file = "$wakeup_root/status";

$no_of_idle_jobs = `$get_idle_jobs`;
$no_of_idle_machines = `$get_idle_machines`;

open( EMAIL, ">", $email_file );
print EMAIL "no of idle jobs = $no_of_idle_jobs";
print EMAIL "no of idle machines =
   $no_of_idle_machines";
close( EMAIL );

if( $no_of_idle_jobs - $no_of_idle_machines > 10 )
{
   # testing only
   # `/usr/bin/mailx -s "woke up Condor pool"
      asdasuyuy\@liv.ac.uk < $email_file`;

   while( ( $centre, $broadcast_address ) = each %
      all_centres )
   {
     $MAC_address_file = "$wakeup_root/MAC_addresses/"
         . $centre;
     print "$wakeup -i $broadcast_address -f
        $MAC_address_file\n";
     print `$wakeup -i $IP_address -f
        $MAC_address_file`;
   }
}
```

[39]

Appendix D

Source code condor power v1

```
#!/usr/bin/perl

use strict;


my $condor_bin = '/opt/condor-7.6.7/bin';

my $queue_args = '-constraint "JobStatus==1" -f "%d\\n
    "clusterid ';
my $queue_win = '-long -constraint "JobStatus==1" |
    grep WINNT';
my $queue_lin = '-long -constraint "JobStatus==1" |
    grep LINUX';

my $condor_q = "$condor_bin/condor_q $queue_args";
my $condor_q_win = "$condor_bin/condor_q $queue_win";
my $condor_q_lin = "$condor_bin/condor_q $queue_lin";

my $status_windows7 = '-constraint \'OpSys=="WINNT61"
    && State=="Unclaimed"\' -f "%s\\n" Name ';
my $status_windowsxp = '-constraint \'OpSys=="WINNT51"
    && State=="Unclaimed"\' -f "%s\\n" Name ';
my $status_linux = '-constraint \'OpSys=="LINUX" &&
    State=="Unclaimed"\' -f "%s\\n" Name ';
my $status_args = '-constraint \'State=="Unclaimed"\'
    -f "%s\\n"Name';
my $status_busy = '-constraint \'State=="Claimed"\' -f
```

```
    "%s\\n"
Name';

my $condor_status = "$condor_bin/condor_status
    $status_args";
my $condor_status_win7 = "$condor_bin/condor_status
$status_windows7";
my $condor_status_winxp = "$condor_bin/condor_status
$status_windowsxp";
my $condor_status_linux = "$condor_bin/condor_status
    $status_linux";
my $condor_status_busy = "$condor_bin/condor_status
    $status_busy";

my $get_idle_jobs = "$condor_q | wc -l | tr -d ' ' ";
#wc is count and -l is the number of lines
my $get_idle_jobs_win = "$condor_q_win | wc -l | tr -d
    ' ' ' ' ";
my $get_idle_jobs_lin = "$condor_q_lin | wc -l | tr -d
    ' ' ' ' ";

my $get_idle_win7 = "$condor_status_win7 | wc -l | tr
    -d ' ' ' ' ";
my $get_idle_machines = "$condor_status | wc -l | tr -
    d ' ' ' ' ";
my $get_idle_winxp = "$condor_status_winxp | wc -l |
    tr -d ' ' ' ' ";
my $get_idle_linux = "$condor_status_linux | wc -l |
    tr -d ' ' ' ' ";
my $get_busy = "$condor_status_busy | wc -l | tr -d ' '
    ' ' ";

my $no_of_idle_jobs;
my $no_of_win_jobs;
my $no_of_lin_jobs;

my $no_of_idle_machines;
my $no_of_idle_win7;
my $no_of_idle_winxp;
my $no_of_idle_linux;
my $no_of_busy_machines;
```

```
my $email_file;
my $required_machines;
my $total_windows_machines;

$no_of_idle_jobs         = `$get_idle_jobs`;
$no_of_win_jobs          = `$get_idle_jobs_win`;
$no_of_lin_jobs          = `$get_idle_jobs_lin`;
my $linjobs              = $no_of_lin_jobs;

$no_of_idle_machines     = `$get_idle_machines`;
$no_of_idle_win7         = `$get_idle_win7`;
$no_of_idle_winxp        = `$get_idle_winxp`;
$no_of_idle_linux        = `$get_idle_linux`;
$no_of_busy_machines     = `$get_busy`;

$total_windows_machines = $no_of_idle_win7 +
   $no_of_idle_winxp;
$required_machines = $no_of_idle_jobs -
   $no_of_idle_machines;
my $required_linux_machines = $no_of_lin_jobs -
   $no_of_idle_linux;
my $required_windows_machines = $no_of_win_jobs -
$total_windows_machines;

open( EMAIL, ">", $email_file );
print EMAIL "
   ------------------------------------------------\n";
print EMAIL "Number of busy machines =
   $no_of_busy_machines";
print EMAIL "Number of idle machines =
   $no_of_idle_machines";

print EMAIL "\nNumber of idle Win7 = $no_of_idle_win7"
   ;
print EMAIL "Number of idle WinXP = $no_of_idle_winxp"
   ;
print EMAIL "Number of required machines =
   $required_machines\n";

print EMAIL "\nNumber of idle jobs = $no_of_idle_jobs"
```

```
    ;
print EMAIL "\nNumber of idle Windows jobs =
    $no_of_win_jobs";
print EMAIL "Number of idle Windows machines =
$total_windows_machines\n";
print EMAIL "Number of Windows machines required =
$required_windows_machines";
print EMAIL "\n\nNumber of idle Linux jobs =
    $no_of_lin_jobs";
print EMAIL "Number of idle Linux machines =
    $no_of_idle_linux";
print EMAIL "Number of Linux machines required =
$required_linux_machines";
print EMAIL "\n
    ------------------------------------------------\n";
if ($required_machines >= '0')
{
print EMAIL "\n
    ################################################\
    n";
print EMAIL "#                                        
    #\n";
print EMAIL "# Number of required machines =
    $required_machines #\n";
print EMAIL "# Machines have been woken                
    #\n";
print EMAIL "#                                        
    #\n";
print EMAIL "
    ################################################\
    n";

if ($required_machines <= '0')
{
print EMAIL "\n##################################\n";
print EMAIL "#                                #\n";
print EMAIL "# Enough machines are online !!! #\n";
print EMAIL "#                                #\n";
print EMAIL "##################################\n";
}
close( EMAIL );
```

Appendix E

HTPC Settings

```
# we will double-allocate resources to overlapping
    slots
NUM_CPUS = $(DETECTED_CORES)*2
MEMORY = $(DETECTED_MEMORY)*2

# we will double-allocate resources to overlapping
    slots
# single-core slots get 1 core each
SLOT_TYPE_1 = cpus=1
NUM_SLOTS_TYPE_1 = $(DETECTED_CORES)

# whole-machine slot gets as many cores and RAM as the
     machine has
SLOT_TYPE_2 = cpus=$(DETECTED_CORES), mem=$(
   DETECTED_MEMORY)
NUM_SLOTS_TYPE_2 = 1

# Macro specifying the slot id of the whole-machine
    slot
# Example: on an 8-core machine, the whole-machine
    slot is 9.
WHOLE_MACHINE_SLOT = ($(DETECTED_CORES)+1)

# ClassAd attribute that is True/False depending on
    whether this slot is
# the whole-machine slot
CAN_RUN_WHOLE_MACHINE = SlotID == $(WHOLE_MACHINE_SLOT
```

```
    )
STARTD_EXPRS = $(STARTD_EXPRS) CAN_RUN_WHOLE_MACHINE

# advertise state of each slot as SlotX_State in
    ClassAds of all other slots
STARTD_SLOT_EXPRS = $(STARTD_SLOT_EXPRS) State

# Macro for referencing state of the whole-machine
    slot.
# Relies on eval(), which was added in Condor 7.3.2.
WHOLE_MACHINE_SLOT_STATE = eval(strcat("Slot",$(
    WHOLE_MACHINE_SLOT),"_State"))

# Macro that is true if any single-core slots are
    claimed
# WARNING: THERE MUST BE AN ENTRY FOR ALL SLOTS
# IN THE EXPRESSION BELOW.  If you have more slots,
    you must
# extend this expression to cover them.  If you have
    fewer
# slots, extra entries are harmless.
SINGLE_CORE_SLOTS_CLAIMED = \
   ($(WHOLE_MACHINE_SLOT_STATE) =?= "Claimed") < \
   (Slot1_State =?= "Claimed") + \
   (Slot2_State =?= "Claimed") + \
   (Slot3_State =?= "Claimed") + \
   (Slot4_State =?= "Claimed") + \
   (Slot5_State =?= "Claimed") + \
   (Slot6_State =?= "Claimed") + \
   (Slot7_State =?= "Claimed") + \
   (Slot8_State =?= "Claimed")

# Single-core jobs must run on single-core slots
START_SINGLE_CORE_JOB =  TARGET.RequiresWholeMachine
    =!= True && MY.CAN_RUN_WHOLE_MACHINE == False && $(
    WHOLE_MACHINE_SLOT_STATE) =!= "Claimed"

# Whole-machine jobs must run on the whole-machine
    slot
START_WHOLE_MACHINE_JOB = TARGET.RequiresWholeMachine
    =?= True && MY.CAN_RUN_WHOLE_MACHINE
```

```
START = ($(START)) && (($(START_SINGLE_CORE_JOB)) || (
    $(START_WHOLE_MACHINE_JOB)))

# Suspend the whole-machine job until single-core jobs
    finish.
SUSPEND = ($(SUSPEND)) || (MY.CAN_RUN_WHOLE_MACHINE &&
    ($(SINGLE_CORE_SLOTS_CLAIMED)))

CONTINUE = ($(SUSPEND)) =!= True

WANT_SUSPEND = ($(WANT_SUSPEND)) || ($(SUSPEND))

# In case group-quotas are being used, trim down the
    size
# of the "pie" to avoid double-counting.
GROUP_DYNAMIC_MACH_CONSTRAINT = CAN_RUN_WHOLE_MACHINE
    == False

#
    #-----------------------------------------------------------------------
```

Appendix F

Final Script

```
#!/usr/bin/perl

use strict;


my $condor_bin          = '/opt/condor-7.6.7/bin';

my $queue_args          = '-constraint "JobStatus==1"
    -f "%d\\n" clusterid ';
my $queue_win           = '-long -constraint "
    JobStatus==1" | grep WINNT';
my $queue_lin           = '-long -constraint "
    JobStatus==1" | grep LINUX';

my $condor_q            = "$condor_bin/condor_q
    $queue_args";
my $condor_q_win        = "$condor_bin/condor_q
    $queue_win";
my $condor_q_lin        = "$condor_bin/condor_q
    $queue_lin";

my $status_windows7     = '-constraint \'OpSys=="
    WINNT61" && State=="Unclaimed"\' -f "%s\\n" Name ';
my $status_windowsxp    = '-constraint \'OpSys=="
    WINNT51" && State=="Unclaimed"\' -f "%s\\n" Name ';
my $status_linux        = '-constraint \'OpSys=="LINUX
    " && State=="Unclaimed"\' -f "%s\\n" Name ';
```

```
my $status_args          = '-constraint \'State==" Unclaimed"\' -f "%s\\n" Name';
my $status_busy          = '-constraint \'State==" Claimed"\' -f "%s\\n" Name';

my $condor_status        = "$condor_bin/condor_status $status_args";
my $condor_status_win7   = "$condor_bin/condor_status $status_windows7";
my $condor_status_winxp  = "$condor_bin/condor_status $status_windowsxp";
my $condor_status_linux  = "$condor_bin/condor_status $status_linux";
my $condor_status_busy   = "$condor_bin/condor_status $status_busy";

my $get_idle_jobs        = "$condor_q | wc -l | tr -d ' ' "; #wc is count and -l is the number of lines
my $get_idle_jobs_win    = "$condor_q_win | wc -l | tr -d ' ' ";
my $get_idle_jobs_lin    = "$condor_q_lin | wc -l | tr -d ' ' ";

my $get_idle_win7        = "$condor_status_win7 | wc -l | tr -d ' ' ";
my $get_idle_machines    = "$condor_status | wc -l | tr -d ' ' ";
my $get_idle_winxp       = "$condor_status_winxp | wc -l | tr -d ' ' ";
my $get_idle_linux       = "$condor_status_linux | wc -l | tr -d ' ' ";
my $get_busy             = "$condor_status_busy | wc -l | tr -d ' ' ";

my $no_of_idle_jobs;
my $no_of_win_jobs;
my $no_of_lin_jobs;

my $no_of_idle_machines;
my $no_of_idle_win7;
my $no_of_idle_winxp;
```

```
my $no_of_idle_linux;
my $no_of_busy_machines;

my $IP_address;
my $MAC_address_file;
my $email_file;
my $centre;
my $broadcast_address;
my $test_mac ='00:13:D3:0F:06:49';
my $test_IP  ='10.71.88.108';
my $required_machines;
my $total_windows_machines;

my $wakeup_root = "/usr/bin/";
my $wakeup      = "$wakeup_root/wakeonlan ";
$email_file     = "/home/u0771649/power/status";

$no_of_idle_jobs        = `$get_idle_jobs`;
my $no_of_idle_jobs;
my $no_of_win_jobs;
my $no_of_lin_jobs;

my $no_of_idle_machines;
my $no_of_idle_win7;
my $no_of_idle_winxp;
my $no_of_idle_linux;
my $no_of_busy_machines;

my $IP_address;
my $MAC_address_file;
my $email_file;
my $centre;
my $broadcast_address;
my $test_mac ='00:13:D3:0F:06:49';
my $test_IP  ='10.71.88.108';
my $required_machines;
my $total_windows_machines;

my $wakeup_root = "/usr/bin/";
my $wakeup      = "$wakeup_root/wakeonlan ";
$email_file     = "/home/u0771649/power/status";
```

```
$no_of_idle_jobs          = `$get_idle_jobs`;
$no_of_win_jobs           = `$get_idle_jobs_win`;
$no_of_lin_jobs           = `$get_idle_jobs_lin`;
my $linjobs               = $no_of_lin_jobs;

$no_of_idle_machines      = `$get_idle_machines`;
$no_of_idle_win7          = `$get_idle_win7`;
$no_of_idle_winxp         = `$get_idle_winxp`;
$no_of_idle_linux         = `$get_idle_linux`;
$no_of_busy_machines      = `$get_busy`;

$total_windows_machines = $no_of_idle_win7 +
   $no_of_idle_winxp;
$required_machines = $no_of_idle_jobs -
   $no_of_idle_machines;
my $required_linux_machines = $no_of_lin_jobs -
   $no_of_idle_linux;
my $required_windows_machines = $no_of_win_jobs -
   $total_windows_machines;

open( EMAIL, ">", $email_file );
print EMAIL "
   ------------------------------------------------------\n";
print EMAIL "Number of busy machines =
   $no_of_busy_machines";
print EMAIL "Number of idle machines =
   $no_of_idle_machines";

print EMAIL "\nNumber of idle Win7 = $no_of_idle_win7"
   ;
print EMAIL "Number of idle WinXP = $no_of_idle_winxp"
   ;
print EMAIL "Number of required machines =
   $required_machines\n";

print EMAIL "\nNumber of idle jobs = $no_of_idle_jobs"
   ;
print EMAIL "\nNumber of idle Windows jobs =
   $no_of_win_jobs";
print EMAIL "Number of idle Windows machines =
```

```
    $total_windows_machines\n";
print EMAIL "Number of Windows machines required = 
    $required_windows_machines";
print EMAIL "\n\nNumber of idle Linux jobs = 
    $no_of_lin_jobs";
print EMAIL "Number of idle Linux machines = 
    $no_of_idle_linux";
print EMAIL "Number of Linux machines required = 
    $required_linux_machines";
print EMAIL "\n
    ------------------------------------------------------------\n";

#waking and service changing section
if ($required_machines >= '0')
{


#
    ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#

#       Finding which hosts are online, and if busy
                #
#
    ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#


#which Windows hosts are online
my @WINDOWS=scalar(`condor_status |grep 'WINNT'|grep '
    Idle'|cut -c 7-18|uniq|cut -f1 -d"."`);
#which Linux hosts are online
my @LINUX=scalar(`condor_status |grep 'LINUX'|grep '
    Idle'|cut -c 7-19|uniq|cut -f1 -d"."`);
#which machines are busy
my @BUSY= scalar(`condor_status |grep 'Claimed\|Owner'
    |cut -c 7-19|uniq|cut -f1 -d"."`);

#
    ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#

# Macthing which Window/Linux machines are online with
     the MAC #
```

```
# and IP info colllected from the WOLinfo.txt
                        #
#
   ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#

open(FILE, 'WOLinfo.txt');

while(<FILE>)
{
#Splits the line by White-Space
($host, $ip, $mac) = split(' ');

#populates a table for when windows machines are idle
   for WINDOWS -> POVB
$wfound= grep(/$host/, @WINDOWS);
$lfound= grep(/$host/, @LINUX);
$bfound= grep(/$host/, @BUSY);

#online Windows
if ($wfound == 1)
{
push (@WINhost,$host);
push (@WINip,$ip);
push (@WINmac,$mac);
}
elsif ($lfound == 1){
push (@LINhost,$host);
push (@LINip,$ip);
push (@LINmac,$mac);
}
#busy machines
#online linux
elsif ($bfound == 1){
push (@BUSINhost,$host);
push (@BUSip,$ip);
push (@BUSmac,$mac);
}
#offline machines
elsif($lfound == 0 && $wfound == 0 && $bfound == 0)
{
push (@OFFhost, $host);
```

```
push (@OFFip,$ip);
push (@OFFmac,$mac);
}

#contents WOLinfo.txt
push (@hostn,$host);
push (@ipadd,$ip);
push (@macadd,$mac);
}
close(FILE);

#
   ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#

# Changes services then if there are not enough
   machines idle    #
# or online wake up machines
                                                #
#
   ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#

if ($no_of_idle_win7>=4 && $no_of_lin_jobs>=4)
{
        my $winRANGE = scalar @WINip+1;
        my $LinWake = int($/no_of_lin_jobs5);

        my $WINvar;
        for ($WINvar = 0; $WINvar < $LinWake; $WINvar
           ++) {

                my $RANDOM = int(rand($winRANGE));
                  net rpc service stop -I $WINip[
                     $RANDOM] -U Adminstrator%$password
                       condor;
                  net rpc service start -I $WINip[
                     $RANDOM] -U Administrator%
                     $password povb;
        }
}
elsif($no_of_idle_linux>=20 && $no_of_win_jobs>=20)
{
```

```
        my $linRANGE = scalar @LINip+1;
        my $WINWake = int($no_of_win_jobs/5);

        my $LINvar;
        for ($LINvar = 0; $LINvar < $WINWake; $LINvar
           ++) {
                my $RANDOM = int(rand($linRANGE));

                  net rpc service stop -I $LINip[
                     $RANDOM] -U Adminstrator%$password
                       povb;
                  net rpc service start -I $LINip[
                     $RANDOM] -U Administrator%
                     $password condor;
                }
}
else
{
        my $offRANGE = scalar @OFFip+1;
        my $OFFWake = int(abs($required_machines/5));

        my $OFFvar;
        for ($OFFvar = 0; $OFFvar < $OFFWake; $OFFvar
           ++) {
                my $RANDOM = int(rand($offRANGE));

                system ("wakeonlan␣$OFFmac[$RANDOM]␣-i
                   ␣$OFFip[$RANDOM]␣");
}
}
#print EMAIL "\n
   ##################################################\
   n";
#print EMAIL "#
                                                    #\n
    ";
#print EMAIL "# Number of required machines =
   $required_machines #\n";
#print EMAIL "# Machines have been woken
                          #\n";
#print EMAIL "#
```

```
                                                                    #\n
    ";
#print EMAIL
    "#######################################################\
    n";



}
if ($required_machines <= '4')
{
print EMAIL "\n#################################\n";
print EMAIL "#                               #\n";
print EMAIL "# Enough machines are online !!! #\n";
print EMAIL "#                               #\n";
print EMAIL "#################################\n";
}
close( EMAIL );
```