PBStoHTCondor System for Campus Grids

John Brennan, Violeta Holmes, Stephen Bonner
HPC Research Group
The University of Huddersfield
Huddersfield, United Kingdom
Email: j.brennan@hud.ac.uk,
v.holmes@hud.ac.uk, s.bonner@hud.ac.uk

Ibad Kureshi
Institute of Advanced Research Computing
Durham University
Durham, United Kingdom
Email: ibad.kureshi@durham.ac.uk

***Abstract*—The campus grid architectures currently available are considered to be overly complex. We have focused on High Throughput Condor HTCondor as one of the most popular middlewares among UK universities, and are proposing a new system for unifying campus grid resources. This new system PBStoCondor is capable of interfacing with Linux based system within the campus grids, and automatically determining the best resource for a given job. The system does not require additional efforts from users and administrators of the campus grid resources. We have compared the real usage data and PBStoCondor system simulation data. The results show a close match. The proposed system will enable better utilization of campus grid resources, and will not require modification in users' workflows.**

Keywords—HTCondor; Campus grids; PBS; Torque

I. INTRODUCTION

Within many Higher Education and Research Institutions computer grids play an important role in making geographically distributed disparate resources accessible to the researchers and students. Research effort into grid computing has generated a number of grid middlewares, which are often integrated with underlying cluster computing middlewares and schedulers. There is increasing pressure for grid solutions to be able to interface with a wider range of end resource types However, grid middlewares are generally considered to be rather complex by users and administrators alike.

At the University of Huddersfield (UoH) the High Performance Computing (HPC) and High Throughput Computing (HTC) resources are unified in the QueensGate Grid (QGG) Campus Grid [1,2]. The QGG consists of a number of disparate resources with a gateway to external and internal resources. There is a common connection point for most systems, and on reaching this point users would branch out to their preferred HPC/HTC system. This multiple system configuration has a serious drawback - the system load balancing is almost impossible to achieve. In its present configuration, to better balance the systems would require action from the users. The users would have to investigate the state of each suitable resource before submitting a job to any of them. In some cases a user may not know that a particular resource is suitable for a job, and such resources would not be utilized. In addition, there are many different job schedulers deployed within the QGG which present further problems to a user when trying to access the resources. Consequently, additional time is required to train the users to get proficient in using a particular scheduler or batch system. Therefore, a solution is required to improve load balancing and utilization of existing resource without affecting users' preferred workflows. We have considered some current campus grids at UK universities and efforts to overcome challenges to deliver services tailored to their users, and identified that they could be improved.

In this paper we are presenting a novel PBStoCondor system design which will lead to better utilization of the resources in our campus grid QGG. The system will autonomously adjust the jobs submitted by the users, without users' intervention. Based on our research, it is evident that similar systems do not exist in currently deployed university grids.

Our aim was to create an innovative system which will allow use of "idle" computers in university library and departmental laboratories, integrated into a HTCondor pool and our campus grid, and facilitate the following:

- Submission of serial (single core) jobs to HTCondor pool, freeing the HPC resources for parallel jobs and better load balancing
- Minimize possible impact on end users
- Minimize requirement for additional administration time in user training on different job schedulers

Our work extends the existing solutions in UK campus grids and QGG UoH grid. The results of review of UK universities campus girds are presented in section II. The rest of the paper is organized as follows: Section III describes the QGG campus grid solution at the University of Huddersfield focusing the HTCondor component; in Section IV and V we present PBStoCondor system design, and evaluation of PBStoCondor simulation results, ending with the summary in section VI.

II. REVIEW OF CAMPUS GRIDS AT UK UNIVERSITIES

Initially we considered deployment of campus grids at UK universities using Condor middleware, Reading, Cambridge, Oxford and Manchester.

A. *The Univeristy of Reading Campus Grid*

The University of Reading grid is a HTCondor pool which is composed of around five hundred worker nodes. These worker nodes have Microsoft Windows and Linux binaries, allowing both to be run, without the need to have multi-boot systems. Since this is a single system it is not a true grid but a number of HTCondor pools with campus wide flocking enabled [3].

B. *University of Cambridge Campus Grid*

According to Calleja et al. [4] CamGrid, the computing grid at the University of Cambridge, is now primarily composed of HTCondor pools. There is a large centrally managed pool which is augmented by other flocked pools administered by individual schools within the university, allowing campus wide resource sharing.

While CamGrid has no central provision for a Portable Batch System (PBS) based cluster it is potentially supported through allowing HTCondor to submit jobs to PBS enabling the users to run jobs which require tightly coupled MPI networks. There are some issues with this system, one being that the PBS queue must be on the same machine as the condor schedd. This means that any PBS based resource which is added in this manner will not truly be an integral part of the CamGrid as a whole. The systems implemented in this way will only be accessible from a school controlled HTCondor head-node and not from the grid primary entry point which is controlled centrally (CamGrid and PBS University Computing Service, 2013). While HTCondor can natively run MPI type jobs it is far from ideal as systems in HTCondor pools are not very tightly coupled. Considering this along with the fact that HTCondor requires dedicated resources for MPI universe jobs and cannot flock MPI jobs between pools, it highlights a significant weakness within the deployment [5].

C. *University of Oxford Campus Grid*

The researchers at University of Oxford recognized almost a decade ago that with individual schools purchasing resources to fulfill their own needs, it was very likely that a situation would arise where some departments may have excess resources, whereas others may struggle to complete their work on smaller resources. Consequently, the decision was taken to implement campus wide resource sharing through a common middleware. This system was to provide data storage along with computational resources. Built around four main components, an information server, resource broker, VO manager and a data vault, the system became known as OxGrid [6].

OxGrid provides a single ’control system’ which is the gatekeeper, that all users of the system, internal or external to the university, must connect through. This control system is built around a Virtual Data Toolkit (VDT) software stack utilizing a Condor-G based Resource Broker (RB) and encompassing Virtual Organization (VO) support. Through this the users, with personal X-509 certificates, are able to utilize PBS, Sun Grid Engine and HTCondor through a single submission interface.In order for this system to work all resource endpoints must advertise their capabilities and supported VOs to an information server. When a user submits a job, the RB must first check a users VO membership with a Virtual Organization Management System (VOMS) server. Then it checks which resources that particular VO is able to use, at which point a job can actually be submitted to one of the short listed endpoints utilizing the GT [6].

D. *Manchester University Campus Grid*

The University of Manchester (UoM) describes clusters interconnected through the use of GT as ’super-clusters’ which can then be considered as a single shared resource [7,8]. The UoM maintains a total of 69 nodes with 2.5TB of storage, all made available to researchers through the GT middleware. The GT GSISSH is used as the primary access mechanism for this system, therefore all users must be in possession of a valid X-509 certificate. The underlying middleware is GT and the UoM supports only internal access to these systems.

Table 1 shows a comparison of the university campus grids examined. It is evident that the HTCondor is the middleware of choice for many, as it offers a relatively simple scaleable resource. HTCondor middleware is open source and is actively maintained, which makes this choice a very cost effective solution.

E. *HTCondor*

HTCondor is a middleware which seeks to utilize CPU cycles which would otherwise be ’wasted’ within an environment where there are many standard desktop computers. Many large institutions provide a large number of workstations to be used for general day to day tasks. A significant number of time these computers are left idle, leaving an expensive resource going to waste.

HTCondor runs a service/daemon on Microsoft Windows or *NIX based systems which advertises itself as a worker node to a management server. This management server, or head node, accepts jobs from a user and then forwards it on to a suitable idle worker node for execution, employing underused resources to solve computational problems. There are number of potential drawback with this system. One of them is potential loss of data when the job execution in the HTCondor is interrupted by a conventional use of the machine. Where there is source code available for the software being used, HTCondor has the ability to compile an executable with specific ’hooks’ allowing checkpointing, meaning that the worker node can periodically report a jobs progress along with any data to a checkpointing server. If a worker node then has to ’dump’ that job then the next worker node can get the progress information from the checkpoint server and continue the job from the point that the last checkpoint save was made, leading to far fewer lost cycles and much better overall utilization of available systems. Another limitation of the HTCondor middleware is that although it is capable of running parallel jobs which require a Message Passing Interface (MPI) environment, to do this requires that a HTCondor pool has specially configured dedicated machines and cannot be achieved within the standard cycle stealing model [9].

TABLE I. CAMPUS GRID MIDDLEWARE COMPARISON

Campus Grid	*Middleware*	*Access*	*Current Support*	*Scale*	*Admin Support*	*Source*
Reading	HTCondor	Key	Y	Y	Med	Open
Cambridge	HTCondor	Key	Y	Y	Med	Open
Oxford	VDT	Cert	N	N	High	Open
Manchester	GT	Cert	Y	N	High	Open

Within the HTCondor ecosystem there is another significant shortcoming. Most institutional general purpose computers with Microsoft Windows based operating systems installed, whereas a large degree of computational science requires *NIX type systems. This limitation is addressed by the Pool Of Virtual Boxes (POVB) project. POVB can be installed on a Microsoft Windows based host. During this process a Linux based VirtualBox Virtual Machine (VM) is created which has HTCondor installed. The created virtual machine is then controlled by POVB passing required input to the virtualised HTCondor instance. This configuration creates a Windows host that advertises itself to the HTCondor head node as a Linux resource and can process any Linux based software. The VM within POVB is always started at system boot time and controlled dynamically. When a user is physically present and using the machine the VM is scaled back, in terms of memory and Central Processing Unit (CPU) usage, as far as is possible to minimise any impact of this system on the user. Once the system has been idle for a predetermined period of time then the VM is 'woken up' and the resources available to it are increased to allow for maximum utilization of the resource [10].

F. PBS/Torque

The PBS project was initially started in the early 1990's and developed by Veridian Information Solutions, funded by National Aeronautics and Space Administration (NASA). Not long after the initial development, Veridian released a commercial version of the software called PBSPro. Following this the open source version came to be called OpenPBS. However, withinin three years the project was acquired by Altair Engineering who discontinued development of OpenPBS. At this point, development of the open source product was taken over by Adaptive Computing who, having no rights to the PBS brand, changed the name of the software to Torque. Since then the status quo has been maintained with the paid for commercial version remaining PBSPro and the open source version being Torque [11].

PBS and Torque consist of three distinct parts, *pbs_server*, *pbs_sched* and *pbs_mom*. There is one *pbs_mom* for each compute node within a system, all of which communicate back to a single *pbs_server* instance. The *pbs_server* makes a queue for requested jobs and delegates them to the *pbs_moms*, along with collecting job related information such as time taken etc. The bundled scheduler, *pbs_sched*, is an interchangeable part of the system that can be replaced with alternatives such as Maui, Moab or even a custom scheduler. The purpose of the scheduler is to decide which jobs should run next, depending on which resources are available. While PBSPro is capable of running under Microsoft Windows as well as with *NIX based systems, this capability has not been extended to Torque, which means that Torque is unsuitable for any software not built against a *NIX based system [12].

Based on our review of existing solutions we envisaged a new PBStoCondor system that will overcome the issues outlined above. The proposed system will aid better utilisation of cluster resources in our QGG campus grid.

III. QGG UNIVERSTIY OF HUDDERSFIELD CAMPUS GRID

The QGG campus grid consists of a number of computer clusters linked via the university network. Table II gives an overview of the systems which are currently available at the UoH.

TABLE II. QGG CAMPUS GRID

Compute Element	*OS*	*Cluster Midd*	*Grid enabled*	*Numb of cores*	*Cores per node*
Taucety	CentOS 6	Torque/ Warewulf	Yes	4	4
Eridani	CentOS 6	Torque/ VDT	Yes	37	4
Sol	CentOS 6	Torque/ Warewulf	Yes	64	4
Condor	CentOS 6	HTCondor	Yes	2000	4-8
Bellatrix	CentOS 6	VDT	Yes	N/A	N/A

As can be seen in Table II the systems in the QGG use Community ENTerprise Operating System (CentOS) which is essentially a re-branded clone of Red Hat (RHEL) along with a Windows Enterprise Linux based Graphical Processing Unit (GPU) cluster. The High Throughput Condor (HTCondor) system is not exclusively a Linux based system, as while the head-node runs CentOS 6 the compute nodes are a mixture of Windows and Linux systems.

The largest system in the QGG, by node and Central Processing Unit (CPU) count, is the HTCondor system. This system is a truly heterogeneous pool utilizing the resources from different hardware configurations and software stacks. Within the HTCondor pool there are around 2000 Windows based systems, each having between 4 and 8 slots depending on number of CPU cores in the individual system. These systems are located in the University's computer laboratories, and on the computers provided by Computing & Library Services. In addition to these resources there are around 140 systems which have Pool Of Virtual Boxes (POVB) installed, allowing Linux based jobs to be run on the system.

As shown in Figure 1. QGG Users can connect to the QGG HTCondor head node and submit all jobs from there. HTCondor defines jobs in subsets, or as the middleware defines it - a job 'universe'. The grid universe is available, allowing jobs to be submitted to any resource which has a Globus gatekeeper. Therefore all job submission can be achieved using a single Job Description Language (JDL).

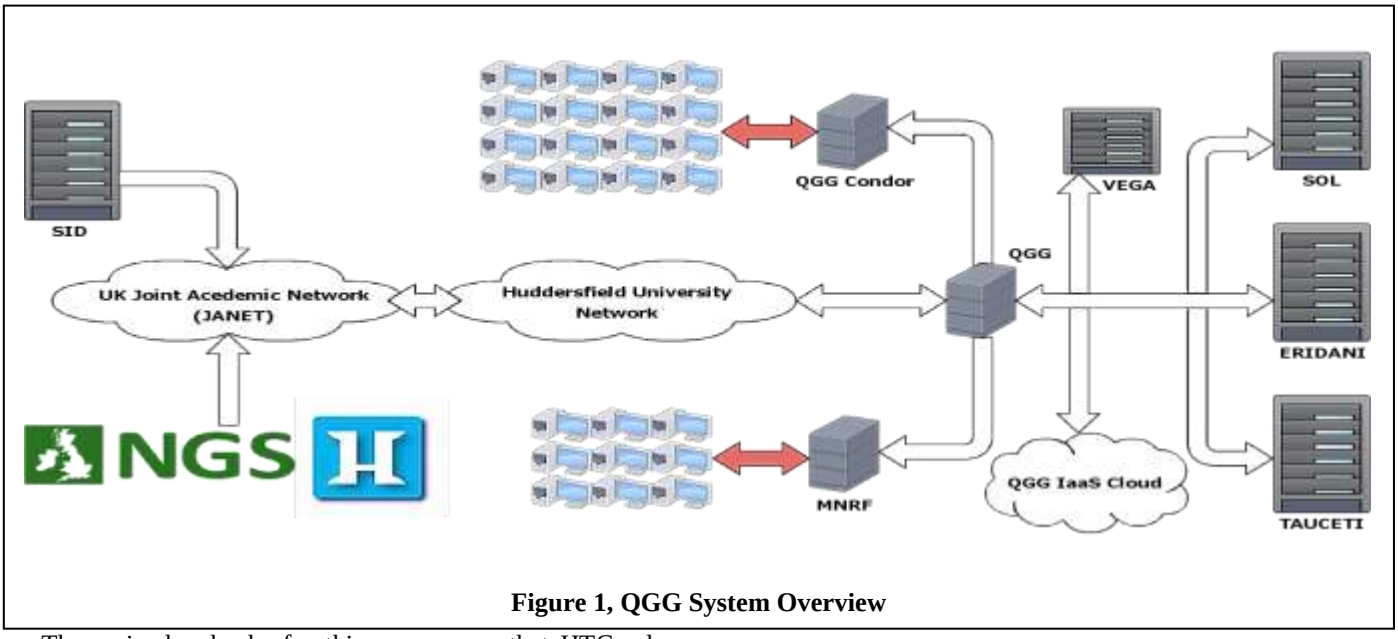

Figure 1, QGG System Overview

The main drawbacks for this process are that HTCondor has no real mechanism for automated discovery of grid resources. Also, the users are reluctant to move away from the familiar Torque resource manager based submission. Without a resource discovery mechanism, the users must specify an endpoint they would like to use in the JDL, which would require prior knowledge about the resources. The users can easily make a direct SSH connection to the known system instead of via the HTCondor head node.

The real power of HTCondor lies within the vanilla and standard universe models. Within the vanilla universe HTCondor will choose suitable resources from the pool of discovered resources based on the requirements of the job. The standard universe is similar to vanilla, but has the added functionality of job check-pointing. This allows binaries that have been compiled with Condor compile to periodically checkpoint their progress. Progress information created in this way is sent to a checkpoint server, which can subsequently be used to restart the job at that point if it fails for any reason, enabling HTCondor to deliver an extremely resilient resource.

Where multiple resources are available, load balancing becomes very difficult as users have a tendency to consistently use the resource they are most familiar with. For many reasons, such as time, cost, etc, it is always preferential to balance load across all suitable resources. One approach address this issue would be to force all users into a particular batch system, such as High Throughput Condor (HTCondor), which is capable of submitting to all resources. This would impact greatly on the users and was considered inappropriate.

Therefore a submission system based on the users' 'favourite' batch system needed to be devised. Such a system needs to provide a submission mechanism which behaves, from a user perspective exactly the same way as Torque, while allowing the administrators to control load balance by sending jobs to an appropriate resource. Another important aspect, from an administration perspective, was centralization of all accounting records.

In order to achieve better load balancing of existing systems without impacting on users' preferred workflows, PBStoCondor system was developed.

IV. PBStoCondor System

The PBStoCondor System was designed and implemented in Python, as the problem to be solved was essentially a scriptable problem but beyond capabilities of a standard shell interpreter.

The first problem to overcome was enabling Torque to recognise the Pool Of Virtual Boxes (POVB) based resources within the HTCondor pool, as potential compute nodes. It was determined very early in the project that it would not be possible to simply run a pbs mom (executor demon running in a background) on each of these nodes, as such an arrangement would be placing jobs on HTCondor resources without any of the HTCondor services being aware of them. Therefore all jobs needed to be run through HTCondor but monitored by Torque.

From Torque version 3.0 it has been possible to run multiple moms on a single node. Through observation of running moms it was found that an average mom uses minimal (<0.1%) Central Processing Unit (CPU) time and around 30MB of memory. The HTCondor head-node in the QGG had a total of 15.5GB of memory of which 13.8GB was free with all required processes running at any given time. This means that the system was theoretically capable of running 471 moms, 235% over the potentially required 200 moms.

A. Initial PBStoCondor system

The most intuitive method for allowing Torque to choose POVB nodes was simply through queue management. Torque was configured to send all jobs which requested a serial queue, and a short enough wall-time, to the HTCondor based moms, where wall-time is the total amount of real time requested by a user to complete a job. The reason for using wall-time in this way was that common installations of applications were being used across all systems, not applications compiles specifically for HTCondor. Meaning jobs could not make use of HTCondor checkpointing mechanisms and henceforth were limited to the vanilla universe. Therefore it was undesirable to have very long running jobs submitted to HTCondor resources.

On the HTCondor head-node, the symlinks pointed to a script which would generate a job submission script suitable for HTCondor and subsequently submit it to the system. This script was also required to monitor the progress of the job and

modify Torque accounting records to maintain accuracy of the accounting across all systems. The basic workflow of this system is quite simple, as described in Figure 2.

Although this system worked, it relied on Torque moms being manually started on the HTCondor head-node. Given the dynamic nature of HTCondor this was not a feasible solution. To address this issue a daemon was developed to monitor the available resources and start moms accordingly.

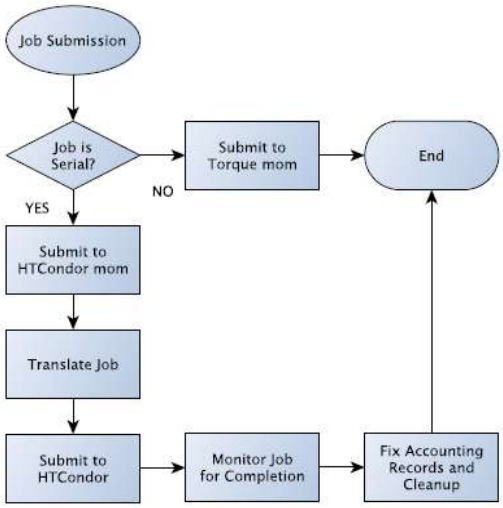

Figure 2, PBStoCondor System Overview

B. PBStoCondor Deamon

In Figure 3 the functionality of the developed daemon is shown. The daemon is started as a standard *NIX system service, or daemon, which will fail if the configuration file is missing. The initial startup procedure creates a table of possible moms, including name and required ports, from the variables found within the configuration file. Once the program is running in the background, it cycles through the while loop once every 30 seconds. This was done in order to prevent the daemon from unnecessarily using CPU cycles. The daemon queries the HTCondor pool to discover the current number of POVB nodes which are in an 'unclaimed' state. If this number is greater than the number of moms currently running, then an appropriate number of moms are started. If the number of 'unclaimed' nodes is less than the number of moms that are running then a suitable number of moms are killed. When this situation is true, the daemon checks that the mom to be killed has no child processes. This ensures that a mom which is currently processing a job will not be killed by the daemon. If the mom currently being evaluated has child processes the daemon will then consider the next mom within the array to determine a suitable mom to stop. Every time the daemon starts or stops a process, the internal data array is updated to ensure there is always valid information of which moms are running for the system to query. If at any point the daemon receives a termination signal it will end all running moms before exiting. This will ensure there are no orphaned processes on the system. PBStoCondor handles all short serial jobs submitted to a Torque batch system which can be pushed to a more appropriate HTCondor resource.

This system allows Torque to delegate particular jobs to a HTCondor system. Most importantly, this added functionality allows for better use of more tightly coupled resources while maintaining a consistent, and familiar, submission method to users. However, there was also a requirement to allow the system to provision for jobs which required resources greater than those currently available. This will be considered in future publications.

V. PBStoCondor System Evaluation

In order to test PBStoCondor further development was required to gain real comparative data. This was developed in the form of a simulator to emulate the system modifications which would have been provided by a PBStoCondor deployment within the QGG. The research work on the simulator will be presented in future publications. The data produced by this system was then analysed.

There was the opportunity to test this system by using historic Torque accounting logs. The aim of which was to highlight improved overall throughput by automatically pushing all serial jobs to HTCondor. The test strategy was based around one year of Torque accounting logs taken from the Eridani cluster in QGG. Torque logs are organized as a single file for each day as they are produced. Using these files a single file of all logs from 2013 was produced, which is referred to as the original Torque log. Once simulated log files had been produced, those logs, along with the original Torque logs, were analyzed to extract useful data, such as jobs arrival and completion rates. This allowed for direct comparison of simulated and real system performance as seen in figure 4.

The final results obtained from comparison of PBStoCondor simulation with historical Eridani cluster logs were very encouraging. These results showed that PBStoCondor provided a significant improvement in job completion times over a period of one year. This improvement was particularly apparent when the system had very large numbers of serial jobs submitted. Had this system been deployed at the beginning of the testing period, then there would have been 306 hours, almost 13 days, where the system would have been idle, compared to still working on completing jobs when only Eridani was being used. These periods could represent significant power savings or allow users to submit more jobs, thus making far better use of existing resources.

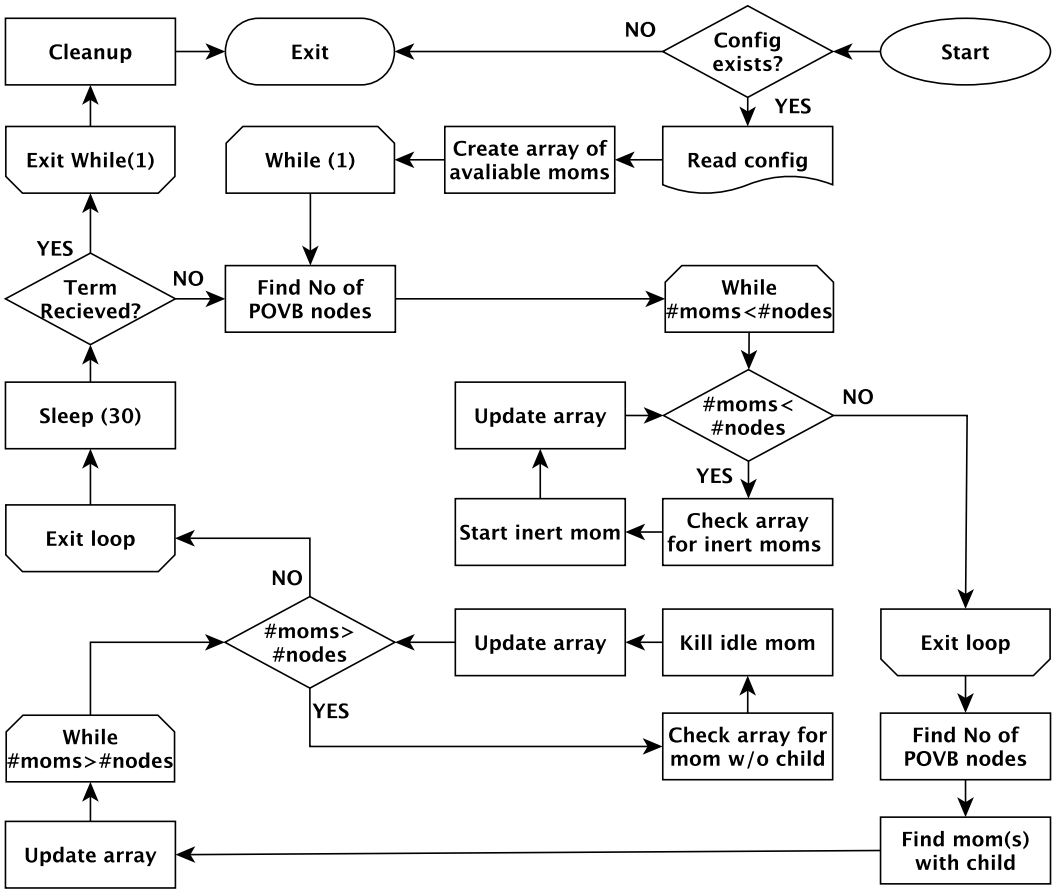

Figure 3, Flowchart for the Daemon

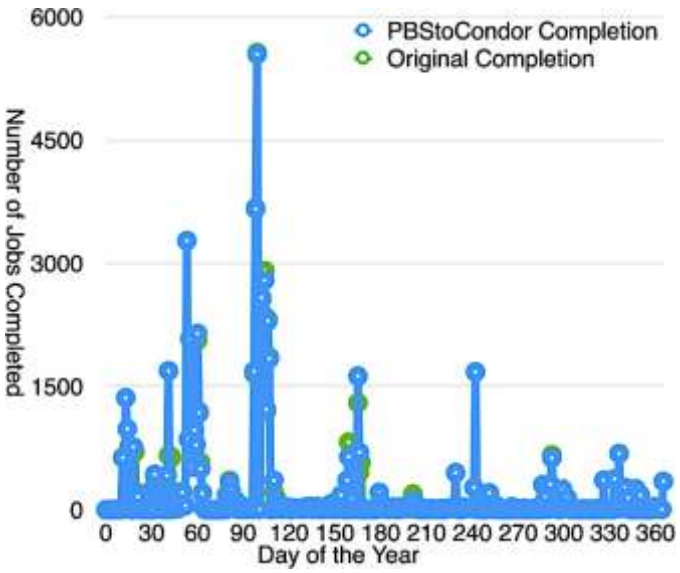

Figure 4, Comparison of Original logs and PBStoCondor job completions rates

VI. Conclusion

This research was focused on making the best use of institutional computational resources and improve load balancing whilst maintaining the user's existing workflows. The existing Grid and Cluster middlewares were investigated as well as their deployment in other UK HE and research institutions. These investigations found a mixture of High Performance Computing (HPC) and High Throughput Computing (HTC) systems. The High Throughput Condor (HTCondor) system was a system of choice for many universities.

HTCondor is also deployed in the QGG campus grid. Within the QGG there is a very large HTCondor pool (over 2000 nodes) which was largely under-utilised. The users within the QGG had displayed a reluctance to move between Torque and HTCondor systems. The campus grid administrators observed that most users remained exclusively on Torque based systems regardless of the type of job being run.

A novel system PBStoCondor was designed in attempt to address this problem, make utilization of resources more efficient and suitable for the jobs submitted. In particular, it should improve campus grid user experience. The developed systems is designed to be simple to deploy. The developed software allowed users to maintain use of the more familiar Torque based submission, seamlessly pushing suitable jobs to the HTCondor pool, thus allowing the overall grid to complete computational processes much quicker. This resulted in better

load balancing of the systems. Consequently this will also potentially provide some power savings on the high power consumption clusters.

PBStoCondor has been bundled into a Red Hat Package Manager (RPM) package to allow for rapid deployment on any system which uses RPM packages.

ACKNOWLEDGMENT

The authors would like to acknowledge the use of the University of Huddersfield QGG Campus Grid.

REFERENCES

[1] V. Holmes, and I. Kureshi, "'Creating an HE ICT Infrastructure Fit for the 21st Century'. In: Higher Education Show 2013, 25 April 2013, London, UK.

[2] J. Brennan, "Developing Trusted Computational Grid", MSc Thesis, The University of Huddersfield, April 2014.

[3] University of Reading. (2013). Campus grid. https://www.reading.ac.uk/internal/its/e-research/its-eresearchcampusgrid.aspx. Retrieved 2013-11-01

[4] M. Calleja, et al, "Camgrid: Experiences in constructing a university-wide, condor-based gridat the university of cambridge", 2008.

[5] D. Thain, at al. , "Distributed computing in practice: the condor experience. Concurrency and Computation: Practice and Experience, 17(24), 323-356, 2005.

[6] D.C. Wallom, and A.E. Trefethen, "OxGrid, a campus grid for the university of Oxford". In Proceedings of the UK e-science all hands meeting, 2006.

[7] University of Manchester, "Access to computational grid systems at UoM made easy", 2013.

[8] R. Allan, "Infrastructure - northwest grid", 2011. Retrieved 2013-11-15, from http://www.nw-grid.ac.uk/Infrastructure

[9] D. Thain, T. Tannenbaum, and M. Livny, "Distributed computing in practice:the condor experience. Concurrency and Computation: Practice and Experience, 17(24), 323-356, 2005

[10] D.J.Herzfeld, L.E. Olson, and C.A. Struble, " Pools of virtual boxes:Building campus grids with virtual machines. In Proceedings of the 19th acm international symposium on high performance distributed computing (pp.667–675). New York, NY, USA: ACM, 2010.

[11] C Samuel, "Torque history", 2008 Retrieved 20-09-2013, from http://www.supercluster.org/pipermail/torqueusers/2008-Febuary/006827.html

[12] D Beer, "torque for windows", 2008 Retrieved 20-09-2013, from http://www.supercluster.org/pipermail/torqueusers/2012-April/014467.html