Experiences with Running MATLAB Applications on a Power-Saving Condor Pool

Ian C. Smith

University of Liverpool Computing Services Department
i.c.smith@liverpool.ac.uk

September 11, 2009

Abstract

The University of Liverpool Computing Services Department (CSD) Condor Pool has been developed over a period of four years and now comprises approximately 300 classroom PCs (giving around 600 job slots) which run the University's Managed Windows (XP) Service. Many of our Condor users have eschewed writing their applications in traditional programming languages such as C and FORTRAN in favour of using MATLAB M-files compiled into "standalone" executables. Applications have varied widely from the simulation of avian influenza transmission in poultry flocks to the modelling of the biological effects of radiation on normal tissue in radiotherapy research. Whilst MATLAB provides users with a convenient environment for rapid development of application codes, there are a number of hurdles to overcome in running MATLAB applications successfully under Condor. MATLAB provides some unique benefits over traditional programming languages and both problems and merits are discussed here. Since 2006, we have adopted a pro-active strategy to reducing IT hardware energy consumption through the adoption of power-saving on PCs across campus. This has required us to adapt the Condor pool so that, firstly, Condor jobs are not terminated early by PCs hibernating and, secondly, that power-saving PCs can be made available on-demand to run Condor jobs. The approach taken to both issues is described here.

1 Introduction

The University of Liverpool Computing Services Department (CSD) Condor Pool was first established as an experimental service around four years ago and has expanded steadily to a point now where there are around 600 job slots available for use by Condor users. The pool consists entirely of classroom PCs, distributed across the campus, which are available for general use by students and staff. Most machines in the pool are Dell PCs with Intel Core 2 (dual core) processors running at 2.33 GHz. There is 2 GB of RAM and around 80 GB of disk space on each PC.

Although there are around 2000 PCs available in total across the University, we have deliberately chosen only those with the highest specifications for use in the pool so that the pool is essentially homogenous with regard to machine performance (there are good reasons for this which are discussed later). All of the PCs run the CSD Managed Windows Service which is based on Windows XP Service Pack 3. Application changes and patches are generally applied via weekly re-imaging although there is scope for implementing changes automatically when machines are rebooted.

The policy implemented on our Condor pool is to only run jobs during office hours if there has been no keyboard or mouse activity for at least 10 minutes and if the net load average is low (< 0.3). Outside of office hours, jobs are allowed to run without restriction since users cannot physically access the machines at these times. Should a user login to a PC running a Condor job, our policy is to kill the job immediately rather than suspending it. All of the dual core machines in the pool are configured with two job slots in order to give better energy efficiency although this is at the expense of available memory per job.

The name of the classroom (referred to as "teaching centre" here) in which each PC is located appears in its hostname, for example ETC1-01.livad.liv.ac.uk refers to a PC in Engineering Teaching Centre 1. This means that the teaching centre name will appear in the Name and Machine attributes

of the machine ClassAds making it easy to identify machines belonging to a particular teaching centre. The teaching centre name is also included in a custom machine ClassAd. This configuration is useful in identifying which hibernating PCs are to be woken up and is discussed in the section on power saving later. The number of PCs in each teaching centre varies from around twenty to sixty.

All jobs are submitted to the Condor pool through a single combined central manager and submit host. Although there are known scaling problems with this, the extra security afforded by a single access point was an overriding consideration. The central manager runs on a Sun V440 SMP machine with 16 GB RAM and a 1.2 TB RAID filestore used exclusively by Condor. The operating system is Solaris 10. Condor users login to this machine via a restricted access shell secured through the main University authentication system. There is also a web interface for some specific applications used in computational chemistry research (namely, GAMESS and PC-GAMESS).

Condor version 7.0.2 is currently used on the central manager and all of the execute hosts although we aim to move to 7.2.x shortly. SSL authentication is used to secure communication between the daemons running on the central manager and the execute hosts and filesystem authentication is used for interactions between daemons and Condor users. The Sun server also acts as a Condor View host and a job submission access point to our campus grid (UL-GRID) via Condor-G.

2 Running Concurrent MATLAB Jobs under Condor

MATLAB is an interactive matrix manipulation program, developed by MathWorks [1], with built-in functions to perform operations such as matrix inversion, transposition, factorisation and the calculation of eigenvalue solutions. It has its own high-level language in which commands can be entered line by line into an interpreter or encapsulated into functions stored as scripts. These are known in MATLAB parlance as M-files. This environment makes it easy to develop code quickly and the powerful matrix/vector syntax encourages an algorithmic approach to problem solving at an early stage rather than users being overly-concerned with implementation issues.

A wide variety of built-in functions are available which are specific to different branches of mathematics, science and engineering. These are are grouped into so-called toolboxes. Toolboxes are available in areas as diverse as genetic algorithms and control system modelling, and their adoption can save users a considerable amount of time by not reinventing the wheel (indeed many users choose MATLAB purely because of this).

Another powerful feature of MATLAB is its built-in file I/O functions. To save all of their workspace, users can simply enter the command `save <filename>` and to load the workspace later, use the command `load <filename>`. If only a subset of variables from the workspace is needed, a list can be supplied as arguments to these commands. It is worth considering the amount of coding effort required to perform this in languages such as C or FORTRAN, particularly if large sparse matrices are involved.

The power and user-friendliness of MATLAB does come at price though and MATLAB programs may run considerably slower than equivalent applications written using C or FORTRAN. This is especially the case where there are many dynamic memory allocation function calls executed by MATLAB behind the scenes.

One way of speeding things up is to identify those parts of the MATLAB code which act as bottlenecks and to code them using C or FORTRAN. MATLAB has APIs for these languages and code can be compiled into dynamically linked subroutines (known as MEX files - Matlab EXecutables) under MATLAB and then accessed in the same manner as built-in functions. The API is quite complicated so that creating MEX files is not a trivial task and its C-like feel makes the use of FORTRAN quite challenging. It should be noted that this interface may also allow the user to employ high performance libraries such as BLAS, LAPACK etc and the performance increases are generally worth the coding effort involved.

In addition to MEX files, MATLAB provides another way of reducing run times which is to compile M-files into "standalone" binary executables. In fact multiple M-files can be linked together to create the executable (which can also access MEX file functions) making this a very flexible approach. MATLAB comes with its own compiler but it is also possible to use third-party compilers such as Microsoft Visual Studio. Almost any MATLAB function can be incorporated into a standalone executable so that the standalone applications can leverage functions from the MATLAB toolboxes. The standalone implementation has the added advantages that it can be run on a platform that does not have MATLAB installed and without the need for a license to be checked out each time it runs.

Many of our Condor users, seemingly with the above points in mind, have chosen to write their applications in MATLAB rather than say C or FORTRAN. There is also, most likely, a more pragmatic reason which is that many users do not have the time or inclination to learn these languages, much less write code in them. The fact that these MATLAB programs are generally less efficient than their equivalents written in C or FORTRAN may dismay some computer scientists and software engineers but if MATLAB can encourage more users to make use of Condor then it seems worth putting aside such concerns.

The applications that have made use of MATLAB at Liverpool have covered a variety of research projects including:

- prediction of the spread of H5N1 avian influenza in British poultry flocks [2] (Department of Veterinary Clinical Science);
- modelling the biological effects on normal tissue during radiotherapy using 3D voxel arrays representing organs (Department of Medical Imaging and Radiotherapy);
- modelling of E. Coli propagation in dairy cattle (Department of Veterinary Clinical Science);
- testing parallel genetic algorithms to optimize the parameters of a complex classification system (Department of Electrical Engineering and Electronics) and
- simulation of the infection of a bacterial cell by a virus (School of Mathematical Sciences)

In each case the researcher has compiled their M-file(s) into a standalone executable which is run on the Condor pool. Originally this was a necessity as only a limited number of licenses where available through our FlexLM licence server which would quickly have been exhausted by Condor jobs. We have since acquired a site license but have stuck with the standalone approach for reasons of efficiency.

The run times of the MATLAB Condor jobs vary enormously depending on the application from around 10 minutes to more than a week. In the later case we have used a simple Condor DAGMan workflow to automatically resubmit jobs after they have run for a few hours (usually this equates to a given number of cycles for iterative-type problems). The MATLAB `save` and `load` commands provide a simple way of implementing a user-level checkpoint from which jobs can be restarted.

We currently have the MATLAB R2008 release pre-installed on all of the machines in the Condor pool and will soon be moving to R2009. Originally MATLAB was installed on a network mapped drive accessible via Novell NetWare, however since moving from Novell to an all-Microsoft solution, jobs began to fail unpredictably when attempting to map the network drive. An important efficiency advantage is gained in having MATLAB pre-installed as the term "standalone" is somewhat of a misnomer. In reality standalone executables need access to the run-time MATLAB libraries known as the MATLAB Runtime Component (MCR). If MATLAB is not available on the target platform, the MCR needs to be installed first. MathWorks suggest doing this only once for each target platform and for good reason - the MCR is supplied in the form of a large (approx 150 MB) self extracting `.exe` file which can take a significant time to install [1]. In theory the MCR would need to be installed every time a MATLAB Condor job is run since *condor_preen* will remove all of the files each time the job terminates. For jobs lasting only around 10 minutes this clearly does not represent a scalable solution and having the MATLAB runtime libraries pre-installed as part of a complete MATLAB installation is clearly a boon.

The term standalone is also misleading in another way. Rather than consisting of a single monolithic executable file, the standalone in fact contains a number of different files which need to be packaged and deployed on the target platform. To run MATLAB jobs on our Condor pool, users therefore need to create a ZIP file containing all of the necessary files then extract this when running the job on the pool (this accomplished via a DOS batch file). This additional complication combined with the fact that our submit host uses a UNIX environment which is unfamiliar to some users can prove a significant hurdle for first-time Condor users. Many users however, run the same MATLAB application code repeatedly and only need to go through the build-package-upload process once. Early experiments with the R2009 MATLAB release suggest that this process may prove easier in the future.

Once users have submitted their code to the Condor pool, there are also potential run-time errors which may occur. These tend to be caused by not specifying the path to the run-time libraries correctly (or at all) and are extremely difficult to spot as MATLAB has a tendency to try to pop up error messages

[1] Simon Caton of Cardiff University's School of Computer Science has come up with novel way [3] of packaging only those toolboxes required by the standalone into the executable itself so that the MCR need not be installed.

on the Windows desktop rather than writing them to conventional stdout/stderr (where they can be captured by Condor). Clearly this is not very illuminating if a job is running on a remote classroom PC. The only way we have found to glean information from these error messages is to run the application under Condor on a local PC. If Condor is configured with `USE_VISIBLE_DESKTOP=True` on the PC, then the error message will appear on the local desktop and will hopefully help elucidate the problem.

A couple of other run-time problems have been seen although they tend to be fairly rare (on the order of 1 in 1000 jobs or less). Firstly, it is sometimes case that MATLAB jobs appear to start and run correctly but on termination do not write to the required output file. Examination of the log files reveals that Condor believes these jobs to have terminated normally and so it can be very difficult to identify the jobs that have failed in clusters of several thousand jobs or more. To deal with this problem, we therefore recommend (as per the Condor manual) that users initially omit the `transfer_output_files` attribute when testing but specify all of the required output files using `transfer_output_files` when submitting large numbers of "real" jobs. In this case, Condor will signal an error if any of the output files are absent and place the jobs in the held state. The jobs that have failed can then be identified and restarted easily using *condor_q -held* and *condor_release* respectively.

An even more rare run-time problem is that some jobs seem to continue for far longer than expected – sometimes days instead of hours. This is not generally an issue during term time as jobs tend to get evicted during the day as users login to the execute hosts. During vacations though, when the number of logins is low, these jobs can continue for several days. In this case, the *condor_evict* command is useful in displacing these long running jobs to different machines where they generally run to completion. It is interesting to note that there seems to be no pattern to this or the previous run-time problem and it seems to be almost impossible to reproduce (even when the same job is run with same input data on the same machine !).

3 Adapting Condor for use in a Power-Saving Environment

The Computing Services Department (CSD) has in the past three years adopted a pro-active strategy of reducing the energy consumption of the several thousand PCs running the CSD Managed Windows Service across campus. Most of these machines are left powered-on overnight, at weekends and through long vacation periods with negligible usage. Initially a policy of powering-off machines after 30 minutes of inactivity (provided no user is logged in) was adopted. More recently, machines are forced into hibernation if there has been no activity for at least 10 minutes. Hibernation was chosen over power-off as users can bring the machines back to full operating power more quickly if needed by briefly pressing the power button. In hibernating mode the memory contents are stored to disk (from where they can be quickly restored) and the power consumption drops almost to zero. By contrast, standby (otherwise known as "sleep") mode allows the machine to be woken very quickly but cuts consumption only by about a half.

To gauge the effectiveness of the policy, we have made use of one of the features PowerMAN power monitoring system from Data Synergy [4]. This comprises two main components a service running on the Windows PC and a Management Reporting Platform server. The Windows service detects PC usage (i.e. keyboard/mouse activity and system load) and can force the machine into a low-power state (hibernation in our case). It also acts as a client which reports PC activity to the PowerMAN server. The PowerMAN server collates activity data from the clients and makes this available in the form of web pages. The activity of all teaching centres can be summarised or individual centres (see figure 2) and it is also possible to "drill down" and examine the activity of individual machines on a hourly basis over arbitrary periods. This makes it easy to spot where machines are powered-up and inactive thus wasting energy. Through careful monitoring and tailoring of the power management policy it has been possible to remove around 200 000 - 250 000 hours of inactivity each week (resulting in an energy saving of around 20-25 MWh based on an average consumption of 100 W per machine). This has led to an estimated saving in electricity bills of around £124 000 per annum.

When the power-saving policy was first introduced we simply opted-out a number of centres so that they could run Condor at any time. Clearly this is not a scalable solution and in order expand the Condor service some way of allowing it to co-exist with power-saving execute hosts was necessary. The problem divides into two distinct parts: firstly, how to ensure that machines do not go into hibernation when running Condor jobs and secondly how to wake up hibernating PCs so that they can run Condor jobs.

Initially, a system process ran a DOS `.bat` file after 30 minutes of inactivity was detected which checked whether a user was logged in before powering-down the machine. Unfortunately, the account which owns a Condor job does not appear as an ordinary logged-in user and this was therefore unable to detect whether a Condor job was running. An additional test was needed to prevent jobs being terminated early and this was implemented by checking for the presence of `condor_exec.bat` in the temporary execute directory in which Condor jobs are started. This should be deleted as soon as a job terminates however files can sometimes be left in the directory are only removed later when the Condor garbage manager (*condor_preen*) runs again.

A more reliable approach was adopted when moving to the PowerMAN system as this can be configured with a list of "protected programs" that when running ensure that the system remains active. By making one of these programs the *condor_starter* process (which is only present whilst Condor jobs are running), it was possible to prevent hibernation whilst Condor jobs are running. In the six months since PowerMAN was introduced we have looked at reducing the inactivity time from the original 30 minutes to just 10. This has had some important repercussions on the wake up process and is discussed in more detail in the section on "Recent and Future Developments". To avoid overly-complicating matters here an inactivity time of 30 minutes is assumed.

When the machines go into hibernation, most components are powered down but the Network Interface Card (NIC) remains active (this is also true when machines are in a "powered-off" state). The NICs on all PCs in the pool have a "wake-on-LAN" capability which causes them to bring hibernating machines back to full operating mode on receipt of a series of so-called "magic packets" [2]. It is this functionality which is crucial to waking up machines in the pool according to the demand for job slots from the submit host. A cron job runs on the submit host / central manager and every 15 minutes checks the state of the Condor queue as well as that of the pool. If the number of idle jobs is found to be greater than the number of unclaimed hosts, then hibernating machines are woken up in order to attempt to satisfy the demand.

The machines are taken out of the hibernation by running a binary executable which sends the required magic packets to them. For this the broadcast address of the machine is needed and its hardware (MAC) address. The MAC addresses are stored in separate files sorted according to teaching centre and the cron script contains a list of the broadcast addresses for each of them. In this way machines are woken up one centre at a time rather than on an individual basis.

Originally the entire pool was woken up if there was a surfeit of idle jobs however, the cron script has recently been modified so that only the minimum number of teaching centres necessary are woken up. By parsing the output from *condor_status*, an estimate can be made of the number of hibernating machines in each centre. The list is sorted according to the number of hibernating machines and centres are woken up in sequence (from those with the highest number of hibernating machines to the lowest) until sufficient are woken up to satisfy the demand (or the entire pool has been woken up). Frequently, users submit large clusters of jobs which tend to saturate the pool so that this adaptive method is only rarely needed.

There was one other change needed for the automatic waking up of the pool to work successfully. It was found that submit host / central manager did not have sufficient time to carry out the matchmaking and to dispatch jobs before the execute hosts went into the hibernating state. By changing the keyboard/console idle time in Condor to 10 minutes (rather than the default 15), recently woken up machines now went from the Owner to Unclaimed state early enough for jobs to be started on them before hibernation occurred. Note that since the idle timer updates only every 5 minutes by default, the Start expression should evaluate to True if at least 10 minutes idle have elapsed (i.e. a strict >= test is needed).

The automatic wakeup process can be seen in action in the Condor View screenshot shown in figure 1. Initially little machine activity is seen as there are no jobs in the queue (only those machines where users are logged in and hence in the Owner state are present) . When increasing numbers of jobs are submitted though, machines are woken up and the size of the pool changes dynamically to suit the demand. Usage statistics from the PowerMAN Management Reporting Platform are shown in figure 2. These cover a three month period over the summer vacation for one of the teaching centres with Condor installed. Since there was little use from ordinary users during this time, almost all of the activity is down to Condor. Apart from a few blips (possibly caused by the problems described below) the amount of wastage caused by running Condor jobs is extremely small with less than £100 of electricity wasted

[2] These are a UDP packets each containing 6 bytes of ones followed by 16 repetitions of the MAC address of the machine to be woken.

during the quarter.

The wake up scheme works well in practice but there are some potential pitfalls and drawbacks to it. To begin with, the scheme assumes that any Condor job can run on any machine in the pool so that it is not possible to have machines with say different amounts of memory suited to different jobs (this is the main reason why all machines in the pool have essentially the same specification). In addition, it is not possible for users to specify particular teaching centres in the job's `Requirements` so for example those centres with particular pre-installed application software are only chosen. By far the most serious problem of this type occurs where there is a mistake in the `Requirements` specification so that it matches none of the machines in the pool. In this case the entire pool may be repeatedly woken up only to go back into hibernation again. A safety check has since been included in cron script so that the wake up is turned off if more than 90% of the machines in the pool remain in the Unclaimed state for an hour. This may still not be foolproof and the status of the pool does need to be monitored fairly regularly.

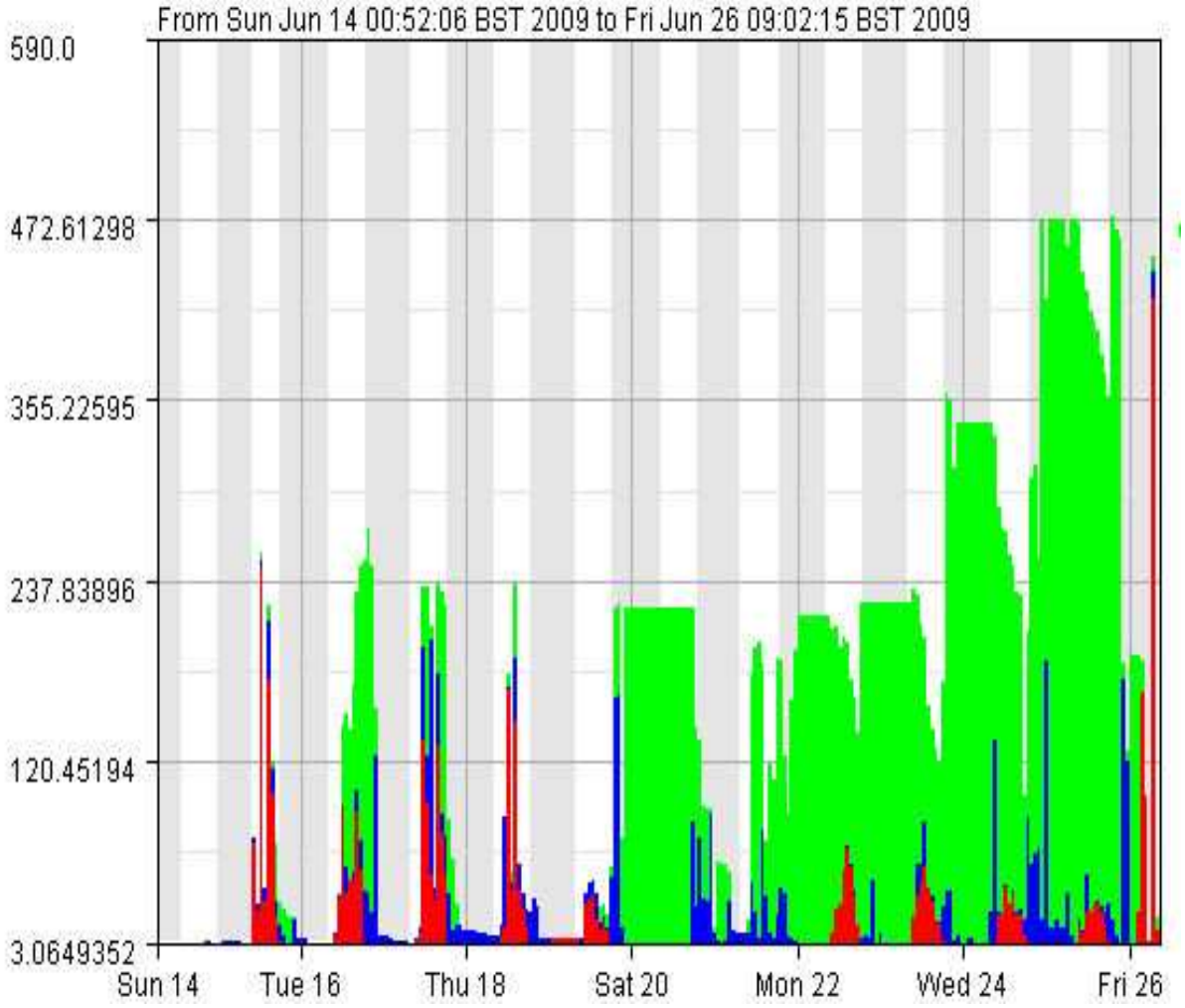

Figure 1: Condor pool machine usage statistics over a period of 12 days. Red indicates machines in the Owner state, blue indicates Unclaimed machines and green, machines in the Claimed state running Condor jobs.

Another problem can occur if users submit large numbers of jobs ($>$ 10 000 approximately) in a single cluster. This can cause the condor scheduler to become so heavily loaded that the queue status is no longer retuned on issuing a *condor_q* command. In this case no machines are woken up by the cron script as current demand cannot be ascertained. The problem should be transitory as the scheduler load will eventually reduce allowing *condor_q* to function properly again but this does lead to an unnecessary delay in getting jobs running. We have tried to educate users not to submit such large clusters but for those who genuinely need to do this a *gentle_submit* script has been created so that jobs are "drip-fed"

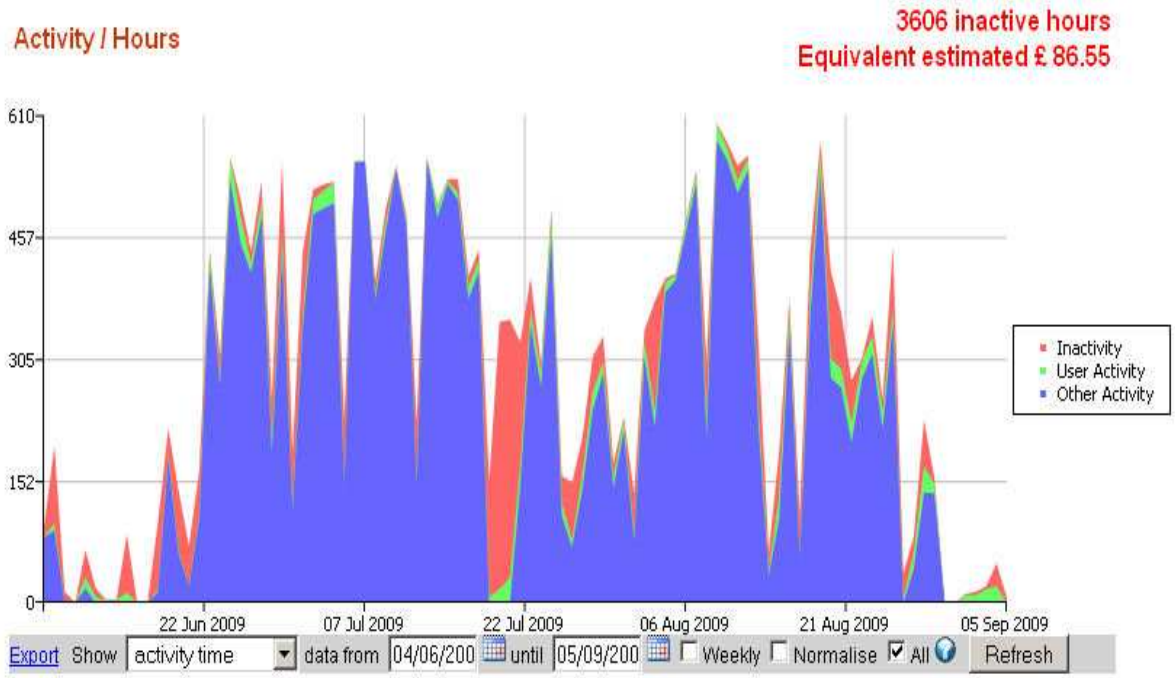

Figure 2: Usage statistics from the PowerMAN Server over a three month period for a teaching centre containing 28 machines. Blue indicates machines running Condor jobs, green, machines where users are logged in and red, machines which are inactive. Vertical scale shows daily activity in hours.

to the Condor scheduler and the number in the queue at any one time is kept within sensible limits.

One final drawback to the scheme is that the cron script can only ever keep an estimate of the number of available machines in each teaching centre since some may be permanently powered off, broken or in a state where they cannot be woken to run Condor jobs. In practice the number of these machines is generally small in comparison to the overall pool size and this does not significantly affect throughput. On occasion though, centres are taken out of service for upgrades etc and it is necessary to reflect this by altering the cron script (this contains a list of centres "in service" and available for wake up).

4 Recent and Future Developments

MATLAB seems to be a very useful way of helping users run jobs on our Condor pool particularly those with little interest in writing their own code in languages such as C or FORTRAN. The fact that many users work in a Windows environment and jobs need to be submitted from a central UNIX-based server does present a significant hurdle to many users though. This is exacerbated by the need to package the standalone applications and a create batch file wrapper to run them although the R2009 MATLAB release does seem to simplify things.

We could of course allow Condor jobs to be submitted from users own desktop machines but there are clear security risks with this approach and with users tending to submit large numbers of jobs to the pool there may be an unacceptable load placed on their local hard disk and network connection (to the extent where the machine becomes too overloaded to perform other office-type tasks). As a halfway house it may be useful to create a web-based interface for running demonstration MATLAB jobs and allowing the submission of small numbers of test jobs. This would give the user some confidence that their application will run under Condor successfully. The server-side script could also perform some sanity checks to ensure the correct packaging of applications as well as automatically create job submission description files and batch file wrappers. These files could then be used as templates when users come to submit large numbers of jobs from the command line.

The run-time MATLAB problems described earlier are relatively minor irritations but it would be useful to try and diagnose them. In the past this has proved difficulty since many of the applications run under Condor are based on stochastic methods and the problems are therefore difficult to reproduce

under the same conditions. We have recently seen the same problems with deterministic codes and these may provide a better avenue to explore. Running the application on a local machine with Condor access to the desktop should also provide some clues.

In the past six months, we have looked at introducing more drastic power-saving measures by hibernating machines after just 10 minutes of idle time. This has made life somewhat harder for the Condor submit host / central manager in getting significant jobs to run on recently woken up machines before they return to hibernation. A number of configuration changes have been made to get jobs running quickly. On the execute hosts, the transition from Owner to Unclaimed state now occurs after just 5 minutes of idle time with the timer now set to update every minute. On the submit host we have also increased the frequency of scheduler and negotiator cycles by reducing `SCHEDD_INTERVAL` and `NEGOTIATOR_INTERVAL` to 60 seconds. The `NEGOTIATOR_TIMEOUT` has also been increased to 10 minutes because of problems with socket timeouts.

Despite all of these configuration changes, around a quarter of machines in the pool go back to hibernation before they can run jobs when the entire pool is first woken up. The result is that the pool only becomes saturated with jobs after several wake up cycles (a period of a couple of hours). For most users this does not cause any real inconvenience. Clearly where machines are woken up only to hibernate again there is energy wastage but we believe this is offset by savings in hibernating machines sooner when users log off. During term time there were will be many thousands of log off events each day during office hours and in many cases the machine will not be occupied by another user within say 30 minutes. This inactive period can contribute to significant wastage. In contrast Condor usage tends to be sporadic and there are unlikely to be more than a few wake up cycles each day (or even each week sometimes). By examining the PowerMAN statistics it will be possible to judge just how effective this new strategy is in reducing energy consumption still further.

We will soon be moving to Condor version 7.2 on both the execute hosts and central condor server. This contains some interesting power management features including support for placing execute hosts in one of several low-power states, a command for waking up powered-down machines (*condor_wakeup*), and a way of keeping a record of powered-down machines through "persistent" machine ClassAds. The built in power-down capability could be useful in catching any machines which have have not hibernated after Condor jobs have finished running on them. However, since only a fraction of the teaching centre machines on campus run Condor, the PowerMAN software is likely to remain our primary power management tool on the PCs.

The information contained in the "persistent" ClassAds could be very useful in allowing more flexible job requirements to be specified and enabling a larger more heterogeneous mix of machines to be used in the pool. Rather than reinvent the wheel and attempt to ape the matchmaking logic already present in Condor in a script, it would be extremely useful if Condor could determine which machines need to be woken up to satisfy the requirements of idle jobs. Also a handy (though possibly difficult to implement) feature would be if powered-down machines could be represented through an additional state seen using *condor_status*.

As mentioned earlier, it is impossible to accurately determine which machines have disappeared from the pool through power-saving and which are permanently powered off, broken or otherwise not responding. By restricting the lifetime of these "persistent" ClassAds, and waking up all of the machines in the pool on a regular basis it should though be possible to get a reasonably accurate picture of the current state of affairs.

Whilst on the subject of a "wish list" for new Condor features it may be worth mentioning a couple of other requests. Our ideal policy for Condor jobs would be to run them only when a user is not logged on and never when a user is. The use of load averages and keyboard/console idle times is in some ways an obfuscation and all that we really require is a simple Boolean-valued attribute in the ClassAds which indicates whether a user is logged in or not. The PowerMAN website contains details of non-user generated loads which may be caused by screen savers for example and phantom mouse events triggered by fluorescent lighting interfering with optical mice. Both of these are likely to inhibit the running of Condor jobs and could be solved by a simple "logged in user" ClassAd.

Finally a way of allowing jobs to claim all of the slots on an execute host would allow jobs with large memory requirements to be accommodated as well as leaving in place the more energy efficient "one-job-per-core" policy. This approach is taken on our high performance cluster machines under Sun Grid Engine where users can choose any number of MPI processes to run on each node with one per core being the default.

5 References

1. The MathWorks website is at: http://www.mathworks.co.uk

2. Epidemiological consequences of an incursion of highly pathogenic H5N1 avian influenza into the British poultry flock, *Proc. R. Soc. B*, 2008 Jan 7;275(1630):19-28.

3. See online under: http://beryl.cs.cf.ac.uk/Web

4. Details on the Data Synergy website at: http://www.datasynergy.co.uk/