Towards a greener Condor pool: adapting Condor for use with energy-efficient PCs

Ian C. Smith

University of Liverpool
Advanced Research Computing
i.c.smith@liverpool.ac.uk

Abstract

Condor provides an extremely efficient way of harvesting unused processor cycles from resources such as desktop PCs. Although these resources may only intermittently be available, there is a tacit assumption that the majority of execute hosts in a given Condor pool will remain powered-up most of the time and capable of running Condor jobs at times when they would otherwise be idle. The introduction of automated power saving on Condor execute hosts undermines this assumption since machines will generally be powered-up only when users are logged into them and hence when they are generally unavailable to run Condor jobs.

In this article, I describe my experiences in providing a Condor service based entirely on a pool of power-saving PCs running the Windows operating system. The intention here is to give insight into how some of the problems were tackled and to describe those difficulties which remain rather than providing an all-round solution. As such, I hope it will be useful to others and I welcome any feedback.

1 Introduction

Given the current pressures on IT departments to reduce costs, there is significant interest in improving the energy efficiency of computing resources like data centres/server rooms through the use of technologies such as multi-core nodes. The recent fashion for all things "green" and "sustainable" also means that extra environmental kudos can be gained by institutions and businesses adopting a more energy-efficient approach to IT provision.

In environments such as universities where large, centrally managed, PC estates abound, IT energy costs are likely to be dominated by the aggregate consumption of desktop machines. Many of these machines will be used for only a fraction of the time during which they are powered-on and overall energy wastage is exacerbated by long periods of inactivity e.g. during vacations (as well as an inherent degree of over-provisioning to cope with peaks in demand at particular times during the academic year). At the University of Liverpool, calculations have shown that our classroom PCs are only in use for around 6 % of the total time in one year and the figure for staff machines is only slightly higher at 8%.

In the past three years, the Computing Services Department (CSD) at Liverpool, has adopted a proactive strategy of reducing the overall energy consumption of the several thousand PCs located across campus [1]. Initially, a policy of automatically powering-off machines after 30 minutes of inactivity (provided no user is logged in) was adopted. Currently, machines are forced into hibernation if there has been no activity for at least 15 minutes.

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.

A handful of UK universities experience significant and steady demand for machines by Condor users most of the time and administrators at some of these institutions have successfully argued against implementing power-saving as the additional cost of running Condor jobs is small and can be justified on return-on-investment grounds. However in our experience, Condor use tends to be bursty i.e. heavy for short periods with almost no usage for relatively long interim periods (this may of course change as we encourage more users to adopt Condor). A typical usage pattern is shown in figure 1. It is therefore difficult to justify the avoidance of power management on economic grounds here.

When the power-saving regime was first introduced, we simply opted-out a number of classrooms containing PCs (referred to locally as teaching centres) so that they could run Condor jobs at any time. Clearly this is not a scalable solution and in order to 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.

In the absence of anything to build on, a home-grown solution was adopted and used up to a few months ago. The approach worked reasonably well but had some fairly significant drawbacks. More recently, use has been made of the built-in power management features provided by Condor version 7.4.x which has allowed much greater flexibility. Both the home-grown approach and the Condor approach are described in detail later.

2 The University of Liverpool Condor Pool

The University of Liverpool Computing Services Department (CSD) Condor Pool was first established as an experimental service around five years ago and has been expanded steadily to a point now where there are up to around 600 job slots available 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 2 000 PCs available in total across the University, we have deliberately chosen only those with the highest specification 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 currently based on Windows XP Service Pack 3 but which will soon move to Windows 7. Application changes and patches are generally applied via weekly re-imaging although there is scope for implementing small 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 5 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 log in 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 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 thus making it easy to identify machines belonging to a particular teaching centre. The teaching centre name is also included in a bespoke machine ClassAd attribute. This configuration is useful in identifying which hibernating PCs are to be woken up and is discussed in the section on power management later. The number of PCs in each teaching centre varies from around twenty to sixty.

All jobs are submitted to the Condor pool via a single combined central manager and submit host. Although there are known scaling problems with

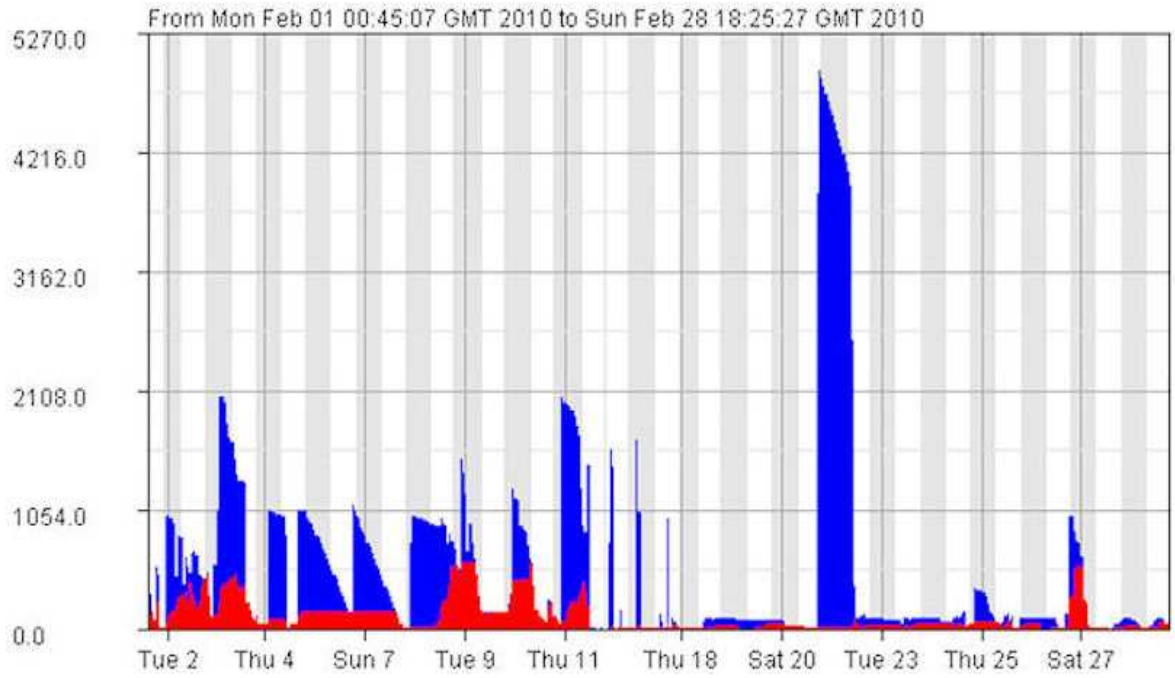

Figure 1: Usage statistics from Condor View for a period of one month prior to Condor power management being used. Idle jobs are shown in blue and running jobs in red. There are large peaks in demand separated by periods of almost no activity.

this, the extra security afforded by a single access point was an overriding consideration. The central manager runs on a Sun Fire V445 server with four cores, 16 GB RAM and a 1.2 TB RAID filestore used exclusively by Condor. The operating system is Solaris 10. Condor users log in 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).

At the time of writing, Condor version 7.4.3 (prerelease) is currently used on the central manager and 7.0.2 on the execute hosts although we aim to move to 7.4.2 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 central job submission point to our campus grid (UL-GRID) which uses Condor-G.

3 A Home-Grown approach to Power Management

As outlined earlier, there are two main difficulties in adapting Condor for use with power-saving machines, namely: how to ensure that machines do not go into hibernation when running Condor jobs and how to wake up hibernating PCs so that they can run Condor jobs.

Initially, to address the first problem, a system process ran a DOS `.bat` program after 30 minutes of inactivity was detected. This 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 and

are only removed later when the Condor garbage collector (*condor_preen*) runs again.

To gauge the effectiveness of the policy, we have made use of the PowerMAN power monitoring system from Data Synergy [2]. This comprises two main components, namely: 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[1]). 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 or individual centres can be summarised (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. There are also freely-available alternatives to PowerMAN.

The PowerMAN system also provided a more reliable method of preventing machines running Condor jobs from being forced into a low-power state. A list of "protected programs" can incorporated into the PowerMAN configuration so that, when any of them is running, the PC remains active. By making one of these programs the *condor_starter* process (which is only present when Condor jobs are running), it was possible to prevent hibernation whilst Condor jobs are running.

When the machines go into hibernation, almost all of their components are powered-down but the Network Interface Card (NIC) remains active (this is also true of other low-power states). The NICs on all PCs in the pool have a "wake-on-LAN" (WoL) capability which allows them to bring hibernating machines back to full operating mode on receipt of so-called "magic packets"[2]. It is this functionality which is key to waking up machines in the pool according to the current demand.

It is worth pointing out that many network configurations do not provide for routing of these WoL packets which are transported using the Internet Control Message Protocol (ICMP). It may be necessary therefore to put in place a number of ICMP gateways giving access to different subnets. Fortunately, the topology and configuration of our network allows WoL packets to be routed using limited IP broadcasts (e.g. using IP addresses of the form 138.253.nnn.255 where nnn is the subnet number).

A *cron* job runs on the submit host / central manager every 15 minutes which checks the state of the Condor queue against that of 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 Perl script which sends the required WoL packets to them. For this to work, 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 was 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 a sufficient number 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.

Usage statistics from the PowerMAN Management Reporting Platform are shown in figure 2. These cover a three month period over the summer

[1] 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 faster but cuts consumption only by about a half.

[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.

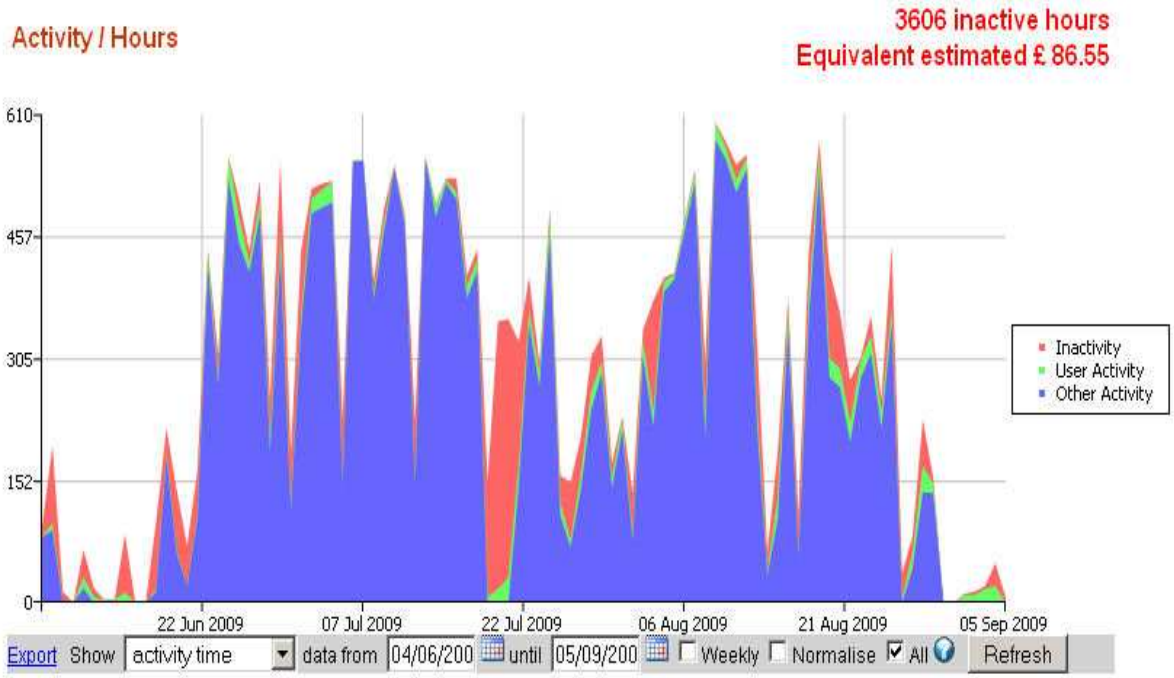

Figure 2: Usage statistics from the PowerMAN Server over a three month period for a teaching centre containing 28 machines. Blue indicates machines running Condor jobs, green, machines where users are logged in and red, machines which are inactive. Vertical scale shows daily activity in hours.

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 attributable 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 worth of extra electricity wasted during the entire quarter for one centre.

All of this presupposes that we know which machines are hibernating as opposed to those that might be permanently powered-off or otherwise out-of-service[3]. This turns out to be a very difficult (and possibly intractable) problem to solve. In the original setup, it was simply assumed that by consulting our "database" of teaching centre machines (actually stored as a number of UNIX text files) we can work out how many machines there *ought* to be available in each centre. Then by subtracting the number of machines which *appear* to be powered-up (derived from *condor_status*) from the total number, an estimate of the number of hibernating machines could be made. There are fairly obvious pitfalls with this approach which are now described in more detail.

4 Drawbacks and Limitations with the Home-Grown Approach

The original automatic wakeup scheme, although fairly crude, seemed to work quite reliably when a period of 30 minutes of inactivity was allowed before hibernation. When this was reduced to 15 minutes, to provide greater energy savings, problems began to appear. It was found that when a large number of machines were woken to satisfy a sudden surge in Condor jobs, many of the machines went back into hibernation before starting to run jobs. This is illustrated in figure 3.

The situation was improved slightly by reducing the keyboard/console idle time limit in Condor to 5 minutes (rather than the default 15) to buy extra time. This meant that recently woken up machines

[3]The variety of situations in which PCs become unavailable to Condor is actually quite surprising and only became apparent through visiting the teaching centres. In some cases it was found that the weekly reimaging process had failed before completion leaving the PC stuck in a limbo state where a manual reboot was needed for it to operate properly again.

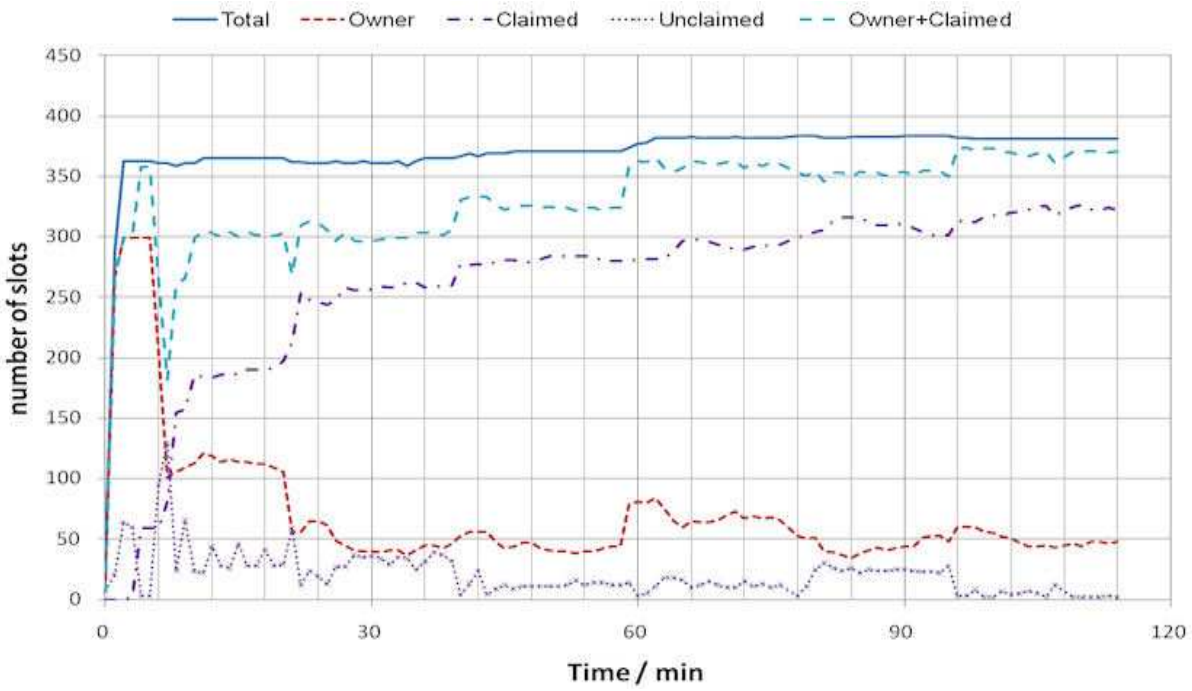

Figure 3: Machine state statistics recorded at 1 minute intervals. All of the machines in the pool were forcibly woken at approximately 30 minute intervals. Only after a few wakeups do all of the available machines start to run jobs.

now went from the Owner to the Unclaimed state[4] after 5 minutes so that the submit host now had 10 minutes to get jobs running before hibernation set in. This means that overall throughput is reduced and it is possible that the same machines are repeatedly woken up only to go back into hibernation thus leading to unnecessary energy wastage. As of writing, this is still under investigation and the reasons for this phenomenon are still unclear.[5].

An important limitation with the Home-Grown scheme is that it assumes that any Condor job can run on any machine in the pool (and at any time) so that it is not possible to employ machines with, e.g. 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 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 To address this, a safety check was included in the *cron* script so that the wakeups are turned off if more than 90% of the machines in the pool remain in the Unclaimed state for an hour.

There are also a few other drawbacks to the scheme which may not be immediately obvious. Firstly, by waking machines one centre at a time to satisfy the demand, those machines which run jobs can become be concentrated in just a few areas. If jobs run for long periods, then heavy overnight use of machines (and most Condor jobs tend by their very nature tend to be compute-intensive) in a par-

[4]Machines in the Owner state are occupied by logged-in desktop users and are unavailable to Condor. Claimed machines are those that Condor is making use of and machines in the Unclaimed state are those which are available to Condor (note that in general this can include offline machines).

[5]There is a school of thought which believes that this creates additional wear and tear on machines, reducing their reliability and possibly lifetime. Steering clear of a potential "religious war", I'll just state, for the record, that since our PC Systems team do not regard this as a problem, I am untroubled by it. As they used to say on USENET though – YMMV.

ticular classroom can lead to the room becoming uncomfortably hot first thing next morning. This is especially true during the summer for centres without air conditioning (even here in the Britain !).

Secondly, some of our classrooms contain over 160 machines which, if woken simultaneously, can create enough of a distraction to disturb, if not annoy, students using the centre (especially if on-line exams are taking place). Indeed there is anecdotal evidence that some users are powering off PCs to prevent this happening (clearly an example of the Law of Unforeseen Consequences !). Both of these problems could be addressed by waking machines up individually and in random order.

5 Power Management using Condor

As of Condor version 7.4.0, a number of features have been introduced to aid in the power management of execute hosts [3]. Condor can now place an execute host in one of several low-power states conditional on how long the host has been inactive for. Before entering the low-power state, the execute host informs the central manager of its intentions and the pool's *condor_collector* notes that the host has gone offline by recording a special persistent ClassAd in a log file (defined by `OFFLINE_LOG`). An optional expiration time for each ClassAd can be specified with `OFFLINE_EXPIRE_ADS_AFTER` (the default is the length of the UNIX epoch). The *condor_negotiator* can perform matchmaking between idle jobs and persistent offline ClassAds and then signal that a match has been made to a new Condor daemon called *condor_rooster*.

The appropriately named *condor_rooster* attempts to wake up machines by running a program called *condor_power* which effectively implements the WoL functionality mentioned earlier. Wakeup is conditional on an expression defined in `ROOSTER_UNHIBERNATE` which defaults to `Offline && Unhibernate`.

The wakeup process does not operate continuously but in cycles, the period of which is defined by `CONDOR_ROOSTER_INTERVAL`. It is possible to substitute *condor_power* for developers' own version using `ROOSTER_WAKEUP_CMD` to specify the developer version. Developers' "roll-your-own" code will generally need to parse the offline ClassAds piped to it by *condor_rooster* in order to extract the broadcast and MAC addresses of machines to be woken via WoL.

6 Implementing Power Management using Condor

At Liverpool, we already have a third-party power-saving scheme in place and so have decided to keep this rather than adopt the Condor implementation. This of course does raise the problem of how to generate ClassAds for offline machines. The approach taken here is fairly straightforward. Given that we know which machines make up our pool in total (a rather big assumption as will be seen shortly), and the number of machines currently active (from *condor_status*), then the set of offline machines is the subset $O = P - A$ where P is the set of all pool machines and A the set of active machines. These then are the machines which we need to publicise via *condor_advertise*.

There are two caveats to this; one small and one large. Firstly, *condor_status* does not provide completely timely information about the pool state since ClassAds are only refreshed periodically (by default every 15 minutes). Some machines listed by *condor_status* may therefore be artefacts of hosts which have since gone into hibernation. In practise this does not cause significant problems.

The second caveat is much more important and concerns the accurate determination of which machines in the pool are hibernating (rather than powered-off or otherwise out-of-service) A *cron* job running on the central manager now wakes up all of the machines in each teaching once a week on different days. Following the wakeup call and after a further delay of 5 minutes, an attempt is made to contact the *condor_startd* on each machine by using:[6]

```
condor_status -l -direct <hostname>
```

If a *condor_startd* responds within 5 seconds, then it is assumed that the machine is available for use by Condor and a record of (some of) its ClassAd information is made for use as a persistent offline ClassAd. Note that this does not guarantee that the host

[6] This is similar to the UNIX *ping* command which provides a sanity check that hosts are online but does not guarantee that any services will be available from them.

will run a Condor job once woken up (and before going into hibernation again) but it does provide an extra degree of confidence.

Of course it may be the case that a machine has gone out of service since the last time it was tested in this way, in which case the ClassAd will be stale and invalid. Testing machines more frequently would help reduce this possibility but at the expense of additional wasted energy spent on wakeups. The problem may also occur with Condor's own power management features if machines are not used for long periods of time. As mentioned earlier, this is a seemingly intractable problem analogous to the Schrödinger's cat thought experiment[7]. Only by forcing machines into a (possibly different) known state can we ascertain what their actual state was.

Only a subset of the machine information is recorded and published as ClassAds, namely these attributes:

```
Name
Machine
Disk
Memory
Cpus
TotalCpus
TotalMemory
KFlops
Mips
HardwareAddress
Start
Subnet
```

A bespoke ClassAd is used to indicate in which teaching centre a PC resides. Two other ClassAds are also used as a time stamp: `ClockMin` and `ClockDay`. The values of these attributes are updated by a *cron* job which runs every 15 minutes and publishes the relevant ClassAds. It first invalidates all of the existing offline ClassAds, then advertises all of the machines which are thought to be offline (by consideration of the machines which are currently active) and finally updates the `ClockMin` and `ClockDay` timestamps. These two attributes can be used in jobs' `Requirements` specifications so that jobs will only run at certain times (e.g. overnight or at weekends for long running jobs).

The *condor_rooster* daemon is configured to run every 10 minutes since this fits in well with the 15 minute inactivity limit used on the execute hosts. Condor's own *condor_power* executable has been replaced by our own Perl script. This limits the number of machines woken up on each cycle to 25 (i.e. a possible 50 job slots) so that the central manager does not get "swamped" as was the case with the Home-Grown approach.

When Condor power management was first tried it was found that *condor_rooster* would attempt to wake up all of the machines which matched a particular job's requirements regardless of the number of idle jobs which the *condor_negotiator* successfully matched. This meant that, in theory, the whole pool could be woken to run a single job. To get around this, a check is made on the number of idle jobs in the queue and this value is used as an additional limit on the number of machines to wake up on each cycle. The matchmaking bug has been addressed in Condor version 7.4.3 and version 7.5.3 adds an extra configuration option (`ROOSTER_MAX_UNHIBERNATE`) to limit the number of machines woken up on each cycle.

Another problem was found when assigning a random `Rank` value to each machine so that machines are woken in random order (to prevent the same machines being woken repeatedly). It was found that this had no effect and to achieve the desired results, all of the machine names passed to *condor_power* by *condor_rooster* were sorted randomly before waking a limited number of them. This is analogous to shuffling a deck of cards before each deal however here the number of cards (i.e. machines) dealt each time may vary. This bug has been addressed in Condor version 7.5.3 through the configuration option `ROOSTER_UNHIBERNATE_RANK`.

One final snag was found when a large number of jobs were suddenly removed from the queue or when the queue eventually drained of jobs as they completed. Here the negotiator would continue to match these now non-existent jobs with offline ClassAds, resulting in machines being woken unneces-

[7] A less scientific animal analogy might be that of Monty Python's famous parrot, whose existential state, you may recall, was a matter of some debate. Like the "Norwegian Blue", the state of an offline Condor PC can only be truly determined by attempting to wake it up. Only then is it clear if we are dealing with an ex-PC which has shuffled of its network or whether it was in fact just sleeping.

sarily. To solve this, a 5 minute cut-off was placed in the last match time (as advertised through the offline ClassAds) using the expression:

```
Unhibernate = CurrentTime - \
 MachineLastMatchTime < 300
```

rather than the recommended:

```
Unhibernate =!= Undefined
```

7 Future Directions

The success of Condor's power management solution will allow our Condor pool to be extended and the intention is to include all of the available teaching centre machines as execute hosts eventually. Some of these machines may only have fairly low specifications but if they are unsuited to certain jobs, then requirements specifications can ensure that they are not used by them (and importantly are not woken up to run them). It may turn out that the most inferior PCs are rarely, if ever, used by Condor however, by including them in the pool, no additional electricity costs are incurred.

In some applications, jobs may complete in a short space of time (say twenty minutes or so) and have only modest memory requirements. Here the low specification machines can be put to good use since users will generally not be too concerned whether a large batch of their Condor jobs takes say two or three days to complete (on mostly slower machines) instead of one (on the faster PCs). In fact, if the overall queue size is large, it may make more sense to run these jobs on slower machines rather than wait for faster ones (running more demanding jobs) to become available.

Machines in the pool can also be ranked using offline ClassAds so that the newer, more energy-efficient, machines are woken up in preference to older hardware. This will ensure that overall energy efficiency is maximised.

The current method of waking up offline machines periodically to determine whether they are available to run Condor jobs works reasonably well but there is still scope for significant improvement. It has become apparent that this method tends to under-estimate the number of machines available. This is evident from the Condor View statistics where the overall pool size increases as the number of ordinary logged-in users tends to peak daily around lunch time. If the number of offline machines was known accurately, then we would expect the overall size to be constant as machines in the Owner state simply replace those offline ones in the Unclaimed state.

One way of improving the accuracy may to use a scoring technique. A record could be kept of the last time (or past few times) that each machine appeared to be powered-up according to *condor_status* and possibly when it last ran a Condor job. The degree of confidence that a particular machine will run a job after wakeup could then be described by using a monotonically decreasing function of the time since it last appeared (e.g. a decaying exponential function). Machines could then be ranked by combining these confidence values with the random rankings so that the machines most likely to run jobs are woken first. Clearly it would still be necessary to wake machines at the lower end of the confidence range periodically or they may (in a kind of self-fulfilling prophecy) disappear from the pool permanently.

At present, it can be difficult to distinguish between online and offline machines from a casual look at the *condor_status* output since both online and offline machines may appear as Unclaimed/Idle. In fact a constraint needs to be added in order to separate them i.e.:

```
$ condor_status -constraint \
            Offline==True
```

for offline machines and for online machines

```
$ condor_status -constraint \
            Offline=!=True
```

(`Offline` is only defined for offline machines). To help clarify this, it would be very useful if an additional machine state could be used to represent offline machines although this would obviously require significant code development by the Condor team.

An additional machine state would also make the Condor View statistics much easier to interpret. Prior to introducing Condor power management, it was immediately obvious from the Condor View statistics where machines were powered-up but not running jobs (thus wasting energy) since these were the ones in the Unclaimed state. Now machines marked as Unclaimed can be offline or online and

it not clear which machines, if any, are powered-up but inactive.

The necessity of ramping up the number of woken up PCs remains an irritation and it would be useful to be able to wake up the pool as quickly as possible so that throughput is maximised. Empirically it has been found that the number of PCs which start to run jobs after a "global" wakeup seems to be linked to the state of the Condor collector and (possibly) scheduler. After restarting these daemons, on the order of two hundred slots begin to run jobs before hibernation sets in however on other occasions this may be reduced to around fifty.

There is one final point concerning the energy-efficient use of Condor on a Windows-based pool which is worth making in closing. Such a deployment restricts Condor jobs to the vanilla universe where built-in checkpointing (as implemented by linking against the Condor checkpointing library) is unavailable. Here job evictions can cause useful work to be lost leading to "badput" rather than throughput and consequent wastage of electricity. By encouraging users to incorporate explicit checkpointing in their own codes though this loss can be minimised. One approach to this for MATLAB applications is described in [4].

8 References

1. For details of PowerDown see online at: http://www.liv.ac.uk/csd/greenit/powerdown/

2. The Data Synergy website is at: http://www.datasynergy.co.uk/

3. See the section on Power Management in the Condor Manual available on the Condor website: http://www.cs.wisc.edu/condor/

4. See online at the Liverpool Condor site: http://www.liv.ac.uk/csd/escience/condor/checkpoint.htm

9 Acknowledgement

Sincere thanks are due to Dan Bradley of the University of Wisconsin Condor Team for his help in the successful adaptation of our Condor pool to Condor power management.