

Analysis of Power-Saving Techniques over a large multi-use Cluster with variable workload

A. Stephen McGough^{1*}, Matthew Forshaw¹, Clive Gerrard^{2‡}, Paul Robinson³ and Stuart Wheater⁴

¹*School of Computing Science, Newcastle University, Newcastle upon Tyne, UK*

²*Information Systems and Services, Newcastle University, Newcastle upon Tyne, UK*

³*Red Hat Inc., Newcastle upon Tyne, UK*

⁴*Arjuna Technologies Limited, Newcastle upon Tyne, UK*

SUMMARY

Reduction of power consumption for any computer system is now an important issue, although this should be done in a manner that is not detrimental to the users of that computer system. We present a number of policies that can be applied to multi-use clusters where computers are shared between interactive users and high-throughput computing. We evaluate policies by trace-driven simulations in order to determine the effects on power consumed by the high-throughput workload and impact on high-throughput users. We further evaluate these policies for higher workloads by synthetically generating workloads based around the profiled workload observed through our system. We demonstrate that these policies could save ~45% of the currently used energy for our high-throughput jobs over our current cluster policies without affecting the high-throughput users experience. Copyright © 2010 John Wiley & Sons, Ltd.

Received . . .

KEY WORDS: Cluster; Energy; Policy; Simulation

1. INTRODUCTION

In recent years the requirement for computing service providers to reduce the energy consumption of their resources has grown. Initial steps to achieve this have included turning computers off when not required and a change to purchasing policies in order to favour more energy efficient offerings. To minimise expenditure it is also desirable to use computing resources for multiple purposes, however this introduces contention between the different user bases and can lead to increased energy consumption through repeating work when one of the parties has to back-off.

[‡]Work carried out whilst working for ISS.

*Correspondence to: School of Computing Science, Newcastle University, Newcastle upon Tyne, UK. E-mail: stephen.mcgough@ncl.ac.uk

†Please ensure that you use the most up to date class file, available from the CPE Home Page at <http://www3.interscience.wiley.com/journal/117946197/grouphome/home.html>

One common approach to multi-use of computers is allowing them to be used for both interactive users and as nodes in a high-throughput computing resource, allowing large numbers of computational jobs to be run. In such scenarios it is common to favour interactive users over the high-throughput users.

Policies can be defined to optimise interactive user experience, high-throughput user experience (turnaround and capacity) and energy consumption. One option is to power down resources when not in use by interactive users. However, this can lead to resource starvation for high-throughput users who previously used these computers when idle, so policy models should allow computers to be woken up for high-throughput computing.

In previous work [1] we proposed an architecture capable of managing policies for power management of computers within an institution. However, the determination of an optimal set of policies is a complex process as the behaviour can often be difficult to determine a-priori. As subtle policy changes can have significant impact on one or more of the key areas.

One solution to determining an optimal policy set is to test policy changes on the live system. This has three significant drawbacks: i) running the system under the new policy for a significant amount of time to ensure statistical relevance – to overcome seasonal load variation; ii) detailed logging, of both interactive users and high-throughput jobs, along with energy monitoring is required; and iii) a danger that changes to the production system could have unpredicted (negative) consequences for the current users. This leads to making minor modifications to the policy set where we are pretty sure that the impact on users will be low; other more significant changes being considered to be too dangerous.

Two alternatives exist: i) a test environment or ii) a simulation of the whole environment. A test environment removes the need for site-wide monitoring and does not affect the production user base, however time is still required to evaluate changes and we need to justify how results from the test environment would map to the whole environment. We have instead developed a simulation of the cluster environment to evaluate different policies. As the traffic running through the environment is highly seasonal we use a trace-based simulation. This allows us to quantifiably, and quickly, compare different policies against the same workload and interactive user requirements.

We further extend our simulation by generating synthetic workloads for high-throughput jobs by analysis of the trace logs. This allows us to generate synthetic high-performance users which exhibit the same characteristics as our real users. We can then arbitrarily scale up the number of high-throughput users allowing us to evaluate the performance of our policies for higher workloads.

Figure 1 illustrates the overall architecture of the computer cluster we are modelling. Computers, distributed through the institution, are assumed to be under their own power management. Although in this work we do show how changes to the policy of these computers can help improve the energy efficiency of the whole system we assume in general that we do not control the policy of the computers. We instead control the policy over how high-throughput jobs are handled.

Interactive users, who are assumed to be able to log into any computer within the cluster, can arrive at any time that a particular room is open and log into the computer of their choice. By contrast, high-throughput jobs are submitted through one centrally controlled access point with the system itself determining the computers that will be used. We possess trace data for both user types. We assume computers may go to sleep based on a pre-defined policy and that interactive users can

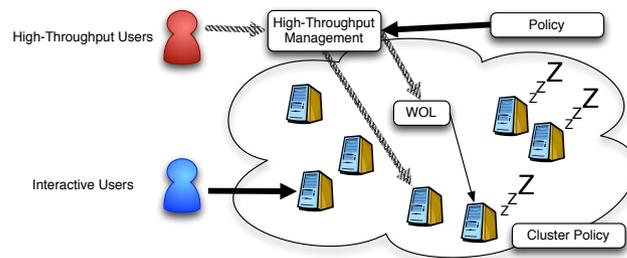


Figure 1. Overall architecture of multi-use Cluster

always wake up a computer. Different policies may allow the high-throughput jobs to wake up the computers.

The user trace data indicates login and logout time for the user, and the specific computer that the user occupied. In this paper we do not simulate alterations to this usage pattern. The high-throughput trace data, by contrast, contains only the time that the jobs were submitted and their duration as changes to the policy will change the computers used and the time at which the jobs complete. Some jobs failed to complete and were later terminated by the user, or in some cases the system administrator. In these cases the time that the job was terminated is used instead of a duration with all simulations terminating the job at the same time.

The aim of the simulation is to determine an optimal set of policies which best satisfy the following constraints:

- Quality of Service should be maintained for Interactive users despite power saving or high-throughput jobs.
- Power will be consumed by interactive users or a high-throughput job; this is acceptable. We seek to minimise power consumed when computers are not performing productive work.
- High-throughput users should have quick turnaround on their work: time between submission and job completion should be as close to the job duration time as possible.

To achieve the first constraint the high-throughput jobs should take lower precedence than the interactive users. This does not necessitate that high-throughput jobs cannot be run on the same computer as an interactive user, only that if the interactive user requires full use of the computer then the high-throughput job should be sacrificed. Jobs may also be evicted from a computer if the computer is rebooted. Slight delays while a computer resumes from sleep or a high-throughput job terminates are considered tolerable. We define the overhead of a job to be the difference between the true execution time of a job and the wall-clock time between job submission and job completion.

We assume that power consumed by the cluster through interactive users is a given. Reduction of this power is being achieved through replacement of older computers with more energy efficient computers; we focus instead on the time when users are not logged in and the computers can be used for high-throughput jobs.

In this paper we define the following three states for computers based on the Advanced Configuration and Power Interface (ACPI) specification [2]:

- **Active:** being used either by an interactive user or a high-throughput job. This equates to ACPI state S0.

- **Idle:** powered up but not actively processing work for interactive user or high-throughput job. Computer consuming less energy than in the active state. Equating to ACPI state S0.
- **Sleep:** computer state stored in RAM which remains powered. All other components are powered down. This allows for quick system resume without the need to restart the operating system. Equating to ACPI state S3.

We seek to target high-throughput jobs to more energy efficient computers, reduce the amount of time that is wasted before a job starts execution and reduce the chances of a job being evicted. This will reduce the time between job submission and completion with the further advantage of decreasing energy consumption. We also look at policy to reduce the amount of time a computer is idle awaiting either an interactive user or a high-throughput job.

In order to determine the effectiveness of our proposed policies we analyse our existing high-throughput logs to identify the characteristics of individual users. We are then able to generate synthetic workloads based around these user characteristics. Allowing us to increase the user workload, hence evaluating our policies.

Our previous work [1] was based around the Condor [3] high-throughput computing system based at Newcastle University. We continue this work by basing our simulations on the same system, exploiting the same data sets. However, the approach could easily be adapted to other high-throughput clustering systems.

The rest of the paper is structured as follows. In section 2 we discuss related work and existing examples of policy that are in use in various institutions. Section 3 discusses the policy architecture used to invoke the desired policy, while section 4 discusses a number of policies that have been identified, this includes ones that we have implemented and those we wish to evaluate before implementation. Section 5 discusses the simulation environment. In section 6 we describe the set of simulations we will perform along with results of the simulations before concluding in section 7.

2. RELATED WORK

The saving of energy whilst still being able to perform good computational work is an area of great interest. Minartz [4] proposed switching off nodes within a high-performance cluster to save energy. We go further in this work to show how different policies over how jobs are distributed around a high-throughput heterogeneous cluster can be more energy efficient. Minartz goes on to model the power consumption of individual components within a system based on the work being performed. This could be adapted to work with our system. The modelling of a Cloud data centre, including switches, routers and nodes is given by Kliazovich [5]. This is for a more homogeneous environment than ours and lacks the inclusion of interactive users.

Niemi [6] demonstrated that running multiple jobs on the same node within a high-performance cluster was more energy efficient. We expect such to be the same here for our work. Though at present we lack the knowledge about execution load for our workload to determine if this would work well.

A number of Grid and Cluster level simulators exist including SimGrid [7], GridSim [8], and OptorSim [9] though these focus more at the resource selection process both within clusters and

between clusters and lack the modelling of energy. SiCoGrid [10] allows for the modelling of computer power, though it models the entire cluster at a much lower level than we require here.

2.1. Existing Examples of Policy

Historically many institutions, like Newcastle University, had allowed their computer labs to remain powered up at all times. With the increasing requirements for institutions to save energy this waste of power was identified as an early target for savings. The initial policy was to place the computer into a shutdown (ACPI S5) state. This had the disadvantage that these computers were no longer available to Condor leading to high-throughput starvation.

The base Condor system allows for the definition of policy over how jobs will run and under what circumstances jobs will be evicted from computers. These policies can affect the amount of time a computer needs to remain idle before it can be used by Condor and which computers should be used, having a direct effect on the responsiveness of the Condor cluster and also on the power consumed by the cluster.

These policies have not been used in the past to provide energy efficiency, however, newer versions of Condor now take such things into account. Condor now includes the ability to send computers to sleep and then wake them up, via *Rooster*, as and when needed [11]. This policy system does require that Condor take control of the power-management of the cluster. If Condor has exclusive use of the computers this is not a problem, though it may cause contention if pre-existing power management tools are in use [1, 12].

Cardiff University has taken an approach in which computers will send themselves to sleep after a set time (normally cluster closure time) if there is no Condor job running on the resource. Powering themselves down as soon as there are no jobs left to service. Any Condor job arriving after this time will be unable to use the computer until it is re-started. This can lead to backlogs of Condor jobs if submitted after the computers have entered an offline state.

Liverpool University have implemented a more advanced policy. Computers still go to sleep when they have no work to service without informing Condor. However, a script run at regular intervals looks for computers which have ‘disappeared’, due to going to sleep (without informing Condor), and inserts a ‘fake’ resource description so that Condor thinks of it as being asleep allowing *Rooster* to wake it up as necessary [12]. This relies on having a pre-defined list of known computers and leads to intervals when Condor ‘loses’ computers, as it is unaware that a computer is sleeping but available for execution.

University College London uses a thin-client system for its open access computers though runs the client on a full-spec computer. This allows them to exploit the unused computing power of the computers as part of their high-throughput computing system. Though in itself this doesn’t reduce power consumption the ability to use computers for multiple purposes simultaneously helps them cut down on capital expense and maintenance.

In previous work we proposed a model in which computers send the ‘fake’ sleep notification themselves just before entering the sleep state, along with a script, run at regular intervals, to catch ‘lost’ computers (ones marked as being awake, but with no update for a pre-defined amount of time), which have failed to send the ‘fake’ sleep notification [1]. This has the advantage of not requiring a list of known computers and reduces the time that Condor is unaware that a computer is sleeping.

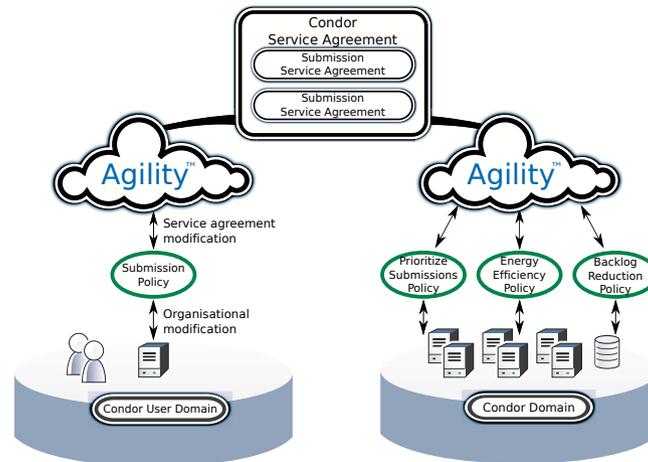


Figure 2. The Condor/Agility Architecture

3. POLICY ARCHITECTURE

In previous work [1] we have developed a policy architecture by layering Agility, the Arjuna Technology Federated Cloud Computing Platform [13] on-top of a standard Condor architecture [3]. Agility transforms traditional IT infrastructure into flexible, agile Private Clouds sharing services without sacrificing organisational independence.

Agility is installed as an overlay (Figure 2), capable of operating independently of existing IT infrastructure, monitoring and taking action when required. Policies can be written to automate regular tasks or to change the interactions of various parts of a system. Typically (intra- or extra-enterprise) interactions are agreed through an informal process, which is rarely recorded or monitored, making it difficult to understand the true costs incurred and benefits delivered across the organisations. Agility provides a means of capturing those relationships without impacting the existing operations. In its interaction with the underlying IT infrastructure Agility may be unobtrusive (simply recording the relationships), may monitor operations (reporting on conformance), or may actively drive resources (taking action to ensure conformance). We adopt the third approach here.

Our aim is not to re-develop services available through Condor, rather to augment them to provide intelligent management. This allows us to retain the advantages of many key features of Condor such as job and computer matchmaking and waking up of computers through Rooster.

4. POLICY

In this section we discuss a number of policies which can be applied to a multi-use cluster similar to the one at Newcastle University. This section is divided into policies we have already implemented, policies we have seen deployed in other clusters and proposed policies we would like to evaluate before attempting to implement them.

4.1. Previously Implemented Policy

We have previously developed a number of policies for managing the energy consumption of our cluster. These can be broadly categorised into cluster management, selection of computers to use and job management.

4.1.1. Cluster management Power management of computers covering when the computer can be awake (active or idle) and when the computer can sleep. The four power management policies are:

P1: Computers are permanently awake. This was the default policy used by most high-throughput computing installations before power saving. This policy can lead to large amounts of wasted energy when a computer is idle. Though as computers are always available it minimises overheads.

P2: Computers are on during cluster opening times or powered off otherwise with no ability to wake up. If the computer is servicing jobs at cluster close time it remains active until this and any subsequent jobs are completed.

P3(n): Computers sleep after n minutes of inactivity with no wakeup for high-throughput jobs. Initially we used a value of one hour as the resume time from shutdown was significant. However, the reliable sleep feature of Microsoft Windows 7 has made this process almost instantaneous. Though at present we still use the one hour time interval.

P4(n): Computers sleep after n minutes of inactivity with Condor being made aware of their availability. This is an extension of policy P3 allowing Rooster to wake up computers when needed.

4.1.2. Selecting computers to use These policies allow us to determine which computer to select for job execution.

S1: Condor default: note that this devolves into a random selection policy favouring powered up computers.

S2: Target the most energy efficient computers. Energy consumption for each computer is defined along with a Power Usage Effectiveness (PUE) the ratio of total amount of power used by a computer facility to the power delivered to computing equipment. This allows us to order the computers by $PUE \times \text{energy}$, targeting jobs as appropriate.

This process is an approximation to the efficiency of a job, as different computers will handle different computational tasks with different degrees of efficiency. One computer may be most efficient on memory intensive jobs whilst another may be more efficient on floating point dominant jobs. However, this policy aims only to steer jobs towards the more energy efficient computers based on our benchmarking.

4.1.3. Job management These policies allow us to alter the behaviour of Condor in terms of when to allow jobs to start running and when to cease attempting to process a job:

M1(n): A computer may not be used until it has been idle for n minutes. This Condor default is intended to prevent computers that are frequently used from being matched.

C1(n): Detection of ‘miscreant’ jobs. Condor attempts to run jobs to completion, this includes re-submitting evicted job due to computer crash, reboot or user precedence. If this happens n times we mark the job as ‘miscreant’. Selection of the value n needs to be made carefully: too small a value will create false positives whilst large values will waste time and energy.

Although a miscreant job may not be broken it may not complete, continuing to consume resources. We may then choose to terminate these jobs. Care needs to be taken as such jobs may be performing good computational work through some other out of bounds mechanism.

4.2. Alternative Proposed Policy

We include for comparison a policy proposed by Smith [12] for the power management of computers:

P5(m, n): Computers sleep after m minutes of inactivity with sleeping computers being advertised every n minutes. When a computer goes to sleep no information is sent to Condor. A service runs every n minutes checking for sleeping computers, posting a ‘fake’ advertisement for them.

4.3. New Proposed Policy

We wish to evaluate a number of proposed policies in terms of how much power they might save and the potential impact on the high-throughput users. Some of the policies may also have an impact on the interactive users of the cluster. It is not possible to determine those effects here, though by using the simulation we can evaluate the impact on high-throughput users and power consumption enabling us to evaluate whether such a policy would lead to a large enough advantage that it was worth considering the policy and potential impact on the interactive users. These can be grouped into cluster management and computer selection.

C2(m, n): Provision of dedicated computing resources. Extending the repeatedly evicted policy C1(n). Once a job has been evicted n times it is allowed to continue execution on a dedicated set of m computers. This would throttle the problem of long running jobs never completing due to repeated eviction though we would still need to monitor these jobs for non-completion.

C3(m, n, t): Timeout for dedicated computers. If there are more miscreant jobs than dedicated computers then policy C2(m, n) then all dedicated computers are blocked with jobs and the policy degrades to C1(n). However, if we select a time interval t over which we assume the job will not complete and can therefore be safely killed we regain the ability for the dedicated resources to allow long running ‘good’ tasks to complete.

M2: High-throughput jobs defer nightly reboots. Allow high-throughput jobs to run through the night and thus for longer than 24 hours. This policy addresses the same issue as M1.

M3: High-throughput jobs use computers at the same time as interactive users. Desktop computers are now more powerful. All computers at Newcastle University are at least dual core with many quad core. From observation the interactive load is often far less than the available computing power. Although some applications are capable of exploiting multi-core (e.g. CAD and MATLAB [14]) many are only capable of exploiting a single core leading to under-utilisation, which can be exploited through Condor.

In this case rather than jobs being evicted when an interactive user logs in an eviction can be triggered when the load placed on the computer exceeds the requirements of both the interactive user and the high-throughput job. Our trace of interactive uses of the computers does not include information on the load the user placed on the computer and we therefore assume for these

simulations that the load of both high-throughput jobs and the interactive user does not exceed the capability of the computer.

An evaluation of the potential energy savings and reduction in overhead time needs to be performed in order to determine if this policy could provide enough of an improvement to warrant live evaluation over the real cluster.

S3(i): Targeting less used computers. Computers in locations frequented by students tend to have short gaps between interactive users and many users each day in contrast with computers in less popular locations. This can also be affected by the ‘opening hours’. It would be beneficial to select less used computers reducing the chance of job eviction and hence less wasted power on incomplete execution.

It is not possible to know a-priori which computers will be unused in the near future, also this information would be seasonally affected. However, we can look for general trends in the usage patterns of computers from historical evidence and use this to help select the computers least likely to have a log in. We can favour computers based on their current state – an idle computer with greater chance of a login is used above a computer which is asleep but has a lower chance of login – or selecting the computer with the least chance of login irrespective of current state. We define the following 14 options for computer ordering:

- [1, 5, 8, 12] : Largest individual average interval: logout – login
- [2, 6, 9, 13] : Largest individual minimum interval: logout – login
- [2, 6, 9, 13] : Largest individual maximum interval: logout – login
- [4, 11] : Smallest number of interactive users

Where options (1, 2, 3, 8, 9, 10) assume the computer will not be rebooted each night, while (5, 6, 7, 12, 13, 14) assume the computers will be rebooted. Options (1, 2, 3, 4, 5, 6, 7) assume that the current state will be taken into account (idle computers before sleeping computers) whilst options (8, 9, 10, 11, 12, 13, 14) will use computers irrespective of whether the computer is idle or sleeping.

S4: Targeting clusters closed for public use. Each computer within the university is part of a cluster with each cluster having pre-defined opening and closing hours. Here we propose selecting computers in clusters which have the greatest amount of time remaining before the cluster is reopened, thus minimising the chances of the jobs being evicted through by interactive users.

S5(i): Target less used clusters. Similar to policy S3(i) this policy targets the least used computers. Though differs by the fact that it is simpler to implement. Clusters can be selected based on the following criteria:

1. Largest individual average interval: logout - login
2. Largest individual maximum interval: logout - login
3. Smallest number of interactive users.
4. Smallest total interactive user duration
5. Smallest mean interactive user duration
6. Number of interactive users.

Condor contains the ability to suspend jobs when an interactive user logs into the computer. This allows the job to resume if the user logs out of the computer quickly after. If this interval is short enough then this will prevent the eviction of the job and allow it to continue, thus saving energy

and overhead. If the interval is long then this will increase the overhead though save on energy. We extend the notion of suspensions here to allow more fine-grained control over when a job should be suspended and when a job should be evicted.

H(initial, subsequent): Allow a job to be suspended given the *initial* policy is satisfied for the first suspension and the *subsequent* policy is satisfied for all future suspensions, otherwise the job is evicted.

The *initial* policies can be defined as:

- **None** : Jobs will be immediately evicted.
- t : Allow the job to be suspended for up to t minutes. After this time the job should be evicted. This is the default Condor policy. A small value of t allows jobs to remain active if there is a brief use by an interactive user.
- p : Allow the job to be suspended for up to p % of its current execution time. This allows jobs which have received little execution to be evicted quickly as this gives the best chance of keeping the overheads low. Whilst tasks which have received significant amounts of service are suspended for longer as their is greater impact if these are evicted.

The *subsequent* suspension policy determines if the job can be re-suspended:

- **None** : Jobs will be terminated when the second user attempts to log in.
- n : If the job has not been suspended n times already it will be suspended, Otherwise it will be evicted. This helps prevent jobs which are allocated to high turnover computers from receiving short burst of execution thus leading to high overheads.
- T : If the total time the job has been suspended is less than T then the job is suspended, otherwise it is evicted. This helps prevent jobs from spending significant amounts of time suspended and not completing.
- P : If the proportion of time that the job has been suspended is less than a given threshold P then the job can be suspended, otherwise it is evicted. This helps prevent tasks which are only achieving a small amount of progress through suspensions.

In all cases if the job can be suspended then the maximum time interval for suspension in *initial* is used.

4.4. Policy Combinations

The policies described above are not mutually exclusive. Most can be used in combination with each other. Table I indicates the groupings of policies which cannot be used in combination with each other. Policies in different policy groups can always be combined with each other.

5. SIMULATION ENVIRONMENT

In this section we describe the environment that we are simulating along with some information on the simulation written to model the real system.

The Condor high-throughput computing cluster at Newcastle University comprises 1359 student access desktops: which were running Microsoft Windows XP during the time interval of our trace

Table I. Computer Types

Policy Group name	Combinable policies	None combinable policies
Power		P1, P2, P3, P4, P5
Selection		S1, S2, S3, S4, S5
Management	M1, M2, M3, H	
Job Termination		C1, C2, C3

logs, distributed around the University Campus in 35 different computer clusters. Computer clusters may share the same physical location, with several clusters within the same room, each room having its own opening hours. These hours vary between clusters that are predominantly for teaching purposes and open during teaching hours (normally 9am till 5pm) through to 24-hour access computer clusters. The location of clusters has a significant impact on throughput of interactive users. From clusters buried deep within a particular school to those within busy thoroughfares such as the University Library.

Computers within the clusters are replaced on a five-year rolling programme with computers falling into one of three broad categories as outlined in Table II. PCs within a cluster are provisioned at the same time and will contain equivalent computing resources. Thus there is a wide variance between clusters within the University but no significant variance within them.

Whilst we expect casual use to migrate onto user owned portable devices and virtual desktops, the demand for compute/graphic intensive workstations running high-end software is, if anything, increasing. Further, these high-end applications are unlikely to migrate to virtual desktops or user owned devices due to hardware and licensing requirements, so we expect to need to maintain a pool of hardware that will also be useful for Condor for some time.

PUE values have been assigned at the cluster level with values in the range of 0.9 to 1.4. These values have not been empirically evaluated but used here to steer jobs. In most cases the cluster rooms have a low enough computer density not to require cooling giving these clusters a PUE value of 1.0. However, two clusters are located in rooms that require air conditioning, giving these a PUE of 1.4. Likewise, four clusters are based in a basement room, which is cold all year round; hence computer heat is used to offset heating requirements for the room, giving a PUE value of 0.9.

By default computers within the cluster will enter the sleep state after a given interval of inactivity. This time will depend on whether the cluster is open or not. During open hours computers will remain in the idle state for one hour before entering the sleep state whilst outside of these hours the idle interval before sleep is reduced to 15 minutes. This policy (as P3 then P4) was originally

Table II. Computer Types

Type	Cores	Speed	Power Consumption		
			Active	Idle	Sleep
Normal	2	~3Ghz	57W	40W	2W
High End	4	~3Ghz	114W	67W	3W
Legacy	2	~2Ghz	100-180W	50-80W	4W

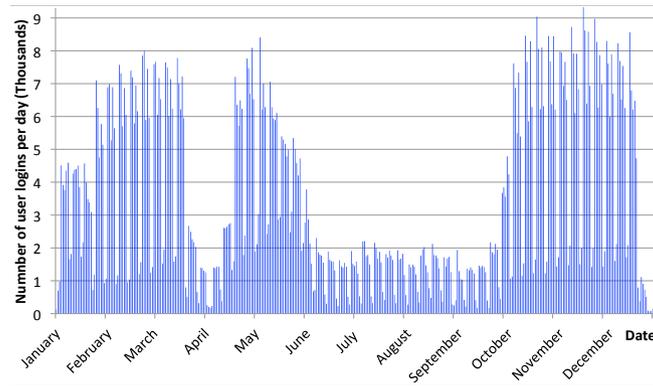


Figure 3. Interactive user logins showing seasonality

trialled under Windows XP where the time for computers to resume from the shutdown state was considerable (sleep was an unreliable option for our environment). Likewise the time interval before a Condor job could start using a computer (M1) was set to be 15 minutes during cluster opening hours and 0 minutes outside of opening hours. The latter was possible as computers would only have their states changed at these times due to Condor waking them up or a scheduled restart.

The simulation is based on trace logs generated from interactive user logins and Condor execution logs for 2010. Figure 3 illustrates the interactive logins for this period showing the high degree of seasonality within the data. It is easy to distinguish between week and weekends as well as where the three terms lie along with the vacations. This represents 1,229,820 interactive uses of the computers.

Figure 4 depicts the profile for the 532,467 job submissions made to Condor during this period. As can be seen the job submissions follow no clearly definable pattern. Note that out of these submissions 131,909 were later killed by the original Condor user or the system administrator as the jobs were not completing and wasting resources. In order to simulate these killed jobs the simulation assumes that these will be non-terminating jobs and will keep on submitting them to resources until the time at which the high-throughput user (or system administrator) terminates them. The graph is clipped on Thursday 03/06/2010 as this date had 93,000 job submissions.

For the simulations we will report on the total power consumed (in MWh) for high-throughput jobs in the period. In order to determine the effect on high-throughput users of a policy we will also

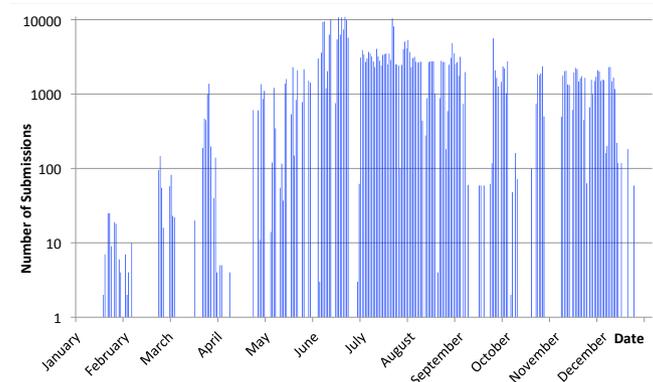


Figure 4. Condor job submission profile

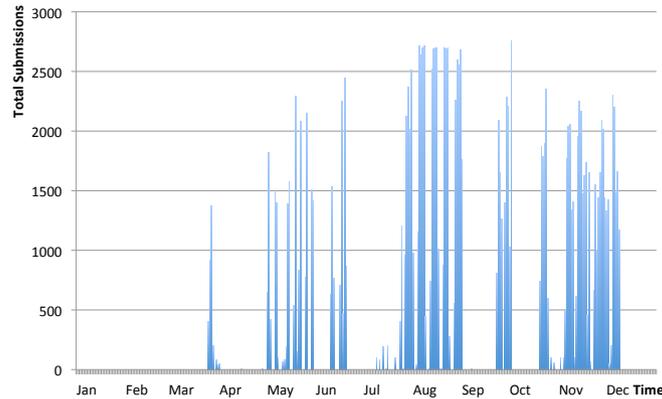


Figure 5. Total job submissions per day for one user

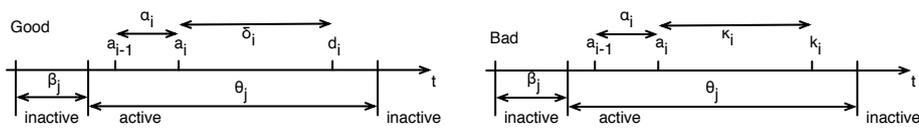


Figure 6. Timings for good and bad jobs

report the average overhead observed by jobs submitted to Condor (in minutes). Where overhead is defined to be the amount of time in excess of the execution duration of the job. Other statistics will be reported as appropriate for particular simulations. It should be noted that running Condor jobs on different hardware may lead to variations in execution times. It would however be difficult to determine how this would be affected without knowing whether the particular job was memory, CPU or IO dominant. As such the simulations ignore this effect and assume the job will require the same time to execute.

As the simulations presented here, apart from the default policy for selection of computer to run on, are entirely trace-driven only a single run of the simulation is considered. For simulations based around the default selection policy (S1) the average of 10 simulation results are reported.

We have extended our Cluster based simulation for Condor [15] to take account of the data transfer times. The *iperf* bandwidth testing software [16] was used to compute the maximum bandwidths available between computers for different payload sizes. Although bandwidth for small (less than 1Kb) of data exceeded 100Mbits/s this quickly capped out at 94.75Mbits/s. It should be noted that these are maximum bandwidth potentials, real use is likely to be less. Thus these are lower estimates of transfer times.

5.1. Generation of Synthetic high-throughput computing jobs

In order to evaluate the policies for greater workloads than those of our trace-logs we have developed a synthetic model scalable to any size. Although the overall job submissions followed no directly observable pattern (Figure 4), if we decompose this log into individual users (Figure 5 is a single user) then a bursty nature can be seen. We therefore model each user as a bursty provider of jobs to the system where they can be actively submitting jobs or inactive.

Jobs are either ‘good’ – they completed in the original trace-log or ‘bad’ – they were eventually terminated by the user. We model the probability p_i that a job i is ‘good’ as uniformly distributed

process. Figure 6 illustrates the timings for a job. We assume that each jobs inter-arrivals – α_i , are modelled by a Poisson distribution. Likewise for the user’s bursty interval lengths – active – β_j and inactive – θ_j for interval j . Where a_i is the arrival time, d_i the departure time and k_i the kill time of job i . We model the data in – ι_i and data out – σ_i as Poisson distribution.

The job duration time and time before jobs are killed follow a more normal distribution. Figure 7 illustrates the frequency of durations for the same user as in Figure 5. Although this frequency graph is trimodal it can be modelled as a normal distribution. Therefore the duration and time before jobs are killed are modelled as normal distributions with means δ_i and κ_i and standard deviations of ϱ_i and ψ_i . Table III illustrates the range for these values.

Table III. Average range for users of the Newcastle Condor Cluster

Parameter	Range
Average inactive interval (β)	7.5 – 182.5 days
Average active interval (θ)	1 – 4.4 days
Average job inter-arrival time ($\bar{\alpha}$)	22.1 – 28,800 seconds
Average job execution time (δ)	0.4 – 190 minutes
Standard deviation for execution time ($\bar{\varrho}_i$)	0 – 16,945 minutes
Average time between submission and job kill ($\bar{\kappa}$)	0.01 – 161 days
Standard deviation for time to job kill ($\bar{\psi}_i$)	0 – 2.6 days
Average data in ($\bar{\iota}$)	0.0 – 80.9MB
Average data out ($\bar{\sigma}$)	0.0 – 1037.1MB
Probability job is good	0.302 – 1

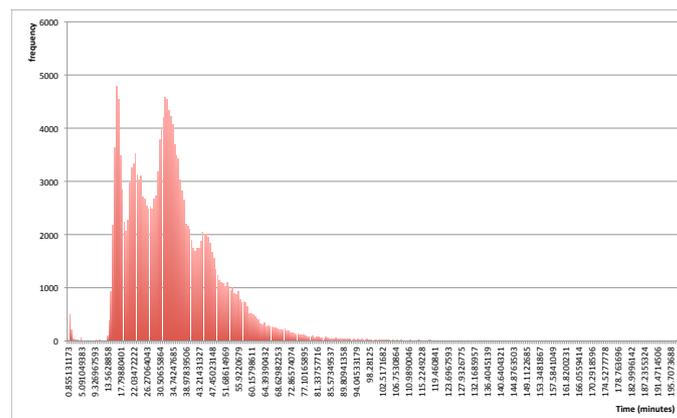


Figure 7. Job duration frequency for one user

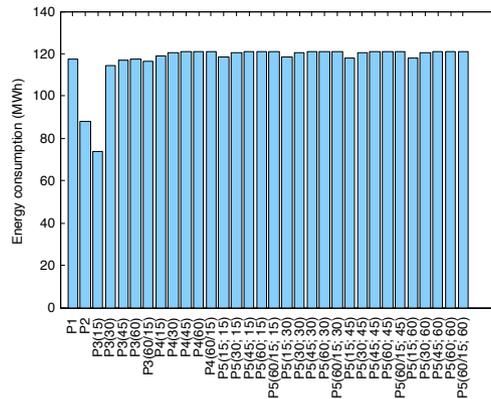


Figure 8. Power Management Policies vs. Energy Consumed

6. SIMULATIONS AND RESULTS

In this section we evaluate the previously described policies in order to assess an optimal set of policies for our cluster. These tests could easily be performed on other clusters and we believe that the general conclusions from this work will be applicable to other similar clusters. These tests are grouped into baseline tests, power management tests, computer selection tests and cluster change tests.

6.1. Baseline Evaluation

This simulation is aimed at providing point of reference for the following tests. In this test power policy P4(0) was used and only interactive users were simulated. This generated 120.7MWh of active power consumption, 33.8MWh idle time consumption and 28.5MWh of energy consumption for sleep time. The energy consumption for Condor is then calculated separately from this. If we add in the execution of Condor jobs this adds ~ 120.9 MWh of energy consumed for these jobs along with an average overhead of ~ 13.33 minutes.

In the remainder of this section we test each of the proposed policies in isolation against our default policy, to determine the effectiveness of each. Before combining the ‘best’ policies and evaluating these for both real and synthetic workloads.

6.2. Power management policies

Here we evaluate power management policies P1, P2, P3, P4 and P5. All tests were performed with selection policy S1 (default Condor selection). The amount of time before computers were allowed to go was also varied for policies P3, P4 and P5. Taking values of 15, 30, 45 and 60 minutes. The current policy of 60 minutes during open hours and 15 minutes outside was also evaluated (60;15). Figures 8 and 9 illustrate the results from these simulations. Policies P2 and P5(15,15) would appear to have the ‘best’ energy consumption result (Figure 8) however when the average overhead (Figure 9) is taken into account these policies clearly starves high-throughput users of their resources. Policy P5(15,15) would seem to be a consequence of computers going offline at the same time as the sweep happening thus leading to computers being absent for longer. The remaining policies show little

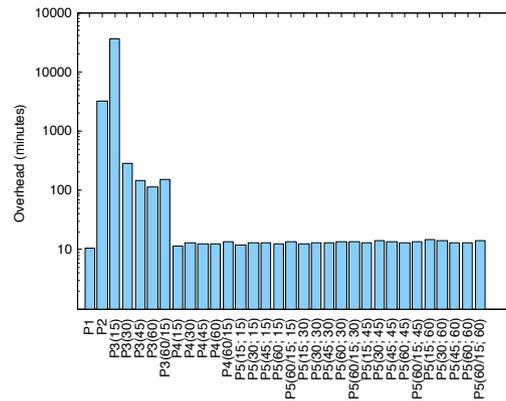


Figure 9. Power Management Policies vs. Overhead

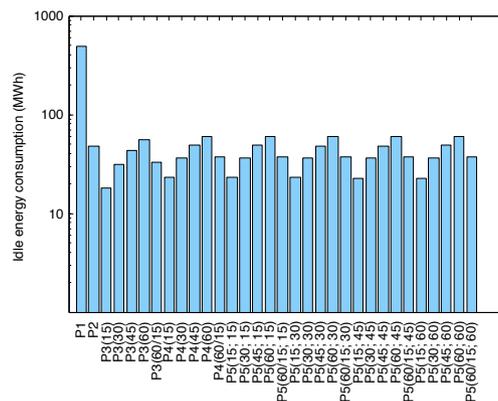


Figure 10. Power Management Policies vs. Idle Energy Consumed

significant statistical difference even from P1. Thus indicating that these policies have little impact on the high-throughput users. Although the policy of changing the time clusters are powered down has no impact on the high-throughput use of the cluster Figure 10 illustrates that this has a marked impact on the energy consumed by idle computers. As this policy can be combined with the other policies this would make sense to adopt and have a low (~ 15 minute) value.

6.3. Computer Selection policies

Here we evaluate the selection policies S1, S2, S3, S4 and S5 under power policy P4(60;15). Figures 11 and 12 shows the result of these simulations. All policies apart from S4 and S5(Largest individual average interval/1) reduce the overall energy consumed with S2 and S5(maximum interval/2) showing the best improvement. All policies apart from S3(1-7) produce no significant change to the overheads for jobs. Thus selection policies S2 and S5(maximum interval) would appear the best choice. Although policies S3(1-7) select computers with the greatest chance of being unused for the duration of the job execution the resources are selected by initial state first – idle over sleeping. Hence an idle computer with little chance of remaining idle during the job's duration will be selected over a sleeping computer which would most likely be idle for the job's duration.

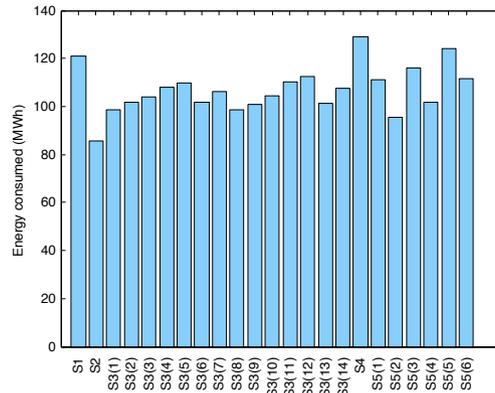


Figure 11. Computer Selection Policies vs energy consumed

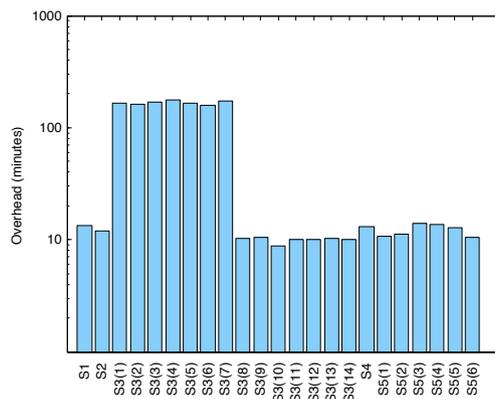


Figure 12. Computer Selection Policies vs overhead

6.4. Management Policies

Policies M1, M2, M3 and H are evaluated here with default selection policy S1 and power policy P4(60;15). Figures 11 and 12 illustrates these results for policies M1, M2 and M3. Policy M1 has little perceivable impact on the power or overhead of jobs. However, this policy does have an impact on the overall energy consumed by the whole system by increasing, by a factor of 10, the amount of energy consumed by idle computers by reducing the energy for sleeping computers when the value of n is low. This is a consequence of Condor waking up computers for short running jobs which then leaves the computer idle. Whilst for larger values of n these short jobs accumulate up and run continuously. There is a slight energy advantage in using $n = 10$ and should be selected. Policy M2 (jobs prevent reboots) provides an advantage for both energy consumed and overheads and should be used.

Policy M3 (use computers at the same time as users) is depicted for both the case where we assume that no energy charge is allocated to the Condor job (M3(NP)) and the case where we assume that there is an energy charge for using the computer (M3(P)). As we do not know the power consumption of the Condor job we assume the worst case – the Condor job is consuming all the processing power. Using the SPECpower [17] power evaluation software we have benchmarked one of the high-end computers at 117W active and 65W idle. Thus in the worst case scenario Condor

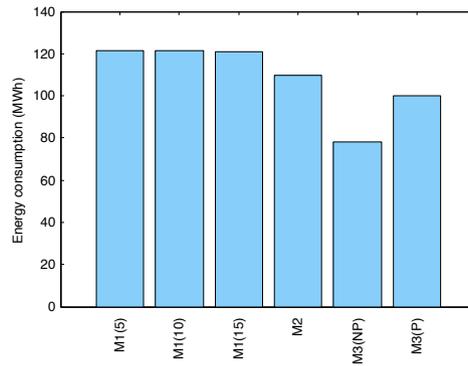


Figure 13. Management policies vs energy consumed

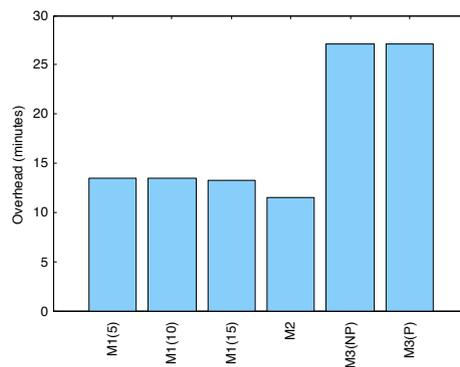


Figure 14. Management policies vs overhead

would consume 52W. Although this policy decreases the overall energy consumed it has a negative impact on the overhead. This is a consequence of ‘bad’ jobs not being evicted when users log in allowing ‘good’ jobs a chance of execution.

Figures 15, 16, 17 and 18 exemplify policy H. For the case of maximum suspension time (Figures 15 and 16) the ‘best’ policy appears to be $H(t, \text{None})$. With all other policies increasing both energy consumed and overhead. Likewise for percentage of execution time (Figures 17 and 18) there appears to be no benefit in using any policy over $H(p, \text{None})$. In fact there appears to be little benefit in using the suspension policy as it increases both energy consumed and overhead over the baseline.

It should be noted that all of the policies in this set can be combined with each other. However, the policies which show the ‘best’ chance of improvement are M1(10) and M2.

6.5. Cluster termination policies

Here we evaluate policies C1 (kill jobs after n resubmissions), C2 (dedicated computers for miscreant jobs) and C3 (timeout for jobs on dedicated computers). Figures 19 and 20 illustrate the energy consumption and overheads for these policies. Note that Figure 20 only shows the lower retry values to help distinguish the different policies. Policy C1 leads to significant numbers of good jobs being killed (defined as a job which originally completed successfully now being terminated) – Figure 21. Addition of dedicated resources (C2) leads to fewer good jobs being killed but can lead to

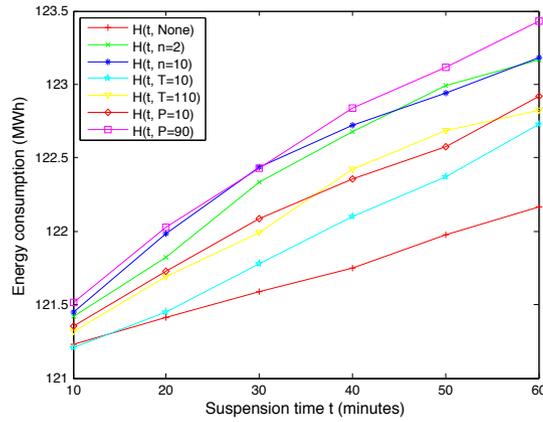


Figure 15. Suspension time vs energy consumed

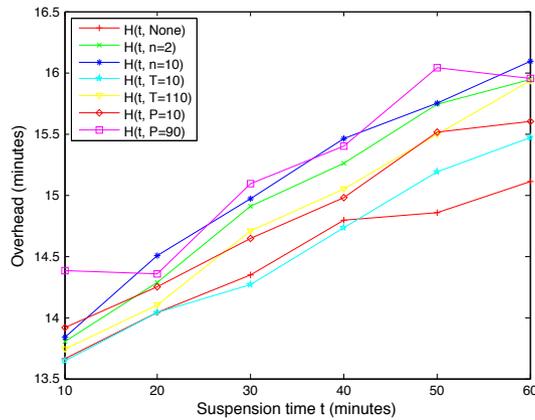


Figure 16. Suspension time vs overhead

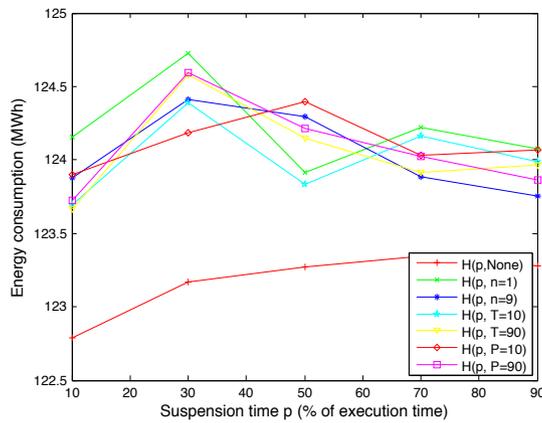


Figure 17. Suspension percentage vs energy consumed

bottlenecks for job overheads if the number of retries are low and excessive energy consumption if retries are high. By the inclusion of a time limit on dedicated resource usage we can bring the energy usage down, by keeping the retries low and preventing ‘bad’ jobs from running indefinitely

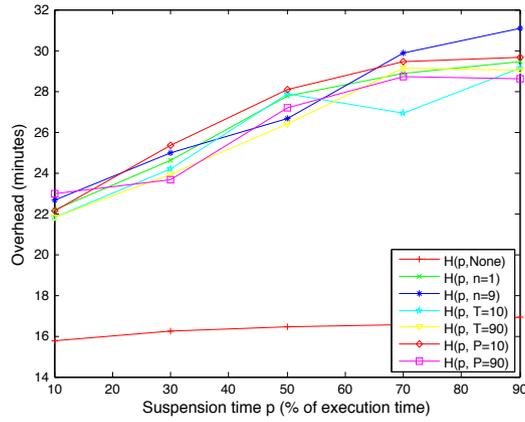


Figure 18. Suspension percentage vs overhead

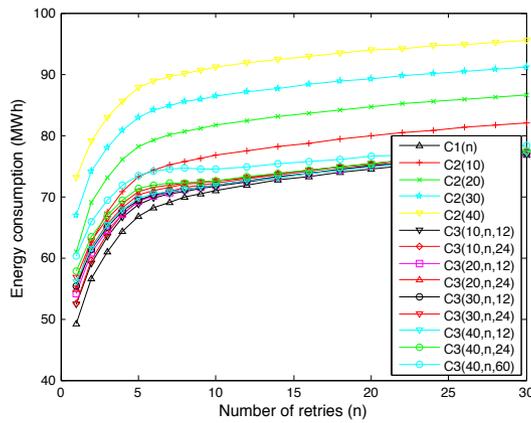


Figure 19. Job Termination Policies vs. energy consumed

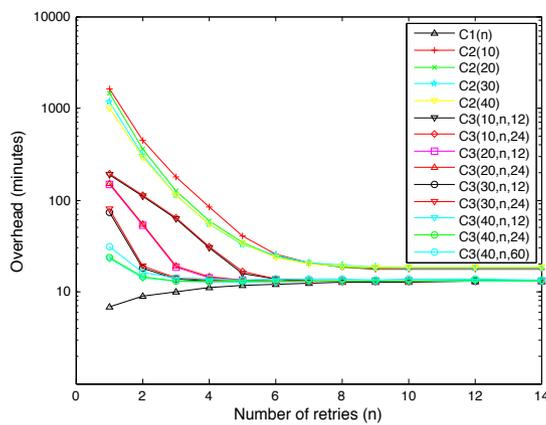


Figure 20. Job Termination Policies vs. overhead

on the dedicated resources, allowing us to still maintain good overheads and low numbers of good jobs killed. The policy C3(40;6;60) gives a good combination as it gives no good job kills.

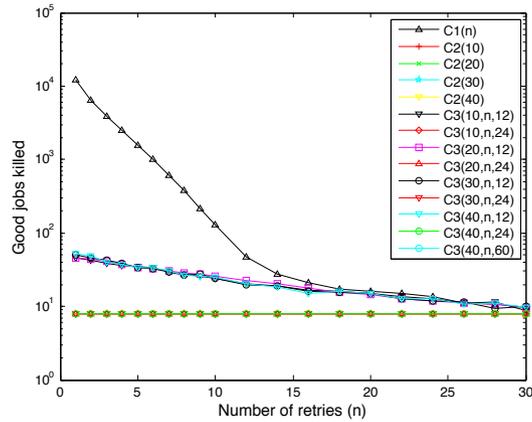


Figure 21. Job Termination Policies vs. 'good' jobs killed

6.6. Combined policies with synthetic jobs

Here we evaluate the 'best' policies identified above against larger workloads. We have identified three power policies (P3(15), P4(15) and P5(15,15) along with the selection policies (S2, S5(2)), the management policies (M1(10) and M2) and the job termination policy (C3(40,6,60)). As the management policies are not mutually exclusive we use both simultaneously here. Thus the four policy combinations are:

- com-1 = {M1; M2; P4; S2; C3(40;6;60)}
- com-2 = {M1 ; M2; P5; S2; C3(40;6;60)}
- com-3 = {M1 ; M2; P4; S5(2); C3(40;6;60)}
- com-4 = {M1 ; M2; P5; S5(2); C3(40;6;60)}

Figures 22 and 23 illustrate the effectiveness of these policies over different workloads. Note that termination policy C4(40;6;60) prevented any good jobs from being terminated. Although this is not guaranteed, the simulated workloads here exhibited this property.

All four policy sets scale consistently with increased workload with policy set com-2 showing slightly worse performance in almost all cases. The power increase in all cases is sub-linear – i.e. doubling the number of jobs does not double the energy consumed. However, overheads do increase in a greater than linear manner. Suggesting that a more stringent policy set for removing bad jobs would be beneficial for higher workloads.

7. CONCLUSION

Selection of an optimal set of policies for energy consumption across a multi-use cluster is a complicated process. Many policies have a significant impact on the power consumed, though also have a (detrimental) impact on the usability of the cluster for high-throughput users.

Power management policies P4 and P5 appear to be the most optimal policies to select. Whilst selection policy S2 has the greater impact on overhead and power consumption, with S5(2) being

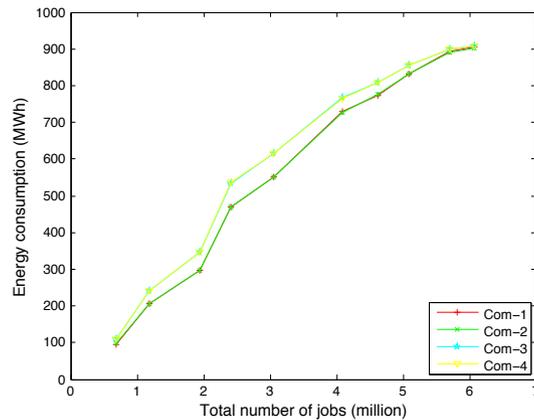


Figure 22. Combined policies vs. energy consumed

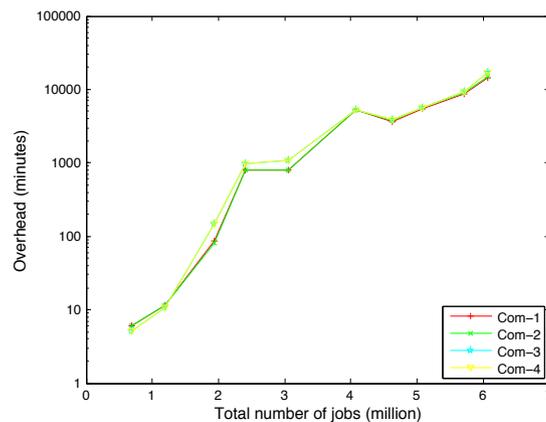


Figure 23. Combined policies vs. overhead

a close second. Merging these policies could give both advantages though computing this ordering may be difficult.

Management policy $M1(n)$ appears to have little effect, though this could be masked since a significant proportion of time the cluster is closed to interactive users where $M1(0)$ applies. Policy $M2$ has a significant impact in both power saving and job overhead, though $M3$ fails to provide a good reduction in energy and increases overheads markedly. This is due to bad jobs continuing to run on resources blocking other jobs and wasting energy. The combination of this policy with a timeout interval similar to that of $C3(m, n, t)$ could make this policy more attractive. Unfortunately suspension policy (H) fails to deliver any significant benefit.

Changing policies for the cluster (dedicated resources for evictees, postponing reboots and allowing jobs to run on the same computers as interactive users) each show the potential to save power and reduce overheads for users. The best effect is likely to come from a combination of these policies. These policies have been combined and evaluated over larger (synthetic) workloads showing that they remain similar in benefit.

The main advantage with these policies comes as they are not mutually exclusive. Combinations of these policies can be produced increasing the overall energy savings without a significant impact on users of the high-throughput resources.

The most significant energy saving that can be made is simply allowing computers to go to sleep when not needed. We have shown that changes to the cluster policy can further reduce energy consumption without significantly affecting the high-throughput users. This can lead to an energy saving of ~65MWh, ~55% of the energy currently consumed by the high-throughput system.

REFERENCES

1. McGough AS, Robinson P, Gerrard C, Haldane P, Hamlander S, Sharples D, Swan D, Wheeler S. Intelligent power management over large clusters. *International Conference on Green Computing and Communications (GreenCom2010)*, 2010.
2. Hewlett-Packard Corporation, Intel Corporation, Microsoft Corporation, Phoenix Technologies Ltd and Toshiba Corporation. ACPI Specification. <http://www.acpi.info/DOWNLOADS/ACPIspec40a.pdf>.
3. Litzkow M, Livney M, Mutka MW. Condor-a hunter of idle workstations. *8th International Conference on Distributed Computing Systems*, 1998; 104–111.
4. Minartz T, Kunkel J, Ludwig T. Simulation of power consumption of energy efficient cluster hardware. *Computer Science - Research and Development* 2010; **25**:165–175.
5. Kliazovich D, Bouvry P, Audzevich Y, Khan SU. Greencloud: A packet-level simulator of energy-aware cloud computing data centers. *GLOBECOM*, 2010; 1–5.
6. Niemi T, Kommeri J, Happonen K, Klem J, Hameri AP. Improving energy-efficiency of grid computing clusters. *Advances in Grid and Pervasive Computing, LNCS*, vol. 5529. 2009; 110–118.
7. Legrand A, Marchal L. Scheduling distributed applications: The simgrid simulation framework. *In Proceedings of the Third IEEE International Symposium on Cluster Computing and the Grid*, 2003; 138–145.
8. Buyya R, Murshed M. Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing. *Concurrency and Computation: Practice and Experience* 2002; **14**(13):1175–1220.
9. Bell WH, Cameron DG, Capozza L, Millar AP, Stockinger K, Zini F. Optorsim - a grid simulator for studying dynamic data replication strategies. *International Journal of High Performance Computing Applications* 2003; .
10. Méndez V, Garca F. Sicogrid: A complete grid simulator for scheduling and algorithmical research, with emergent artificial intelligence data algorithms.
11. The Condor Team. Condor manual. <http://www.cs.wisc.edu/condor/manual/>. Oct 2011, University of Wisconsin.
12. Smith IC. Experiences with running matlab applications on a power-saving condor pool. http://www.liv.ac.uk/csd/escience/condor/cardiff_condor.pdf. Oct. 2009.
13. Arjuna Technologies Ltd. Agility Cloud Computing Platform. <http://www.arjuna.com/agility>. Oct. 2011.
14. MathWorks. MATLAB. <http://www.mathworks.co.uk/>.
15. McGough AS, Gerrard C, Noble J, Robinson P, Wheeler S. Analysis of power-saving techniques over a large multi-use cluster. *International Conference on Cloud and Green Computing (CGC2011)*, 2011.
16. Sourceforge project. The iperf project. <http://iperf.sourceforge.net/>.
17. Standard Performance Evaluation Corporation. SPECpower_ssj2008. http://www.spec.org/power_ssj2008/.