Trace-driven simulation for energy consumption in High Throughput Computing systems

Matthew Forshaw, Nigel Thomas
School of Computing Science
Newcastle University
Newcastle upon Tyne, NE1 7RU
{m.j.forshaw,nigel.thomas}@ncl.ac.uk

A. Stephen McGough
School of Engineering and Computing Sciences
Durham University
Durham
stephen.mcgough@durham.ac.uk

***Abstract*—High Throughput Computing (HTC) is a powerful paradigm allowing vast quantities of independent work to be performed simultaneously. However, until recently little evaluation has been performed on the energy impact of HTC. Many organisations now seek to minimise energy consumption across their IT infrastructure though it is unclear how this will affect the usability of HTC systems. We present here HTC-Sim, a simulation system which allows the evaluation of different energy reduction policies across an HTC system comprising a collection of computational resources dedicated to HTC work and resources provided through cycle scavenging – a Desktop Grid. We demonstrate that our simulation software scales linearly with increasing HTC workload.**

I. Introduction

Modern computational power allows researchers to perform work hitherto unimaginable. This is often achieved through the processing of vast quantities of data (Big Data), performing large scale simulations or ensembles of smaller simulations. However, our desire to solve such problems has now far-outstripped the computational power of a single computer. Parallel computing, where multiple processing units are employed in the solving of a single piece of work, is a common solution to such problems. Where this work can be sub-divided into separate jobs that can be run independently of each other we refer to this as a pleasingly parallel problem and solve it using High Throughput Computing (HTC). Many HTC systems exist, such as HTCondor [1] and BOINC [2], with these systems being used to help solve research problems from small scale up to grand research challenges.

Traditionally HTC systems were provisioned as either dedicated resources or as a shared facility (often referred to as a Desktop Grid) with resources powered up all the time either servicing jobs or sitting idle. The performability and reliability of such systems is generally well understood [3]. However, with the power consumption of western european data centres alone estimated at 56 TWh/year in 2007 and projected to double by 2020 [4] it is imperative to improve energy efficiency of IT operations. Due to their nature, HTC systems would appear to be a prime candidate for such savings.

Aggressive power management policies, over the resources which constitute an HTC system, are often proposed, though these policies could have significant impact not only on the energy consumption but also performance, reliability and availability of resources for HTC users. Placing idle resources into a sleep state too rapidly could lead to HTC resource starvation, while weak policies may offer little energy savings.

It is therefore highly desirable to determine the 'best' set of energy conservation policies which can be applied to both the HTC system and the underlying hardware. Controlling such factors as when idle resources are sent to sleep, when to wake up resources, the selection of the resource to use in order to minimise energy consumption or how to deal with jobs which fail to complete. This is particularly important for Desktop Grids, with priority for interactive users, as job eviction does not imply that the job cannot complete with more time on a different resource.

In previous work [5] we proposed an architecture capable of managing policies for power management of resources in a live HTC system. 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: running the system under the new policy for a significant amount of time to ensure statistical relevance; detailed logging to determine energy consumption and monitoring of the high-throughput architecture is required; and a danger that changes could have unpredicted (negative) consequences. This leads to making minor modifications to the policy set where we are pretty sure that the impact on users will be low; significant changes being considered too dangerous.

Two alternatives exist: a test environment or a simulation. Test environments remove the need for site-wide monitoring and do not affect the production system, however time is required to evaluate changes and we need to justify how results would map to the whole system. Instead we present here HTC-Sim a Java based trace-driven simulation we have been developing as part of our work in energy saving for HTC systems. The simulation system allows for the quantifiable, and quick, evaluation of different policies against the same workload and interactive user patterns, if present.

The simulation system allows the modelling of energy consumption and performance characteristics of the HTC system.

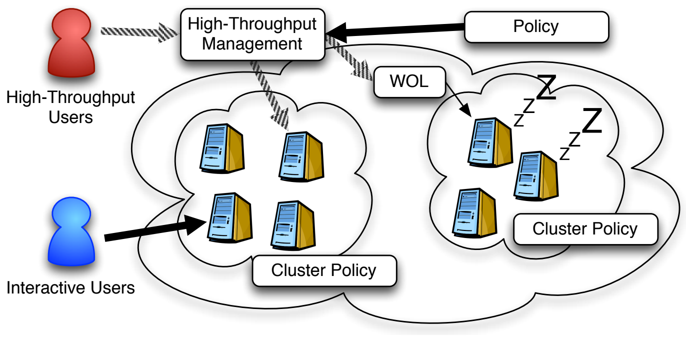

Fig. 1. Model of an HTC system and multi-use environment

Thus it can be seen as a powerful tool for administrators to evaluate new policies as well as the impact of changes to the infrastructure itself.

The overall model for HTC-Sim is shown in Figure 1, where two types of user can interact with the system – HTC users and interactive users. These are handled through historical trace logs for both user types. In both cases an empty log file indicates absence of that user type. Interactive user trace logs contain the login and logout time along with the resource used – it is assumed that this is a fixed interaction. However, for the HTC workload only the job submission time and the execution time are considered – the execution start time and resource used may change due to the active policy set. Resources within the system are grouped into *clusters*, each representing a set of homogeneous resources under the same policy set. In this way we can model both sets of resources purchased together or resources co-located and acting under identical policies. The HTC system has its own policy set.

The remainder of this paper is structured as follows. In Section II we discuss related work. In Section III we provide details of the simulation model. We present a case study of using our simulation with a HTCondor cluster in Section IV. Performance evaluation of the HTC-Sim is presented in section V before we discuss work we have performed with the simulation to date in section VI. Conclusions and future plans for extending HTC-Sim are presented in Section VII.

II. Related Work

A. Simulation frameworks

A number of Grid and Cluster level simulators exist including SimGrid [6], GridSim [7], and OptorSim [8] though these focus more at the resource selection process both within clusters and between clusters and lack the modelling of energy. More recently Cloud simulators have been proposed which are capable of modelling the tradeoff between not only cost and Quality of Service, but also energy consumption. These include CloudSim [9], GreenCloud [10], and MDCSim [11]. Our simulation environment is novel in its ability to model multi-use clusters with interactive users, and use real-world workload traces. We further support modelling of fault tolerance mechanisms including checkpointing and migration.

B. Energy efficient high throughput computing

The evaluation of techniques to reduce energy consumption within high throughput computing environments is receiving increasing attention. Minartz *et al* [12] explore energy saving through the dynamic provisioning of resources within a high performance computing cluster, while Niemi *et al* [13] demonstrate energy savings through the consolidation of multiple jobs onto the same hardware. Ponciano *et al* evaluate strategies for energy-aware resource provisioning and job allocation within opportunistic grids [14]. Terzopoulos *et al* investigate the use of Dynamic Voltage Scaling techniques to reduce energy consumption in a heterogeneous cluster to conform to power budgets [15] imposed by infrastructure. These could be modelled through our simulation system.

C. Power modeling

The energy consumption of server and commodity hardware has been studied extensively. Early works leveraged low-level metrics such as performance counters [16] when developing predictive models of energy consumption. Often requiring significant architecture knowledge and not generalisable to other hardware, nor scalable to entire systems. Linear correlation between energy consumption and CPU utilisation allows this to be used as an energy predictor [17], while others derive linear regression models based on utilisation of CPU, memory and storage subsystems for single servers [18], [19], groups of systems [17], [20] and virtualised environments [21]. Again these could be modelled through our simulation system.

III. System Model

We introduce our generic model of the entities and resources within a HTC system along with our metrics for user impact, energy, cost and environmental implications.

A. Compute resources

We model compute resources as being either dedicated - whether local or cloud infrastructure - or multi-use cluster machines shared with interactive users. We model a number of characteristics for machines, namely architecture type, operating system, performance measures (e.g. CPU speed, number of cores, memory) and energy profile. We further allow users of the simulation to specify custom attributes for machines which are specific to the environment which they are modelling.

We adopt the SPECpower [22] model for energy consumption within a system. Here discrete values for CPU load are equated with specific energy consumption levels. This allows the energy consumption of a resource to be derived from the current CPU load, if known. As shown in the state transition diagram in Figure 2, resources are modelled as being in one of three states based on the ACPI specification [23]:

- **Active**: in use 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. This state has much lower energy consumption than the active state –

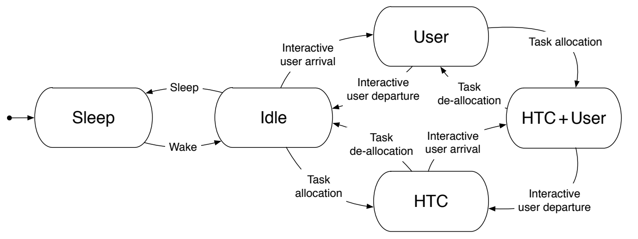

Fig. 2. State transition diagram for an HTC resource

typically having around 5-10% CPU load. This is also ACPI state 0.

- **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. ACPI state S3. The CPU is inactive consuming only a base level of energy.

B. Interactive user sessions

We model interactive user sessions as a tuple $\langle s_i, c, u, e_i \rangle$ where s_i and e_i are the login and logout timestamps respectively, c is the name of the computer, and u is a hash of the interactive users identity. Hashing of the user identifier provides anonymity to the user, while allowing us to correlate multiple sessions from a particular user.

C. Cluster

We group resources into 'clusters' defined as a homogeneous group of machines, whose specifications are identical, provisioned at the same time, co-located in the same physical space, and governed by the same operating policies. The Power Usage Effectiveness (PUE) [24] of the cluster can also be taken into account here. We model the changing behaviour of cluster machines over time. Factors include: times of scheduled reboots, whether HTC jobs are currently permitted, whether machines are currently available for use by interactive users, whether HTC jobs are allowed to run on a machine currently occupied by an interactive user, how long must a resource remain idle before transitioning into a low power state, and how long after a resource enters the idle state does it become available to run HTC jobs.

We are further able to model *'special'* events through the course of the simulation where the policies enacted on the cluster may vary. Examples of this include clusters being closed for upgrades, different policies for different days of the week, or bank holidays.

D. HTC Job

The HTC workload comprises of jobs which may be part of a batch. We define a job by the tuple $\langle j, b, q, d, h, e, u, d \rangle$, where j is the identifier of a job (or batch of jobs), b is the identifier of a job within a batch (if present), q is the time the job was submitted into the system, d was the job duration, h is the hash of the user who submitted the job, e is the HTC result state of running the job (either 'success' or 'terminated') and u, d represent the data transfer to and from the resource which ran the job. Note that if a job was terminated then f represents the time that the job termination was submitted. Although most HTC systems can provide much more information on the jobs which were run these are the core elements currently used within the simulation.

Each job will transition through a number of states as depicted in Figure 3. Jobs arriving into the system will be initially queued, though if possible they will be allocated immediately to a resource and enter the running state. In the ideal case the running job will finish without any further state transitions. However, if an interactive user takes possession of a resource then the job will enter a suspend state where execution is temporarily suspended – in the hope that the user will leave soon afterward – after which the job can resume running. If the suspension time becomes too great then the job will be evicted back to the queue. If checkpointing is being used then jobs will be checkpointed at intervals defined by policy. Jobs may be terminated at any time in which case they end up in the final 'Job Removed' state.

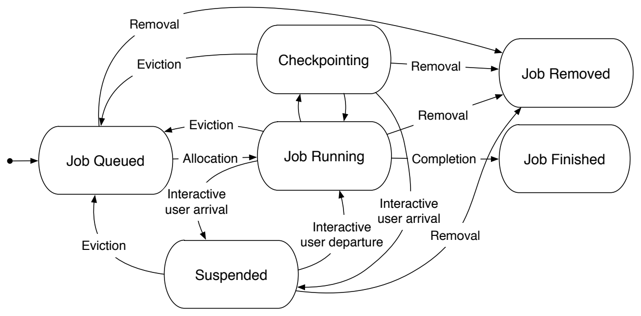

Fig. 3. State transition diagram for a job within an HTC system

E. Policy decisions - HTC

A number of common policy decisions exist within HTC systems. We discuss those which have already been built into our simulation model here:

1) Resource allocation: Given a set of available resources, the HTC system must select the most appropriate resources to optimise the required metrics. These may include random allocation, lowest energy consumption, least chance of eviction, or fastest resource. Further discussion of these policies can be found in [25], [26].

2) Job resubmission: In an system where jobs can be evicted through activities outside of its own control (reboots and interactive users) the decision of whether to try and re-submit a job which has previously been evicted needs to be made. This is non-trivial as a job which has been evicted many times may indicate that it is 'broken' and will never complete or might just indicate that the job has been unfortunate in its previous allocation to resources [27].

3) Reboots (deferral): Many Desktop Grid installations have nightly reboot policies. Given that the best time for running HTC workloads tends to be at night the ability for HTC jobs to defer these reboots can significantly improve the chance of jobs completing [25].

4) Suspension: Suspending jobs offers great potential for 'saving' the effort already exerted on a job. However, if the suspension timeout is too short then this benefit can be lost, whilst if the timeout is too long then a significant penalty is imposed on the time a job takes to complete [25].

5) Checkpointing: Checkpointing can save both time and energy by allowing jobs which are evicted to resume from the last checkpoint. However, as the process of checkpointing consumes both time and energy a careful balance is required to minimise energy consumption [28].

F. Policy decisions - Infrastructure

A number of policy decisions can be made for the underlying infrastructure [25], these include:

1) Time before HTC usage: Once a computer becomes idle it is a potential target for HTC work. However, in busy clusters a logout could be quickly followed by a login causing a job eviction. Therefore allowing some time between user logouts and HTC use is desirable.

2) Time to sleep: Energy is saved by sending resources to sleep as soon as they become idle. However, this increases the time for HTC jobs as the resources need to return from the sleeping state first. This is exacerbated if resources are required to be idle for a set amount of time before they can be used for HTC work.

3) Waking up resources: If the HTC system can not wake up resources then this can lead to resource starvation once the resources have gone to sleep. Likewise if they can wake up computers then this leads to potentially more energy usage.

4) Allow HTC usage: At busy times of day it may be desirable to disable HTC workload on specific clusters.

G. Metrics

When evaluating proposed policies, a number of metrics are of particular interest, providing insight into the performance, energy consumption and cost of operation. Below we outline the range of metrics currently supported by HTC-Sim.

1) Performance: is measured as average job overhead - defined as the time difference between the job entering and departing the system, and the actual job execution time.

2) Energy consumption: reporting fine grained energy consumption results, at per-computer, cluster and system levels. Providing a breakdown of energy consumption for each state, e.g. sleep, idle, HTC and/or interactive user. Total energy consumption is calculated as follows:

$$\sum_{c=0}^{n}\sum_{p=0}^{m} t_{c,p}E_{c,p} \qquad (1)$$

where n is the number of computers, m is the number of power states, $t_{c,p}$ is the time spent by computer c in state p and $E_{c,p}$ is the energy consumed by computer c in state p.

In the case of resources based in data centres / machine rooms, we utilise the Power Usage Effectiveness (PUE) [24] value for the environment, describing the ratio of power consumed by compute resources to the power consumed by the cooling and lighting infrastructure to support the resources. It is important to note that PUE values may not legitimately be applied to desktop machines based on users' clusters due to the multi-use nature of the environment in which the machines reside, and variations introduced by user occupancy.

3) Good jobs terminated: Policies governing the resubmission of evicted jobs may lead to good jobs being terminated.

4) Data transfer: is often a significant overhead. This is particularly evident for jobs with large datasets, or when using checkpointing. The simulation models the bandwidth available between nodes, imposing time delays on data ingress/egress.

5) Cluster utilisation and throughput: Policies such as fault tolerance and replication have the potential for significant impact on throughput and overall cluster utilisation. We report utilisation both in terms of the HTC workload in isolation, and also including interactive user load, and report measures of average and peak throughput.

6) Cost and environmental impact: It is insufficient to evaluate energy consumption and performance of policies without also considering their implications for cost. We model electricity cost per kWh, and a carbon emissions charge for each kilogram of CO2 produced from energy [29] (currently £16 per metric tonne in the UK). These figures may be specified at a system- or cluster-specific level to reflect the costs associated with the users' infrastructure, and any cost differences in federated and cloud contexts. We have in previous works extended the energy model to account for additional costs including the hardware and network infrastructure [30].

IV. Case Study of HTCondor

In this section we validate our simulation environment by modelling the HTCondor deployment at Newcastle University and use the simulation environment to explore a set of simple resource selection policies. We also discuss the process of obtaining interactive user and HTCondor workload trace data across a twelve month period to drive the simulation.

A. Newcastle University HTCondor pool

In 2010 the Newcastle University's HTCondor pool comprised $\sim$1400 desktop computers spread through 35 clusters on campus. The opening hours of these clusters varied, with some respecting office hours, and others available for use 24 hours a day. Clusters may belong to a particular department within the University and serve a particular subset of users, or may be part of a common area such as the University Library or Students' Union building. Computers within the clusters are replaced on a four-year rolling programme with computers falling into one of three broad categories as outlined in Table I. In this work we lack resource utilisation information for the HTC worker nodes, so adopt a power model employing easily obtained manufacturer*'nameplate'* power consumption values for each of the ACPI states.

The University has a policy to minimise energy consumption on all computational infrastructure which has been in place for a number of years. Hence the 'Normal' computers have been chosen to be energy efficient. 'High End' computers are provisioned for courses requiring large computational

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

TABLE I
COMPUTER TYPES

and/or rendering requirements such as CAD or video editing, as such they have higher energy requirements. 'Legacy' computers pre-date the policy of purchasing energy efficient computers and are also the oldest equipment within the infrastructure. All computers 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 clusters.

These computer clusters are provisioned for the needs of the primary (interactive) users of the system. Students generally demand Windows-based machines so the proportion of resources capable of checkpointing (i.e. Linux) is limited. At Newcastle University, Linux-based machines constitute only ∼5% of resources available to HTCondor.

All cluster machines within the pool reboot between 3am and 5am each day to install new software, perform updates and install patches. The reboot also helps to clear any temporary faults which may be present on the machine.

B. HTCondor-specifics

We extend our generalised simulation environment to model the operation of an HTCondor environment. HTCondor uses ClassAds [31] to define jobs. A ClassAd is a name / value pair document containing all information about a given job. A ClassAd can contain any number of element pairs, our system producing over 50, however, there are only currently nine elements we require for our simulations. Table II maps these to characteristics of a job which we identify in Section III-D. Note that `JobStatus` can have values here of 4 for completed jobs and 3 for terminated jobs. Note also that the computation for d neglects the fact that jobs can accumulate time through suspensions which would be included here. This can easily be removed by subtracting `CumulativeSuspensionTime`.

HTCondor provides powerful resource matching through the 'Matchmaker' which takes in two ClassAd pairs namely `Requirements` and `Rank`. Requirements is used to indicate characteristics which must be present on a resource for successful matching, such as type of operating system and minimum memory, whilst rank indicates how to order all those resources which match the requirements – with the top-raking resource being used. As our main intention here is comparison of energy consumption and overheads, and `Requirements` and `Rank` were almost completely unused in our log [5], we have ignored this information here. However, it would not be difficult to extend the resource allocation code to take this into account.

Job characteristic	Tuple term	HTCondor parameter or expression
Job identifier	j	`ClusterId`
Batch identifier	b	`ProcId`
Submission time	q	`QDate`
Job duration	d	`EnteredCurrentStatus -JobCurrentStartDate`
Owner	h	`Owner`
Result state	e	`JobStatus`
Data transfer in	u	`BytesSent`
Data transfer out	d	`BytesRecvd`

TABLE II
JOB CHARACTERISTICS TO HTCONDOR MAPPINGS

C. Preparing User logs

Interactive logins on resources at Newcastle University are handled through a central Managed Desktop Service (MDS). By taking a dump of the user logins and user logouts from 2010 we are able to construct an amalgamated user trace log. Unfortunately the MDS dumps are separate for logins and logouts and each file can contain duplicate records – both identical in time and separated by a few milliseconds. This is a consequence of the login to the resource and the mounting of remote user file-space. Further to this the records are not generated in chronological order. We have developed a tool which is able to remove the duplicates, match logins to logouts and order the trace log. A further complication arises in the case where a computer crashes or is powered off manually during a logged in session. In this case there will be no corresponding logout. As this accounted for less than 0.1% of the trace log these were ignored.

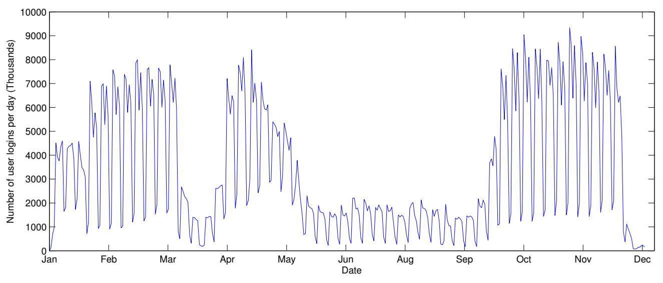

Fig. 4. Interactive user activity trace for 2010

Figure 4 illustrates the user profile from 2010, representing 1,229,820 user logins. It is easy to see the weekly cycle of computer usage, with lower usage at the weekends, along with term patterns indicating when the easter and summer breaks occurred. We currently do not possess resource utilisation information from user sessions therefore assume 100% resource utilisation of the computers whilst users are active.

D. Preparing HTCondor logs

HTCondor collects historical logs of jobs which have either completed or been terminated. This can be extracted with the `condor_history -long` command. However, this history may only contain the previous N jobs (where N

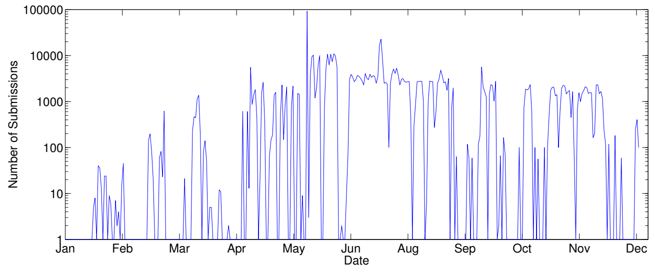

Fig. 5. HTCondor workload trace for 2010

is configurable in HTCondor) and the jobs are ordered by completion rather than submission time. In order to overcome the former a regular capture of the history can be performed, however, this may lead to duplicates. To solve this and the ordering of records we have produced a tool which orders jobs by submission time and removes duplicates. The simulation itself is then able to read the processed HTCondor log directly through an HTCondor translator.

Figure 5 illustrates the number of jobs submitted each day during 2010. In total 561,851 jobs were submitted, with a mean job submission rate of 1,454 jobs per day. There is no clearly visible pattern to this trace log.

Furthermore, since December 2012 we have extend our data collection to include event logs which include additional information including periodic memory and disk utilisation information throughout a jobs execution, and complete logs for resource re-allocation, suspension and checkpointing. This fine-grained event logging is typically only provided to the submitting user of a job, but centralised collection of this data may be enabled by including the following options in an HTCondor configuration.

```
EVENT_LOG = /some/file/path
EVENT_LOG_USE_XML = True
EVENT_LOG_MAX_SIZE = 52428800
EVENT_LOG_MAX_ROTATIONS = 3
```

The HTCondor log files comprising our dataset were collected using Condor v6.6, but our simulation remains compatible with current versions of HTCondor (currently v8.1.6).

To facilitate the sharing of HTCondor traces across organisational boundaries, we provide tooling support to automatically sanitise logs obtained from running systems, removing sensitive or personally identifiable information. Fields such as job owner and executable name are replaced with hashes to facilitate more detailed analysis of workload traces.

E. Example policies: Energy-aware resource allocation

Here we demonstrate the use of HTC-Sim by evaluating a class of energy-efficient resource allocation strategies. The efficacy of these policies is measured in terms of their impact both on average job overhead and total energy consumption.

S1: HTCondor default: random resource selection favouring powered up computers.

S2: Target the most energy efficient computers.

S3(i): Target computers with the least interactive user activity, ranked by; *a)* largest average inter-user interval, *b)* smallest number of interactive users.

S4: Target clusters closed for use by interactive users.

S5(i): Target less used clusters, ranked by; *a)* smallest total interactive user duration, *b)* smallest mean interactive user duration.

S6: A policy observing the number of interactive user arrivals to each cluster across a sliding window of Δ minutes, with arriving jobs allocated to resources ordered by availability. This policy may be expressed as:

$$\min_{c \in C} \left\{ |E_{c,t,\Delta}| \right\} \tag{2}$$

where $E_{c,t,\Delta}$ is the set of interactive user sessions starting in cluster c during the time frame $[t - \Delta, t)$, C is the set of all clusters and t is the current time.

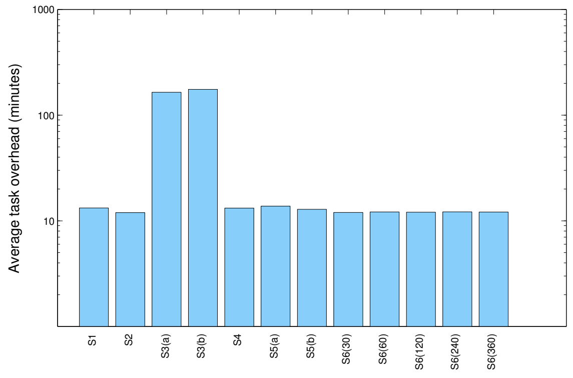

Fig. 6. Overhead results from exemplar policy

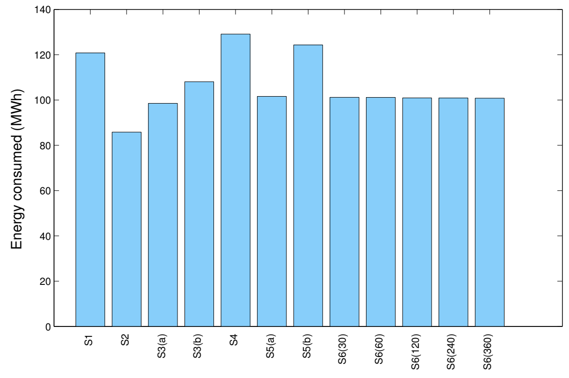

Fig. 7. Energy consumption results from exemplar policy

Figure 6 shows the average overheads for the previously define policies. Policy **S3** has a large detrimental impact on overheads of jobs whilst all other policies have little impact – with **S2**, **S5(b)** and **S6** having slightly lower overheads. We observe that policy **S6** is capable of achieving savings comparable with **S5** which assumes perfect knowledge, with sliding window size having little impact. Figure 7 shows that the energy is lowest for policy **S2** (target energy efficient computers), making this the best policy to use.

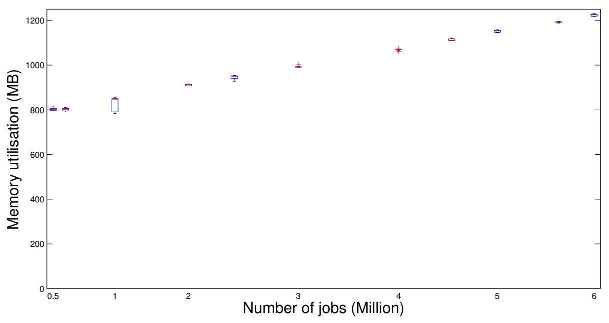

Fig. 8. Maximum memory footprint

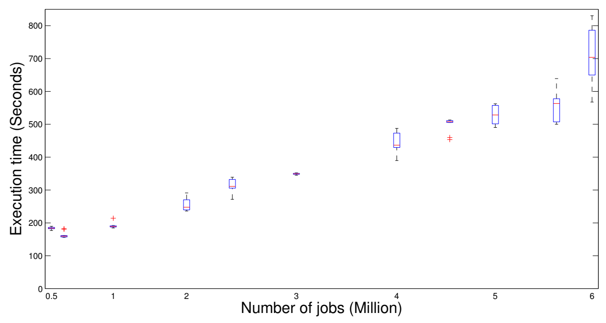

Fig. 9. Execution time

V. Performance Evaluation

Here we evaluate the performance of our simulation software and justify its applicability to arbitrary sized HTC data sets. We evaluate this in terms of the wall-clock time to run the simulation and the maximum memory footprint. The timing for a simulation and the memory footprint will be a direct consequence of the policy set being evaluated. For example a simulation such as **S6** which holds a sliding window of prior user logins will require more memory to maintain this set along with more time to process the set than a simulation based solely on random resource selection. We therefore present figures here for simulations based on policy **S1**.

Each simulation was run on a machine with an Intel Core i7 860 2.80GHz processor with 4GB RAM and 500GB 7,200RPM Western Digital Blue hard drive, running the Fedora 19 operating system. Results are based on ten simulation runs using different machines to reduce random effects.

Running our real historical trace log of HTCondor workload requires an average of 3:03 minutes. Running the simulation without the HTCondor requires 2:06 minutes, whist running without interactive users (representing a dedicated cluster) requires 1:13 minutes. Note that you cannot just sum these two times to give the overall simulation time due to simulation book-keeping and the processing of cluster events such as computer reboots and clusters opening and closing. The memory footprint for these simulations are 802MB, 750MB and 795MB respectively. The higher memory footprint from the HTCondor only simulation most likely a consequence of the larger ClassAds log file.

In order to evaluate the scalability of our simulation software we investigate the execution time and memory footprint when running larger (synthetic) workloads [25] – over ten times our real workload ($\sim$six million jobs). Figures 8 and 9 show the memory footprint and execution times respectively for both our original simulation and synthetic trace logs. In both cases the memory / time increases linearly with workload indicating the simulation scales well with workload. The only exception to this is the execution time for the largest synthetic workload. However, as this requires a memory footprint close to the normal Java memory allocation this is likely to be a consequence of aggressive garbage collection.

VI. Prior use of HTC-Sim

We have been developing HTC-Sim for three years, using it as a tool in order to evaluate the power-consumption of HTC workload based at Newcastle University. Initial work [25] investigated different resource selection algorithms, simple polices to deal with jobs which are repeatedly evicted from resources and cluster-based polices for determining when computers should be sent to sleep.

The most significant energy consumption within a Desktop Grid HTC cluster is normally owing to jobs which are evicted due to interactive users taking control back from a resource running a job. When jobs were evicted we called them miscreant jobs. As such we used HTC-Sim to evaluate a number of polices to reduce wasted energy in these circumstances [27]. We evaluated miscreant policies based on last execution time, total number of evictions and the reasons for job eviction. This lead to a potential saving of 50% of the energy for a HTC cluster when only considering evictions due to resource reboots. As this still left significant energy consumption for miscreant jobs we are now evaluating two different approaches to energy saving – those of checkpointing and migration of jobs to different resources and more intelligent job placement using Reinforcement Learning [32].

Checkpoint and Migration allows the current state of a running job to be stored and the job to resume execution from that point. In the case where a job is evicted by interactive user this saves both execution time as the jobs need not restart from the beginning but also energy as effort is not expended repeating the previously performed work. However, careful balancing is required in order to determine how often checkpointing should be performed as too many checkpoints will waste time and energy in performing the checkpoints which will not be used – including time and energy required to move checkpoints to a new resource, whilst too few checkpoints will require more work to be repeated – re-doing the work performed between the last checkpoint and the point of eviction [28].

Significant energy can be saved by placing work onto a resource which will not be used by an interactive user before the job completes. However, this is not possible to compute a priori as the times when an interactive user will login nor the execution times for jobs can be known at the time of resource selection. Analysis of interactive user trace logs shows a high degree of seasonality at the level of: within a day, across days within a week and between clusters within an institution. These patterns are, however, quite complicated hence we evaluate the use of Reinforcement Learning in order to determine when and where jobs should be allocated within the institution in order to minimise the number of evictions and minimise the energy consumption. Through this work we have shown that it is possible to save up to 53% of the wasted energy within an institution [26].

A modified version of the simulation was used for evaluating the cost implications for running work on the cloud [30] – using the same HTC workload trace log. These two simulation models were then brought together in order to compare the cost of running workloads on the Cloud versus running the same workload on a Desktop Grid.

Alongside developing our simulation of a HTC cluster we have also been developing an extension to the job scheduling and management system within HTCondor [1] allowing us to deploy a number of our developed polices [5]. Our simulation software is allowing us to build up confidence in our policy sets before deploying them to a production cluster.

VII. Conclusions and Future Work

We have presented details of HTC-Sim for simulating High Throughput Computing systems comprising both dedicated resources and resources shared with interactive users. We have presented the core model of the simulation along with discussion of the trace logs required and the methods needed to produce such logs, demonstrating how a set of resource selection policies can be tested using the simulation software. Though we focus on the modeling of our HTCondor system, our simulation base and system model is easily generalisable to other HTC systems.

We evaluate the impact of running the simulation software both in terms of memory footprint and execution time and show, through the use of synthetic trace logs that the simulation software scales linearly in both memory and execution time as the number of jobs to simulate increases. Thus making this an applicable tool to use with other simulation workloads.

We are now investigating more advanced policy in order to further reduce the energy consumption whilst maintaining low overheads. We are also extending the simulation base which will allow for federation of HTC systems both on owned resources and resources obtained through Cloud bursting.

Acknowledgment

We thank the authors of previous papers culminating in HTC-Sim. Newcastle University: B.Allen, C.Gerrard, P.Haldane, J.Noble, D.Sharples & D.Swan; Arjuna Technologies Ltd: S.Hamlander & S.Wheater; Red Hat Inc: P.Robinson.

References

[1] M. Litzkow, M. Livney, and M. W. Mutka, "Condor-a hunter of idle workstations," ser. ICDCS '88, 1998, pp. 104–111.
[2] D. P. Anderson, "Boinc: A system for public-resource computing and storage," in *Grid Computing, 2004*. IEEE, 2004, pp. 4–10.
[3] S. Jarvis, N. Thomas, and A. van Moorsel, "Open issues in grid performability," *IJSSST*, vol. 5, no. 5, pp. 3–12, 2004.
[4] P. Bertoldi and B. Anatasiu, "Electricity Consumption and Efficiency Trends in European Union Status Report 2009."
[5] A. McGough, C. Gerrard, P. Haldane, D. Sharples, D. Swan, P. Robinson, S. Hamlander, and S. Wheater, "Intelligent Power Management Over Large Clusters," in *CPSCom*, 2010, pp. 88–95.
[6] A. Legrand and L. Marchal, "Scheduling distributed applications: The simgrid simulation framework," in *CCGrid*, 2003, pp. 138–145.
[7] R. Buyya and M. Murshed, "Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing," *CCPE*, vol. 14, no. 13, pp. 1175–1220, 2002.
[8] W. H. Bell, D. G. Cameron, L. Capozza, A. P. Millar, K. Stockinger, and F. Zini, "Optorsim - a grid simulator for studying dynamic data replication strategies," *IJHPCA*, 2003.
[9] R. Buyya, R. Ranjan, and R. N. Calheiros, "Modeling and simulation of scalable cloud computing environments and the cloudsim toolkit: Challenges and opportunities," in *HPCS'09*. IEEE, 2009, pp. 1–11.
[10] D. Kliazovich, P. Bouvry, Y. Audzevich, and S. U. Khan, "Greencloud: A packet-level simulator of energy-aware cloud computing data centers," in *GLOBECOM*, 2010, pp. 1–5.
[11] S.-H. Lim, B. Sharma, G. Nam, E. K. Kim, and C. Das, "MDCSim: A multi-tier data center simulation, platform," in *CLUSTER '09*, 2009.
[12] T. Minartz, J. M. Kunkel, and T. Ludwig, "Simulation of power consumption of energy efficient cluster hardware," *Computer Science-Research and Development*, vol. 25, no. 3-4, pp. 165–175, 2010.
[13] T. Niemi, J. Kommeri, K. Happonen, J. Klem, and A.-P. Hameri, "Improving energy-efficiency of grid computing clusters," in *Advances in Grid and Pervasive Computing*. Springer, 2009, pp. 110–118.
[14] L. Ponciano and F. Brasileiro, "On the impact of energy-saving strategies in opportunistic grids," in *GRID 2010*. IEEE, 2010, pp. 282–289.
[15] G. Terzopoulos and H. D. Karatza, "Dynamic voltage scaling scheduling on power-aware clusters under power constraints," in *IEEE/ACM DS-RT*. IEEE, 2013, pp. 72–78.
[16] F. Bellosa, "The benefits of event: driven energy accounting in power-sensitive systems," in *ACM SIGOPS EW*. ACM, 2000, pp. 37–42.
[17] X. Fan, W.-D. Weber, and L. A. Barroso, "Power provisioning for a warehouse-sized computer," in *ACM SIGARCH Computer Architecture News*, vol. 35, no. 2. ACM, 2007, pp. 13–23.
[18] D. Economou, S. Rivoire, C. Kozyrakis, and P. Ranganathan, "Full-system power analysis and modeling for server environments." International Symposium on Computer Architecture-IEEE, 2006.
[19] S. Rivoire, P. Ranganathan, and C. Kozyrakis, "A comparison of high-level full-system power models." *HotPower*, vol. 8, pp. 3–3, 2008.
[20] P. Ranganathan, P. Leech, D. Irwin, and J. Chase, "Ensemble-level power management for dense blade servers," in *ACM SIGARCH Computer Architecture News*, vol. 34, no. 2, 2006, pp. 66–77.
[21] G. Dhiman, K. Mihic, and T. Rosing, "A system for online power prediction in virtualized environments using gaussian mixture models," in *DAC 2010*. IEEE, 2010, pp. 807–812.
[22] SPEC, "SPECpower_ssj2008," http://www.spec.org/power_ssj2008/.
[23] Hewlett-Packard Corporation, Intel Corporation, Microsoft Corporation, Phoenix Technologies Ltd and Toshiba Corporation, "ACPI Specification," http://www.acpi.info/DOWNLOADS/ACPIspec40a.pdf.
[24] C. Belady, A. Rawson, J. Pfleuger, and T. Cader, "Green grid data center power efficiency metrics: PUE and DCiE," Green Grid, Tech. Rep., 2008.
[25] A. S. McGough, M. Forshaw, C. Gerrard, P. Robinson, and S. Wheater, "Analysis of power-saving techniques over a large multi-use cluster with variable workload," *CCPE*, vol. 25, no. 18, pp. 2501–2522, 2013.
[26] A. S. McGough and M. Forshaw, "Reduction of wasted energy in a volunteer computing system through reinforcement learning," *Submitted to SUSCOM*, 2014.
[27] A. McGough, M. Forshaw, C. Gerrard, and S. Wheater, "Reducing the number of miscreant tasks executions in a multi-use cluster," in *CGC*, 2012, pp. 296–303.
[28] M. Forshaw, A. S. McGough, and N. Thomas, "On energy-efficient checkpointing in high-throughput cycle-stealing distributed systems," in *SMARTGREENS*, 2014.
[29] Department of Energy and Climate Change, UK Gov, "CRC Energy Efficiency Scheme Order: Table of Conversion Factors 2013/14," 2014.
[30] A. S. McGough, M. Forshaw, C. Gerrard, S. Wheater, B. Allen, and P. Robinson, "Comparison of a cost-effective virtual cloud cluster with an existing campus cluster," *Future Generation Computer Systems*, 2014.
[31] M. Solomon, "The ClassAd Language Reference Manual," *Computer Sciences Department, University of Wisconsin, Madison, WI, Oct*, 2003.
[32] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*, 1998.