Energy-aware simulation of workflow execution in High Throughput Computing systems

A. Stephen McGough
School of Engineering and Computing Sciences
Durham University
Durham, DH1 3LE, UK
Email: stephen.mcgough@durham.ac.uk

Matthew Forshaw
School of Computing Science
Newcastle University
Newcastle, NE1 7RU, UK
Email: matthew.forshaw@newcastle.ac.uk

Abstract—**Workflows offer a great potential for enacting co-related jobs in an automated manner. This is especially desirable when workflows are large or there is a desire to run a workflow multiple times. Much research has been conducted in reducing the makespan of running workflows and maximising the utilisation of the resources they run on, with some existing research investigates how to reduce the energy consumption of workflows on dedicated resources. We extend the HTC-Sim simulation framework to support workflows allowing us to evaluate different scheduling strategies on the overheads and energy consumption of workflows run on non-dedicated systems. We evaluate a number of scheduling strategies from the literature in an environment where (workflow) jobs can be evicted by higher priority users.**

I. Introduction

In recent years our ability to solve ever larger computational challenges has increased our expectations for what can be achieved using computers. We have now reached a stage where a single computer is no longer capable of solving such challenging problems. This has fuelled the need for large-scale computing systems – often achieved through the use of dedicated computer facilities containing thousands to millions of processors. These dedicated computer facilities are normally defined by the use of high speed interconnections between processors, to overcome the major problems of data exchange and loss of performance through synchronisation. However, a class of problems can be solved in parallel without the need for synchronisation or data exchange – the so-called *pleasingly parallel* problem. In these cases the work to be performed can be decomposed into completely independent jobs which can be run in any arbitrary order.

Systems which allow the enactment of these pleasingly parallel solutions are often referred to as High Throughput Computing (HTC) systems, examples include HTCondor [1] or BOINC [2]. These systems often exploit the lack of inter-communication between jobs by running jobs on non-dedicated resources where other users may have higher priority – HTC jobs being evicted from these resources and re-run at a later date; suspended and resumed later; or migrated to different hardware before continuing. Work may not progress as rapidly as on dedicated resources, however, resources purchased for another purpose can be exploited for little, if no, cost. As jobs may be evicted and re-run elsewhere this leads to an increase in their makespan (time between job submission and final results becoming available) and energy consumption – which can be detrimental to both the job submitter and infrastructure owner. In previous work we have evaluated these impacts and shown how we can reduce the energy consumption without adversely affecting individual job makespan [3].

By relaxing the requirement that each job must be completely independent of all others we can solve many more problems. If we restrict interactions to only being pre-conditions on the jobs – e.g. job C can only start once jobs A and B have completed – then jobs can still be run independently, though there is a temporal relationship now between jobs. This approach is often referred to as a workflow. This approach allows existing HTC systems to be utilised for workflow execution, as jobs can be submitted to the system as soon as all their pre-conditions have been satisfied. If there are enough (independent) jobs within the workflow or enough workflows executing concurrently then the HTC system will be highly utilised. Workflow interactions are often represented as a Directed Acyclic Graph (DAG) *cf.* Section III.

HTC systems have received considerable attention in recent years to determine optimal configuration – such as minimising the makespan or maximising utilisation of resources. Likewise, the area of workflow enactment has received much attention – maximising throughput or minimising overall makespan. This has included the use of simulation approaches allowing alternative configurations to be compared quickly and without the need to deploy them across a large infrastructure.

Computer systems have come under great scrutiny in recent years for their energy consumption, with the U.S. Environmental Protection Agency (EPA) attributing 1.5% of U.S. electricity consumption to data centre computing [4] and the ICT industry estimated as being responsible for 2% of global CO_2 production in 2007 [5]. There are calls for reducing computer energy consumption to bring it in line with the amount of work being performed – so-called *energy proportional computing* [6]. Hence, we need a greater understanding of how energy is consumed within computer systems, along with the ability to perform *'what if?'* analysis, to determine how possible configuration changes could affect the system more rapidly than deploying to a real environment. Major inroads have been made in the areas of individual system energy consumption [7] and more recently in the energy consumption of collections of computer systems – this has included Cloud simulations [8], simulations of energy consumption within non-dedicated volunteer computing systems for HTC [3] such as the scenario outlined in Section VI-B, and simulations of energy consumption of workflows running on dedicated hardware [9]. However, to the best of our knowledge, no-one has attempted to model the energy impact of workflows running over a non-dedicated volunteer computing cluster.

In this work we extend our earlier HTC simulation system (HTC-Sim [3]) to allow for the execution of workflows on a non-dedicated hardware infrastructure. This simulation allows us to rapidly evaluate the impact on high throughput users in terms of the impact of running their jobs and workflows, but also on the energy impact that this has on the overall system.

The overall architecture for HTC-Sim, extended for workflows, is shown in Figure 1, where two types of user can interact with the system – HTC users and interactive users. These are handled through 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. HTC users can submit either individual jobs – submitted direct to the HTC system – or workflows submitted through a workflow management service which then submits workflow jobs to the High Throughput Manager once their pre-conditions are satisfied. For jobs, only the submission time and the duration are considered – the execution start time and resource used may change due to the active policy set. Workflow jobs have only a duration and the submission time of the workflow. Resources within the system are grouped into *clusters*, where each cluster represents 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 rules. The HTC system has its own policy set, as does the Workflow Management service.

In Section II we present related work. In Section III we introduce our system model, while Section IV discusses the metrics we use to evaluate the policies we describe in Section V. We discuss our implementation in Section VI before evaluating of the policies and strategies in Section VII. Conclusions are presented in Section VIII.

II. Related Work

A. *Workflow schedulers*

Workflow schedulers enact jobs as defined within a workflow specification language thus allowing arbitrary workflows. Numerous workflow execution systems have been developed each focusing on different aspects [10]. e-Science Central [11] merges the concepts of social networking, Cloud computing and workflow enactment. Each workflow is enacted on a single resource with the focus on high-throughput of workflow enactment on dedicated resources. The Imperial College e-Science Networked Infrastructure (ICENI) [12] focuses on the scheduling of different implementations of the jobs within a workflow in order to meet QoS constraints. Scheduling

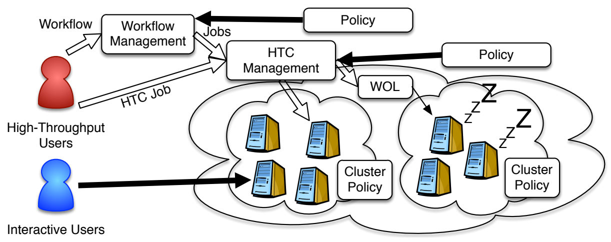

Fig. 1. Overall architecture of HTC-Sim with workflows

approaches include Simulated Annealing, game theory and MILP [13]. The above systems are all based around exclusive access to resources. However, their scheduling approaches could be used within our simulator.

The following two workflow enactment systems are capable of deploying jobs to non-dedicated HTC infrastructures (HTCondor [1]) and as such provide much of the motivation for our work here. The HTCondor DAGMan (Directed Acyclic Graph Manager) [1] is a meta-scheduler which sits above HTCondor allowing users to specify workflows as simple dependencies between jobs. Alternatively, the Pegasus [14] system can be deployed over HTCondor and allows for workflows to be defined at a higher level.

B. *Energy considerations in workflows*

The performance of workflow scheduling has been investigated extensively in the literature. Yu *et al* [15] provide a survey of workflow scheduling approaches, categorising existing approaches as either best-effort based scheduling approaches suited to multi-use grids such as our own, and QoS-constraint based scheduling which is better suited to dedicated resources. However, to date, the energy impact of workflow scheduling approaches have been subject to less scrutiny.

Durillo *et al* [16] present workflow scheduling as a bi-objective optimisation problem, factoring energy into HEFT [17] in order to explore the trade-off between makespan and energy consumption, demonstrating that small sacrifices in makespan may lead to significant energy savings.

Zhu *et al* [18] propose *pSciMapper*, a framework for the energy-aware consolidation of workflow jobs to virtualised compute resources, demonstrating total power consumption reductions of $\sim 56\%$ with a 15% slowdown for workflows.

Pietri *et al* [19] consider the scheduling of scientific workflows onto homogeneous compute resources subject to energy constraints in the presence of budget or deadline constraints.

Guérout *et al* [20] investigate the use of Dynamic Voltage and Frequency Scaling (DVFS) to promote energy efficient enactment of workflows, by slowing down the execution of jobs which do not form part of the critical path of a workflow.

Goiri *et al* [21] present a greedy algorithm for the scheduling of jobs and workflows to maximise use of solar energy. Job scheduling to promote the use of renewable energy sources has been explored outside of the context of workflows in the literature [22] and we see these approaches as complementary.

C. *Simulations with support for workflows*

A number of general-purpose Grid and Cluster level simulators exist including CloudSim, SimGrid, GridSim and OptorSim, but with the exception of SimGrid, these lack the required modeling support for DAG-based workflow scheduling.

Chen and Deelman [9] extend CloudSim [8] to support the modelling of Workflows, and evaluate the performance of FCFS, MCT, MinMin and MaxMin heuristics, with particular emphasis on overheads incurred through scheduling decisions.

We previously introduced HTC-Sim [3], a Java-based trace-driven simulation environment, demonstrating novelty in its ability to model the scheduling decisions of a HTC system in multi-use cluster environments, and the inclusion of fault-tolerance mechanisms [23]. In this work we extend HTC-Sim to support workflow execution and scheduling with an emphasis on energy consumption.

D. *Duration prediction*

Workflow scheduling strategies employed in the literature often make use of estimated completion times (durations) of workflow jobs to inform resource allocation and scheduling. User provided estimates have, however, been widely criticised by the scheduling community for their inaccuracy [24], [25]. Niu *et al* [26] analyse the traces of four large-scale systems from the Parallel Workloads Archive [27] finding only 17% of jobs completed within 90-110% of their estimate.

Hiden *et al* [28] demonstrate the ability to develop predictive models of workflow execution time based on prior executions of workflow jobs, with dynamic models that update with subsequently collected performance information.

Miu *et al* [29] present a machine learning approach to characterising the execution time of a workflow jobs based on features of input data. While encouraging levels of prediction were shown to be possible, the authors acknowledge this as a knowledge-intensive process, requiring significant numbers of exploratory prior runs to build models – good for frequently run workflows with the same data, but not for new ones.

III. System Model

We provide a brief overview of the HTC-Sim system and our extensions allowing the simulation of workflows. A more complete description of HTC-Sim can be found in [3]. The HTC-Sim model comprises of four main entities: Computers, Interactive Users, HTC Jobs and the HTC management system. We have now added Workflow as a fifth element.

A. *Computers*

Computers are realised as entities within the model, they are associated with a *cluster* within the organisation. Clusters are a mechanism to group together computers which are co-located and share the same physical parameters – often provisioned at the same time. Computers can be in a number of states, each with associated energy consumptions: sleep, idle or active. Active indicates use by an interactive user or executing a HTC job – linked either with the Interactive User entity or Job entity as appropriate. Computers within a cluster will share a set of policies governing how they are used – such as idle time before going to sleep or when to perform upgrades. All energy consumption statistics for jobs are collected globally.

B. *Interactive Users*

Interactive users, the primary users of the system, are represented by a tuple $\langle s, c, u, e\rangle$, where s and e are the login and logout timestamps respectively, c is the name of the computer, and u is a hash of the interactive users identity.

Interactive User entities will link themselves with the appropriate computer and in general will evict any job entity linked with that computer – though the simulation allows policies where Jobs are not evicted due to interactive users.

C. *HTC job*

HTC Job entities are represented by the tuple $\langle j, b, q, d, h, e, i, o\rangle$, 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 job submission time, d is the job duration, h is the hash of the submitting user's id, e is the HTC result state of running the job (either 'success' or 'terminated') and i, o represent the data transfer to and from the resource which ran the job. Note that if a job was terminated then $q+d$ represents the time that the job termination was submitted. The job will be managed by the HTC management entity and may be associated with computers at various stages.

D. *HTC management*

One HTC management entity will take in each HTC job at the time of submission and attempt to execute it on computers within the system. Policies govern when a job can be executed on a computer (such as only when the computer has no interactive user) and when to evict a job (when an interactive user logs in). The HTC system will repeatedly attempt to execute jobs on computers until either the job obtains enough time on a computer to complete, the job is terminated, or the job violates some policy associated with the HTC management system (such as being re-submitted to computers too many times). The HTC management captures statistics on jobs run.

E. *Workflow*

Jobs in our system may be either individual, or form part of a workflow. We model workflows as a *directed acyclic graph* G comprising a set of nodes – jobs – N_G and a set of arcs (or *directed edges*) – pre-conditions – A_G, each having the form $(u \rightarrow v)$, where $u, v \in N_G$. Arc $(u \rightarrow v) \in A_G$ implies v cannot execute until u has completed. We assume that a child requires all data produced by all of its parents.

We extend our tuple set for a Job by adding the DAG (G) to each (workflow) job. Each node in the DAG stores a list of its parent jobs and its child jobs. By storing the workflow in this manner it is easy to traverse the graph, in order to discover such things as siblings, ancestors and descendants.

Execution of a workflow is achieved through a new entity – the workflow manager – which first determines all jobs within the workflow which have no unresolved pre-conditions.

Each of these (workflow) jobs is then submitted to the HTC management system for execution. On completion of each workflow job the child nodes of the job which has completed are checked to see if all of their pre-conditions have been satisfied – if so then those jobs are submitted to the HTC management system. If the completed job has no children, then the whole workflow is checked for completion (all jobs complete), in which case the workflow statistics are collected and the workflow considered no further.

Workflows may be terminated at any stage, achieved by the termination of any job within the workflow. A workflow job which receives a termination request will send a termination request to the entire workflow. Workflow jobs which have already been submitted to the HTC management system will be terminated. After all workflow jobs have been terminated statistics about the failed workflow will be collected.

By using a workflow management service, along with the HTC management service, we are able to exploit all the features that the HTC management system provides whilst allowing workflows to execute. However, by using a HTC system which may not have dedicated access to resources, we bring in the complication that jobs within the workflow may start execution on a computer but be later evicted due to higher priority use. In order to reduce the impact of this on both the workflow submitter (makespan) and the system owner (energy) we can provide a policy set over the workflow management service, as well as the existing policies applied to the other parts of the system. Policies for a workflow may govern how to prioritise which jobs to run next from the workflow, which workflows to prioritise or when to deem a workflow as unsuitable to continue execution.

IV. Metrics

We have previously employed three prevalent performance metrics in evaluating HTC systems, namely; overhead, slowdown and bounded slowdown [3]. Although these may be applied to the individual jobs within a workflow, we define here a new set of metrics more applicable to workflows:

Critical Path Workflow Overhead: Here we compute overhead with respect to the critical path through the workflow:

$$CPWO = f_i - s_i - C_i$$

where C_i is the duration of the critical path (longest execution path through the workflow), s_i and f_i are the submission time and time the last data transfer back finishes for workflow W_i. As the critical path is the shortest time required to complete the workflow $CPWO$ can never be negative and will give a good estimate of how optimal the execution was.

Overall Workflow Slowdown / Speedup:

$$OWSS = \frac{f_i - s_i}{\sum_{j \in W_i} r_j}$$

where r_j is the execution time for job j in workflow W_i. Values of $OWSS$ greater than one indicates that the workflow ran slower as a consequence of executing through the HTC system, while values less than one indicate how the HTC system has exploited the parallelism implicit within the workflow.

Critical Path Slowdown: Here we can determine how close the execution approached optimal parallel execution:

$$CPS = \frac{f_i - s_i}{C_i}$$

where CPS approaches 1 as the execution approaches optimal parallelism. It should be noted that $CPWO$, $OWSS$ and CPS only make sense for workflows which have completed. For workflows which have failed to complete we do not concern ourselves as to how *optimally* they ran but how much resources they wasted – both time and energy.

Energy: Total energy consumed by the workflow:

$$TE = \sum_{j \in W_i} E_j$$

where E_j is the energy consumed in running job j:

$$E_j = \sum_{k \in A_j} (e_{j,k} - b_{j,k}) \cdot E_{j,k}$$

where A_j is the set of all attempts to run job j, $e_{j,k}$, $b_{j,k}$ are the finish and start times of invocation k of job j on a compute resource, $E_{j,k}$ is the energy consumption rate for the compute resource used for that attempt. Note that for terminated workflows A_i may be empty and jobs active at the termination time will have $e_{j,k}$ set to the termination time.

Energy increase: This is the extra energy incurred when running the workflow through the HTC system:

$$\sum_{j \in W_i} E_j - r_j \cdot E_{opt}$$

where E_{opt} is the energy rate of the best available computer.

Workflow evictions: The number of evicted workflow jobs:

$$WE = \sum_{j \in W_i} max(|A_j| - 1, 0).$$

Ratio of average parallelism achieved: This indicates the proportion of parallelism achieved from the real run in comparison to the maximum theoretical level of parallelism. We ignore here any evicted jobs as they would give a false level of parallelism. The ratio of average parallelism is:

$$R = \frac{average(real)}{average(optimal)} = \frac{\frac{1}{f_i - s_i} \sum_{r_i \in W_i} r_i}{\frac{1}{C_i} \sum_{r_i \in W_i} r_i} = \frac{C_i}{f_i - s_i}$$

V. Policies

We describe here a number of existing workflow scheduling and resource selection approaches before evaluating them in SectionVII in the context of a non-dedicated multi-use cluster.

A. Task scheduling strategies

We identify here the following workflow scheduling approaches common in the literature [15], used to determine which job will be next allocated to a computer.

FCFS: Under First Come First Serve (FCFS), jobs are allocated to resources in ascending order of their arrival time.

Min-Min: Jobs whose execution times are predicted to be shorter are given greater priority.

Max-Min: Jobs whose execution times are precticted to be longer are given greater priority.

Heterogeneous Earliest Finish Time (HEFT): [17] Prioritised based on future impact – dependant jobs and communications. Priority of job n_i is calculated recursively:

$$rank(n_i) = \overline{w_i} + \max_{n_j \in succ(n_i)}(\overline{c_{i,j}} + rank(n_j))$$

where $succ(n_i)$ is the set of immediate successors of n_i, $\overline{c_{i,j}}$ is the average communication cost of edge (i, j), and $\overline{w_i}$ is the average computation cost of job n_i.

B. Execution time estimation

Previous workflow scheduling approaches have assumed *a priori* knowledge of job execution times. However, user estimates of job execution time have been shown to be unreliable [24], [25], [26]. We evaluate three estimation policies:

Perfect: Perfect *a priori* knowledge of job duration.

Binned execution time: Users are capable of placing their jobs into pre-defined intervals of duration such as: *up to 1 hour, 1-2 hours, 2-6 hours*.

Order of magnitude: Users are only capable of estimating job durations to within an order of magnitude. Estimates are calculated as *a priori*$\times 10^r$ where r is uniformly distributed on $[-1, 1]$.

C. Resource allocation strategies

We evaluate here a number of resource selection stratergies for (workflow) job placement based on our prior work [3]:

S1: Default HTCondor [30] policy, random resource selection favouring computers which are powered up.

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.

VI. Implementation

We have developed HTC-Sim and our workflow extensions as a *pluggable* simulation framework written in Java. This allows the development of new strategies and functionality through new classes which implement a given API. The selection of classes to use in any given invocation is handled through the use of a configuration script. Below we discuss the practical implementation issues for our approach.

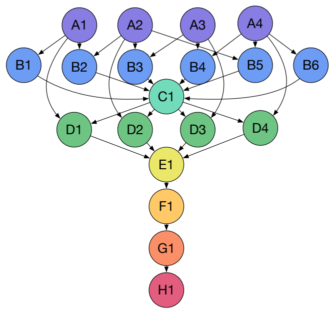

Fig. 2. Example of a Montage workflow

A. Generating Synthetic Workflows

Bharathi *et al* [31] characterise a number of workflow applications from scientific communities ranging from astronomy to bioinformatics. Statistical models are provided for the mean runtime and variance, and the size of input and output data for each task type comprising workflow applications. Silva *et al* [32] go further to present a synthetic workflow generator based on the profiling efforts in [31]. We employ this to generate synthetic DAGs, modelling five different types of real scientific application as described further in [33], namely Montage, LIGO, CyberShake, Epigenomics, and SIPHT.

Here we focus on Montage [34], an open-source astronomy workflow created by NASA/IPAC to assemble astrophotography images in Flexible Image Transport System (FITS) format. An example Montage workflow is shown in Figure 2.

We use our 2010 HTCondor trace data, randomly replacing $p\%$ of the jobs with workflows. Workflow sizes are distributed uniformly in the range $[s_l, s_u]$. We average the results of multiple simulations to ameliorate random effects.

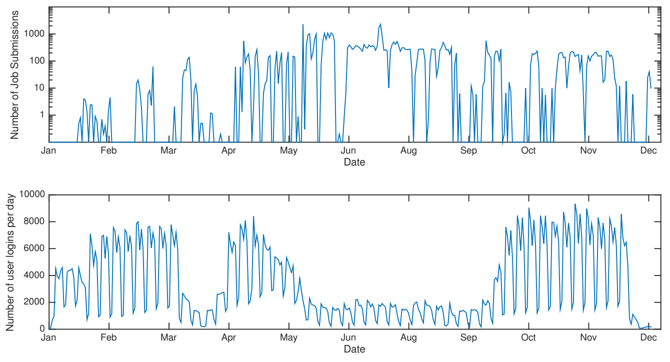

Fig. 3. Interactive user activity and HTCondor workload trace for 2010 [3]

B. HTCondor at Newcastle University

We drive our trace-driven simulation based on our HTCondor and interactive user traces from 2010. An outline of these traces is shown in Figure 3. Full details of these traces can be found in [3], [23]. The traces are based on 1359 student access computers, distributed around the Newcastle campus in 35 clusters, with some clusters dedicated for teaching and others open access, and with each cluster exhibiting a different interactive user profile. Computer energy consumption is based on a tuple$\langle a, i, s\rangle$, where a, i and s are the energy rates for active, idle and sleep. Three computer types were present at the time: Normal $\langle 57W, 40W, 2W\rangle$, High-end $\langle 114W, 67W, 3W\rangle$ and Legacy $\langle 100-180W, 50-80W, 4W\rangle$. As our focus is the comparison of different polices for reducing energy, we ignore performance differences between computers and assume the execution time will match the original execution time.

HTC-Sim models the bandwidth available between nodes, imposing time delays on data ingress/egress. Estimated data transfer delays may then be used to inform resource allocation and other decisions. Further details can be found in [35].

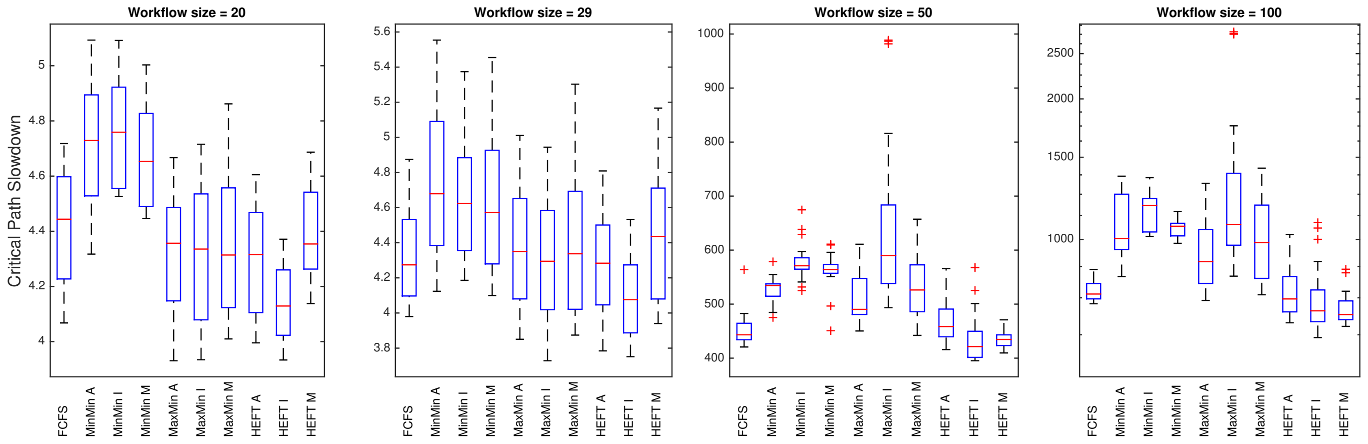

Fig. 4. Critical path slowdown due to job selection policy

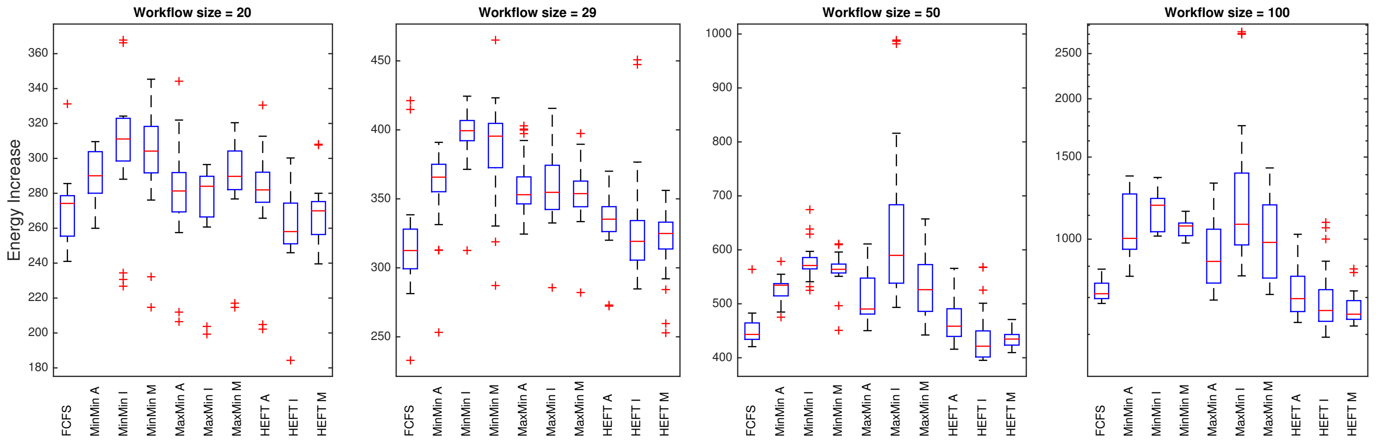

Fig. 5. Energy increase due to job selection policy

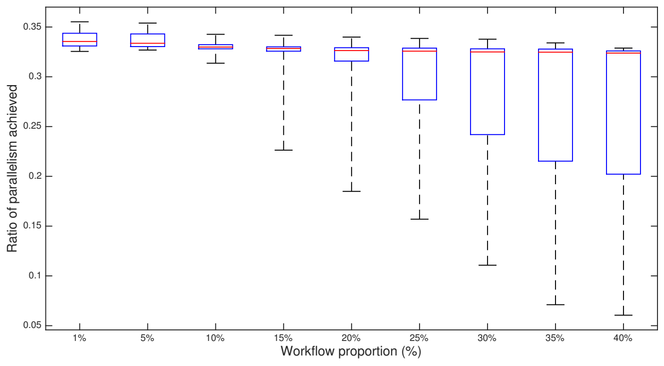

Fig. 6. Impact of workflow proportion on workflow parallelism achieved

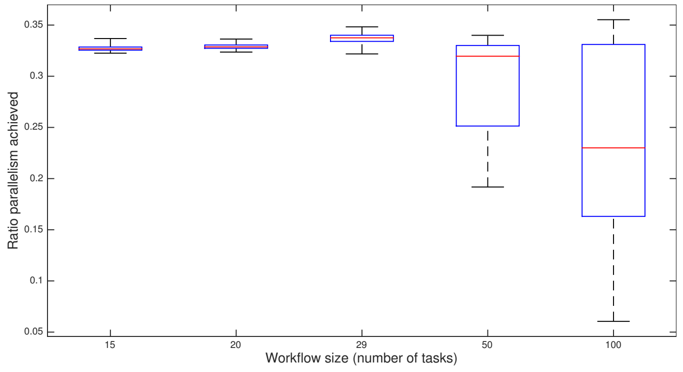

Fig. 7. Impact of Workflow size on workflow parallelism achieved

VII. Experimental Results

Here we present our experimental results. To the best of our knowledge this is the first attempt to evaluate the energy consumption of workflow scheduling approaches over a non-dedicated volunteer computing cluster.

For this paper we fix parameters s_l and s_u governing the number of jobs comprising each generated synthetic workflow such that they are equal, $s_l = s_u \in \{15, 20, 29, 50, 100\}$, allowing us to more readily isolate the impact of workflow size on performance and energy consumption.

Figure 4 (A = *a priori*, I = Interval, M = Order of magnitude): The best policy combination is $\langle$HEFT,I$\rangle$ for all workflow sizes apart from 100 where $\langle$HEFT,M$\rangle$ is slightly

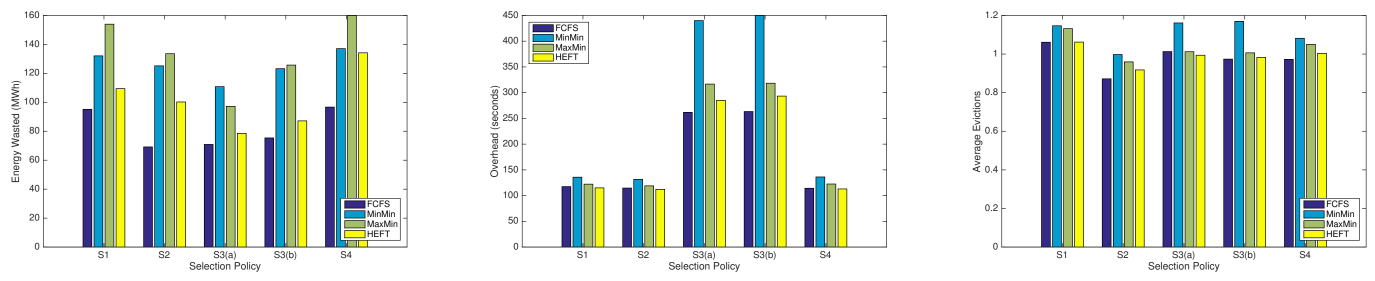

Fig. 10. Impact of resource allocation strategies on Energy Wasted, Overhead and Average Evictions.

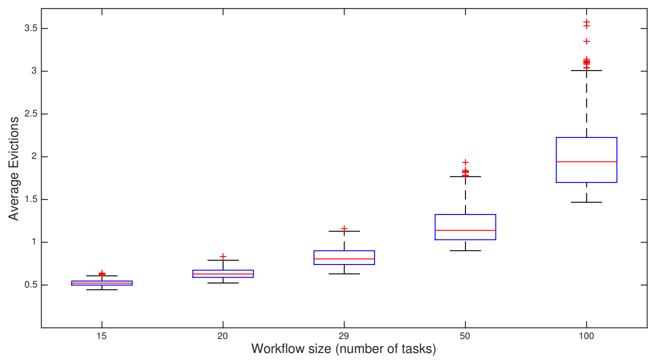

Fig. 8. Impact of Workflow size on evictions

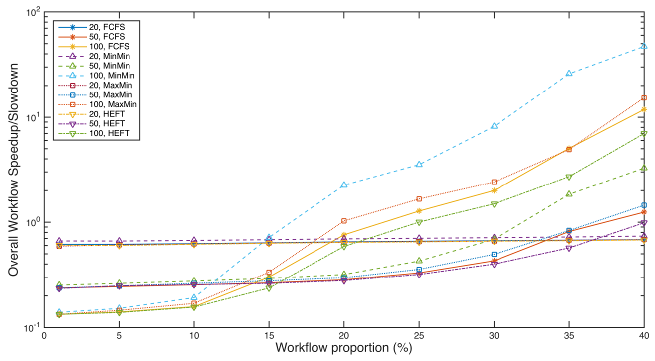

Fig. 9. Workflow speedup / slowdown for different workflow sizes

better. MinMin is a bad choice in all cases, being beaten by all other policies, though if chosen it should be used with order of magnitude. The slowdown for FCFS is diminishing as the workflow size increases – potentially better for larger workflows. For small workflow sizes MaxMin does not seem to be affected by job execution time estimation. Figure 5: In this case the best policy combination seems to be either ⟨HEFT,I⟩ or ⟨FCFS,I⟩ for the case of workflow size 29. As the workflow size increases, ⟨HEFT,M⟩ increases to slightly outperform ⟨HEFT,I⟩ for workflow size 100. FCFS is a close second in most cases. Although Interval is good for HEFT it is not good for the other combinations. When we consider critical path slowdown and energy increase in combination (Figures 4 & 5), ⟨HEFT,I⟩ performs best.

Figure 6 shows the impact of increasing the proportion of synthetic workloads ($p\%$ of our 2010 dataset) on the proportion of parallelism achieved. We see little effect for workflow mixes between $p = 1\%$ and $p = 10\%$ - giving a parallel ratio of ~ 0.34. However, as the value of p increases, contention over resources in the HTC pool have a detrimental effect on the proportion of parallelism achieved. Similarly, in Figure 7 we observe the proportion of workflow parallelism achieved, as the number of jobs in each workflow increases. Parallelism rises only marginally between 15 and 29 – most likely limited by the limited parallelism from the Montage workflows at this size. However, as the workflow size increases the median parallelism drops – a consequence of the number of jobs now being executed leading to an increase in the number of job evictions. This can be evidenced from Figure 8 which shows that the number of evictions does increase as the workflow size increases. Though the variance indicates that if $p\%$ is low then the workflow parallelism is maintained.

Figure 9 illustrates the speedup ($y < 1$) and slowdown ($y > 1$) of workflows run within the system. For workflows of size 20 (or less) speedup is achieved in all offered workflow loads - this remains constant across $p\%$ – a consequence of the system not becoming overloaded for these small workflows. However, for larger workflows, a small value of $p\%$ gives parallel speedup – due to the inherent parallelism in the workflow – though as $p\%$ increases this speedup is lost and becomes a slowdown – a consequence of the large number of jobs within the system leading to delays in job execution and a higher probability of job eviction.

In Figure 10 we evaluate the different resource selection policies. In terms of energy consumption, Policy S2 (lowest power computers first) coupled with FCFS is the best option. For the other job selection policies policy S3(a) gives a lower energy impact; however, this results in significantly increased overheads. This is due to short jobs being placed on computers most likely to be idle, meaning that longer jobs get placed on computers which are more likely to see evictions due to interactive users. This leads to an increase in average evictions. Policy S4 (use clusters while closed) gives good overheads (as jobs tend not to be evicted) though energy consumption is not as good. This is a consequence of the university placing their ‘best’ (low energy) computers in popular open-access clusters, meaning closed clusters tend to be legacy.

VIII. Conclusion

We have presented details of an extension to the HTC-Sim simulation framework to support the modelling of workflow enactment within a High Throughput Computing system comprising multi-use cluster resources.

We evaluate the effectiveness of workflow scheduling strategies (job selection and execution time estimation) normally applied to dedicated systems in terms of energy consumption and performance. We evaluate these workflow scheduling heuristics using a number of metrics to determine how appropriate they are to a multi-use cluster. We see our system as having great potential for evaluating different (novel) strategies for workflow enactment on non-dedicated resources.

Results suggest that using HEFT, along with asking users to select a time interval for their workflow job execution times, gives the lowest energy consumption and best experience for workflow users, better than the results for *a priori* information.

We seek to extend our workflow model to take into account more general workflow enactment systems, along with modelling other workflow scheduling approaches. For example, advanced reservations could exploit clusters which are closed, hence aid ability to perform back-filling. Checkpointing and migration, which has already been built into our simulation [36], [37], has been shown to be effective in reducing overheads and energy consumption for individual jobs and would seem applicable to workflows. We are currently investigating novel energy-aware workflow scheduling policies for multi-use clusters of dedicated and non-dedicated resources.

References

[1] T. Tannenbaum, D. Wright, K. Miller, and M. Livny, "Condor: a distributed job scheduler," in *Beowulf cluster computing with Linux*. MIT press, 2001, pp. 307–350.

[2] D. P. Anderson, "Boinc: A system for public-resource computing and storage," in *Grid Computing, 2004*. IEEE, 2004, pp. 4–10.

[3] M. Forshaw, N. Thomas, and A. S. McGough, "Trace-driven simulation for energy consumption in high throughput computing systems," in *18th IEEE/ACM International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*, 2014.

[4] R. Brown, "Report to congress on server and data center energy efficiency: Public law 109-431," *Lawrence Berkeley National Laboratory*, 2008.

[5] C. Pettey. (2007) Gartner estimates ICT industry accounts for 2 percent of global CO2 emissions. [Online]. Available: http://www.gartner.com/newsroom/id/503867

[6] L. Barroso and U. Holzle, "The case for energy-proportional computing," *Computer*, vol. 40, no. 12, pp. 33–37, Dec 2007.

[7] D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: a framework for architectural-level power analysis and optimizations," in *ISCA*, June 2000, pp. 83–94.

[8] 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.

[9] W. Chen and E. Deelman, "Workflowsim: A toolkit for simulating scientific workflows in distributed environments," in *e-Science 2012 IEEE 8th International Conference on*. IEEE, 2012, pp. 1–8.

[10] J. Yu and R. Buyya, "A taxonomy of workflow management systems for grid computing," *Journal of Grid Computing*, vol. 3, no. 3-4, pp. 171–200, 2005.

[11] H. Hiden, S. Woodman, P. Watson, and J. Cala, "Developing cloud applications using the e-Science central platform," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 371, no. 1983, Jan. 2013.

[12] A. S. McGough, L. Young, A. Afzal, S. Newhouse, and J. Darlington, "Workflow enactment in ICENI," in *UK e-Science All Hands Meeting*, 2004, pp. 894–900.

[13] A. Afzal, A. S. McGough, and J. Darlington, "Capacity planning and scheduling in grid computing environments," *Future Generation Computer Systems*, vol. 24, no. 5, pp. 404 – 414, 2008.

[14] E. Deelman, K. Vahi, G. Juve, M. Rynge, S. Callaghan, P. J. Maechling, R. Mayani, W. Chen, R. Ferreira da Silva, M. Livny, and K. Wenger, "Pegasus, a workflow management system for science automation," *FGCS*, vol. 46, pp. 17–35, 2015.

[15] J. Yu, R. Buyya, and K. Ramamohanarao, "Workflow scheduling algorithms for grid computing," in *Metaheuristics for scheduling in distributed computing environments*. Springer, 2008, pp. 173–214.

[16] J. J. Durillo, V. Nae, and R. Prodan, "Multi-objective workflow scheduling: An analysis of the energy efficiency and makespan tradeoff," in *IEEE/ACM CCGRID*, 2013, pp. 203–210.

[17] H. Topcuoglu, S. Hariri, and M.-y. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE TPDS*, vol. 13, no. 3, pp. 260–274, 2002.

[18] Q. Zhu, J. Zhu, and G. Agrawal, "Power-aware consolidation of scientific workflows in virtualized environments," in *IEEE SC'10*, 2010, pp. 1–12.

[19] I. Pietri, M. Malawski, G. Juve, E. Deelman, J. Nabrzyski, and R. Sakellariou, "Energy-constrained provisioning for scientific workflow ensembles," in *IEEE CGC*, 2013, pp. 34–41.

[20] T. Guérout, T. Monteil, G. Da Costa, R. N. Calheiros, R. Buyya, and M. Alexandru, "Energy-aware simulation with DVFS," *Simulation Modelling Practice and Theory*, vol. 39, pp. 76–91, 2013.

[21] Í. Goiri, K. Le, M. E. Haque, R. Beauchea, T. D. Nguyen, J. Guitart, J. Torres, and R. Bianchini, "Greenslot: scheduling energy consumption in green datacenters," in *ACM SC'11*, 2011, p. 20.

[22] C. Stewart and K. Shen, "Some joules are more precious than others: Managing renewable energy in the datacenter," in *HotPower'09*, 2009.

[23] M. Forshaw, "Operating policies for energy efficient large scale computing," Ph.D. dissertation, Newcastle University, UK, 2015.

[24] C. Bailey Lee, Y. Schwartzman, J. Hardy, and A. Snavely, "Are user runtime estimates inherently inaccurate?" ser. LNCS, vol. 3277, 2005, pp. 253–263.

[25] S. Srinivasan, R. Kettimuthu, V. Subramani, and P. Sadayappan, "Characterization of backfilling strategies for parallel job scheduling," in *ICPPW'02*, 2002.

[26] S. Niu, J. Zhai, X. Ma, M. Liu, Y. Zhai, W. Chen, and W. Zheng, "Employing checkpoint to improve job scheduling in large-scale systems," in *JSSPP*. Springer, 2013, pp. 36–55.

[27] http://www.cs.huji.ac.il/labs/parallel/workload/.

[28] H. Hiden, S. Woodman, and P. Watson, "A framework for dynamically generating predictive models of workflow execution," in *ACM WORKS'13*, 2013, pp. 77–87.

[29] T. Miu and P. Missier, "Predicting the execution time of workflow activities based on their input features," in *IEEE SCC'12*, 2012.

[30] M. Litzkow, M. Livney, and M. W. Mutka, "Condor-a hunter of idle workstations," in *ICDCS '88*, 1998, pp. 104–111.

[31] S. Bharathi, A. Chervenak, E. Deelman, G. Mehta, M.-H. Su, and K. Vahi, "Characterization of scientific workflows," in *ACM WORKS'08*, 2008, pp. 1–10.

[32] R. F. d. Silva, W. Chen, G. Juve, K. Vahi, and E. Deelman, "Community resources for enabling research in distributed scientific workflows," in *e-Science (e-Science), 2014 IEEE 10th International Conference on*, 2014.

[33] G. Juve, A. Chervenak, E. Deelman, S. Bharathi, G. Mehta, and K. Vahi, "Characterizing and profiling scientific workflows," *FGCS*, vol. 29, no. 3, pp. 682–692, 2013.

[34] J. C. Jacob, D. S. Katz, G. B. Berriman, J. C. Good, A. Laity, E. Deelman, C. Kesselman, G. Singh, M.-H. Su, T. Prince *et al.*, "Montage: a grid portal and software toolkit for science-grade astronomical image mosaicking," *IJCSE*, vol. 4, no. 2, pp. 73–87, 2009.

[35] 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," *FGCS*, vol. 41, no. 0, pp. 65 – 78, 2014.

[36] M. Forshaw, A. S. McGough, and N. Thomas, "On energy-efficient checkpointing in high-throughput cycle-stealing distributed systems," in *SMARTGREENS*, 2014.

[37] ——, "Energy-efficient checkpointing in high-throughput cycle-stealing distributed systems," *ENTCS*, vol. 310, pp. 65–90, 2015.