# Simulation of Virtual Machine Live Migration in High Throughput Computing Environments

Osama Alrajeh
*School of Computing*
*Newcastle University*
Newcastle upon Tyne, UK
o.alrajeh1@ncl.ac.uk

Matthew Forshaw
*School of Computing*
*Newcastle University*
Newcastle upon Tyne, UK
matthew.forshaw@ncl.ac.uk

Stephen McGough
*School of Computing*
*Newcastle University*
Newcastle upon Tyne, UK
stephen.mcgough@ncl.ac.uk

Nigel Thomas
*School of Computing*
*Newcastle University*
Newcastle upon Tyne, UK
nigel.thomas@ncl.ac.uk

***Abstract*—Virtual Machine (VM) live migration is one of the strategic approaches that can be employed to reduce energy consumption and increase the utilisation of a single computer in large computing infrastructure. However, virtualisation in HTC has received limited attention in the literature. In this paper, we present an extension of an existing trace-driven simulation to incorporate virtualisation. Furthermore, we implement the pre-copy live migration algorithm to provide a test environment for job live migration in HTC system. Our simulation provides the total number of migrations and their overall time of migrations as well as calculates the energy consumption of migrations during its runtime. We validate our tool by presenting some outcomes of simulation. We demonstrate that energy consumption of jobs migrations may consume up to 10% of system total energy.**

## I. Introduction

Computer simulation is a powerful way of evaluating complex systems and understand the reaction of the systems in the real world. As VM live migration becomes a mechanism for reducing overall energy consumption in large-scale computing by consolidating VMs onto fewer physical machines, simulation assists to enhance the policies of VM live migration decisions to gain more saving in energy and better performance.

High Throughput Computing (HTC) is powerful for a large number of jobs with a long period of execution time, where jobs can be executed over a distributed set of computers. HTC systems such as HTCondor [1], BOINC [2], and SGE [3] are popular choices for academic as well as industry researchers, to do complex computational tasks on existing, idle, shared facility (desktop grid) or dedicated resources. Since the resources in HTC environment is shared usage, the HTC jobs might be affected by other users. For instance, universities use students' computer clusters as a resource pool for their HTC system where HTC jobs can be executed on computers that are not being used by students. There is a high probability that HTC job can be interrupted when a user starts using the computer. Such an interruption cause job eviction or suspension which increase the power consumption and job makespan.

It is significant to run these long-running jobs in the most energy-efficient machines to decrease the energy consumption. However, the most energy-efficient machines might be not available at the beginning of the HTC jobs initiation. Furthermore, it is necessary to prevent the HTC jobs from being evicted from the computer or suspended for a long time to minimise energy waste and makespan of the jobs.

Virtual Machine (VM) Live Migration [4] refers to the method of transferring running VMs between physical machines without impacting client processes or applications. It has become a solution for managing tasks in large-scale computing environments to accomplish the purposes of energy management, load balancing, fault tolerance, and zero-downtime hardware maintenance. VM consolidation is a common technique used to reduce the energy consumption and improve the utilisation in Cloud datacenters. Many VM consolidation algorithms have been introduced in the literature to control the energy waste in Cloud environment. Therefore, we find it necessary to adopt some of these algorithms by applying them in HTC environments as well as proposing new VM consolidation algorithms that work the most in HTC system. By deploying the virtualisation into HTC-Sim, researchers have the opportunity to develop and evaluate their algorithms based on real data.

Virtualisation has previously been applied to HTC systems such as HTCondor where VMs considered as jobs to be executed [5]. The power of VM live migration could be used to gain more saving in energy by migrating the jobs into energy efficient machines when they become available. Moreover, it can prevent the jobs from being evicted or suspended for a long time due to user login by migrating them to other idle machines. However, VM live migration in HTC system has received a little attention in the literature. Existing research has considered the use of the Checkpointing [6]–[9] in this context to minimise the effect of jobs interruption on overall performance and energy consumption of HTC systems.

Checkpointing is a well-known approach to achieve fault tolerance by regularly storing snapshots of application state into storage. In the event of failure, these snapshots can be used to avoid repeating the execution of the task from the beginning. However, some checkpointing algorithms could lead to useless checkpoints which increase the overhead as well as the energy consumption. Also, checkpoint might have large latency which affects the overall performance of the system. Implementing the migration in HTC systems to avoid task's failure by transferring the task to a safe node when the current node of the task seems likely to fail, will reduce the

overhead as well as the energy consumption of the system.

In this paper, we propose a different approach in which we use the ability of VM live migration to mitigate the energy waste and the makespan of jobs in HTC environment. In order to achieve this, we extend a trace-driven simulation HTC-Sim [10] to support virtualisation, in particular, VM live migration. We implemented the pre-copy migration algorithm [4] in the same manner as in the real world. The simulation tracks the number of successful and failed VM migrations, the time of VM migrations, and the energy consumption of VM migrations. Additionally, the expansion of this simulation will ultimately assist the development and evaluation of algorithms for VM consolidation in large-scale computing.

The rest of this paper is organised as follows. Section II discuss the related work. We introduce the simulation environment in Section III. In Section V, we present two scenarios to perform the live migration in our simulation. We explain the outcomes of the simulation in Section VI before concluding in Section VII .

## II. Related Work

### A. Simulation Tools

The simulation of Grid and Cluster level has formed the basis for many previous works such as SimGrid [11], GridSim [12], OptorSim [13], and Chi-Sim [14]. These tools assist researchers in understanding the parallel and distributed systems and evaluate new policies of managing tasks in HTC. In addition, Cloud simulators such as CloudSim [15], GreenCloud [16], iCanCloud [17], and MDCSim [18] can determine the tradeoff between performance and cost as well as the energy. Though, unlike our extinction to HTC-Sim, the tools as mentioned above might not be ready to evaluate VM consolidation techniques and policies for the purpose of reducing energy waste in HTC environment. Also, our simulation is unique in its capability to model live migration in multi-use clusters with interactive users besides using real-world workload traces.

### B. Virtual Machine Consolidation

Clark *et al.* [4] purposed the concept of pre-copy live migration algorithm. This algorithm copies and transfers the memory pages of VM from the source host to the target host. Then, when there are relatively few uncopied pages, the VM get suspended on the source host, and the remaining pages get transferred to target host. In this way, the VM can be migrated from machine to another with minimal downtime.

As part of an attempt to lower energy usage and SLA infringements, Beloglazov *et al.* [19] formulated algorithms and policies linked to live migration and dynamic VM consolidation. The researchers drew on CloudSim instruments to facilitate the assessment of their approaches. Initially, to carry out the reallocation of the VMs, the physical machines were categorised into the following two groups: overload and under load machines. Following this, three distinct VM selection policies were employed to determine the VM which needs to be transferred from the present to the novel host. Although useful insights were gathered from their experiment, it should be noted that the researchers' method was not appropriate to ensure that the VM live migration costs were lower than the advantages gained.

In approaching possible solutions to the issue of workload consolidation, Feller *et al.* [20] employed the multi-dimensional bin-packing (MDBP) problem. In addition to this, by taking inspiration from the natural world, the researchers formulated a new workload consolidation algorithm which extended the Ant Colony Optimisation (ACO) algorithm linked to energy efficient cloud computing. The central objective of the researchers was to lower the physical machine requirement for computing current workloads. To facilitate the assessment of their method in view of the limiting factors of the CloudSim toolkit, a Java-based simulator was formulated by considering the ACO in relation to the First-Fit Decreasing algorithm (FFD). After the simulation was completed, the findings illustrated that ACO was far more conducive to energy conservation than FFD, but the key piece of information missing from their findings was migration cost.

## III. Simulation Environment

In this paper, we implement virtualisation and the pre-copy live migration algorithm into HTC-Sim simulator. HTC-Sim is a Java-based trace-driven simulation for simulating multi-use resources shared with interactive users as well as dedicated resources. Moreover, the tool is based on a real dataset collected from Newcastle University HTCondor system in 2010.

### A. Datasets

The Newcastle University employed HTCondor to provide a high throughput computing environment to its researchers. The HTCondor pool contains 1400 desktop computers spread through 35 clusters on campus. When a student login to the computer, the job on that computer get evicted and rejoin the queue of the HTCondor to be restarted on another machine. Figure 1 shows the number of user logins per day during 2010 while Figure 2 illustrates the number of HTCondor jobs in the same period.

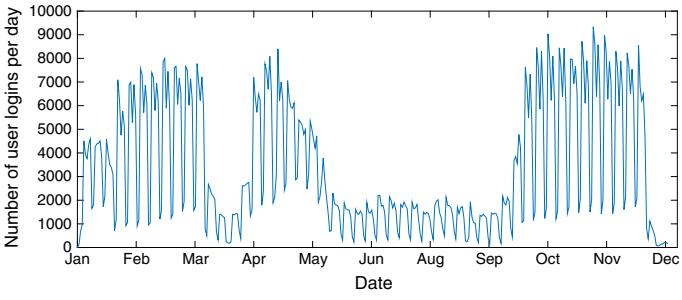


Fig. 1. Interactive user logins per day in 2010 [10].

A total of 1,229,820 user logins were occurred over 2010, and the total number of submitted HTCondor jobs was 561,851. Where we can see a high probability for a job to

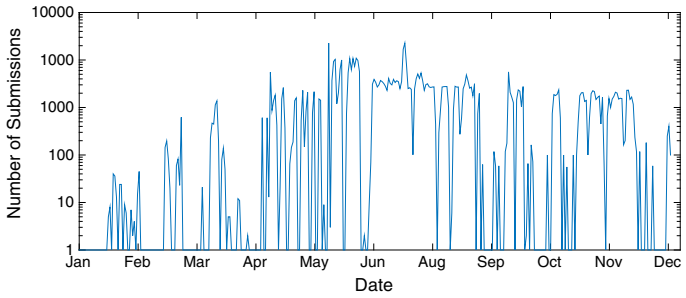


Fig. 2. HTCondor jobs submission in 2010 [10].

be interrupted by an interactive user. We can stop jobs from being disturbing by migrating them to resume their execution on other machines. To achieve this, the HTCondor jobs that run in a closed cluster can be moved to other idle computers just before its opening time for students. Also, when a user logs in while HTCondor task is running, the task can be migrated to another machine.

### B. Migration Model

The simulator requires three files in order to perform. The first file contains the policies configurations to needs to be evaluated by the simulation. The trace log of HTCondor workloads is included in the second file. The workloads records specify the submission time of the jobs, their duration, and their memory usage at the time of completion. The third file has the trace log of user login to computers and logout from computers in addition to the specifications of computers.

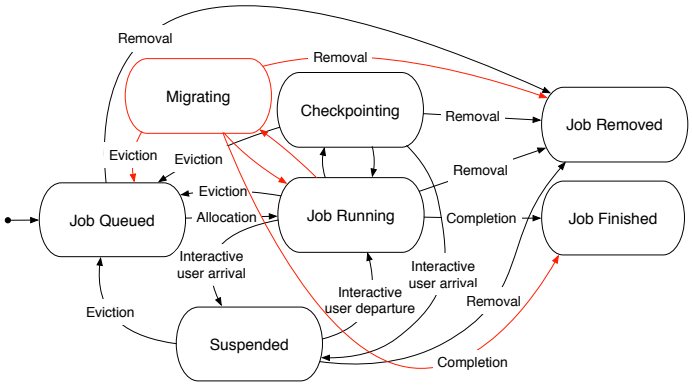


Fig. 3. Job state transition diagram with migration state.

The original state transition diagram for a single job in HTC-Sim has been presented in [10]. The HTCondor job enters the queue when it is submitted by a user and then waits to be allocated on a machine. During its runtime, the job might be interrupted by interactive users. As a result, the job gets suspended for a while or evicted immediately. The task can also regularly checkpoint during its execution time. Furthermore, a system administrator or the owner of the job may manually remove the job at any state.

In Figure 3 we add the Migration state to the initial workload state diagram. The migration of the workload can occur at any time of its execution time. Additionally, migration policies manage the migration process by determining which job, when, and where to migrate. Also, live migration allows jobs to continue executing while migration is taking place. Moreover, we assume the completion of the job might happen during the migration along with job eviction.

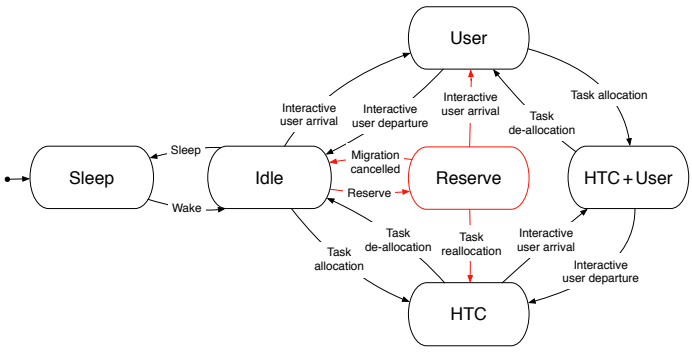


Fig. 4. Computer state transition diagram with reserve state.

Figure 4 shows the computer state transition diagram which previously introduced in [10]. We include the Reserve state in it where computer enters when it is selected as a target host for a migrated job. When the job starts transferring from the source host to the target host, the computer enters the HTC state. Furthermore, the Reserve state changes to the User state if an interactive user starts using it. Also, the reserved computer begins idle when the migration is canceled. When a machine is in a Sleep state, it powered-down except the RAM. In this way, the machine consumes very low energy and can be powered-up very quickly without restarting the operating system. Unlike the machine when it is idle or reserved, it consumes more energy than the Sleep state, but much lower than HTC, User, and HTC+User states.

## IV. Simulation Scenario

In this section, we present two different techniques to perform the live migration in Newcastle University's HTC system. The techniques are based on our observation of the system in order to mitigate jobs energy waste and makespan. To achieve this, the system needs to prevent jobs from rejoin the queue and restart executing due to jobs evictions.

### A. Migration Interval

In this technique, we introduce a fix period of time during job execution to check if the condition of migration occurs. When there is a need for job migration, the system selects a target computer from HTC system computer pool. Then, the process of migration starts on the source and target computers. However, the job on the source might be interrupted by an interactive user before or during the migration which leads to job eviction. Additionally, an interactive user also may login into the target computer during the migration process. As a result, the migration process stops, and the job resumes on the source computer.

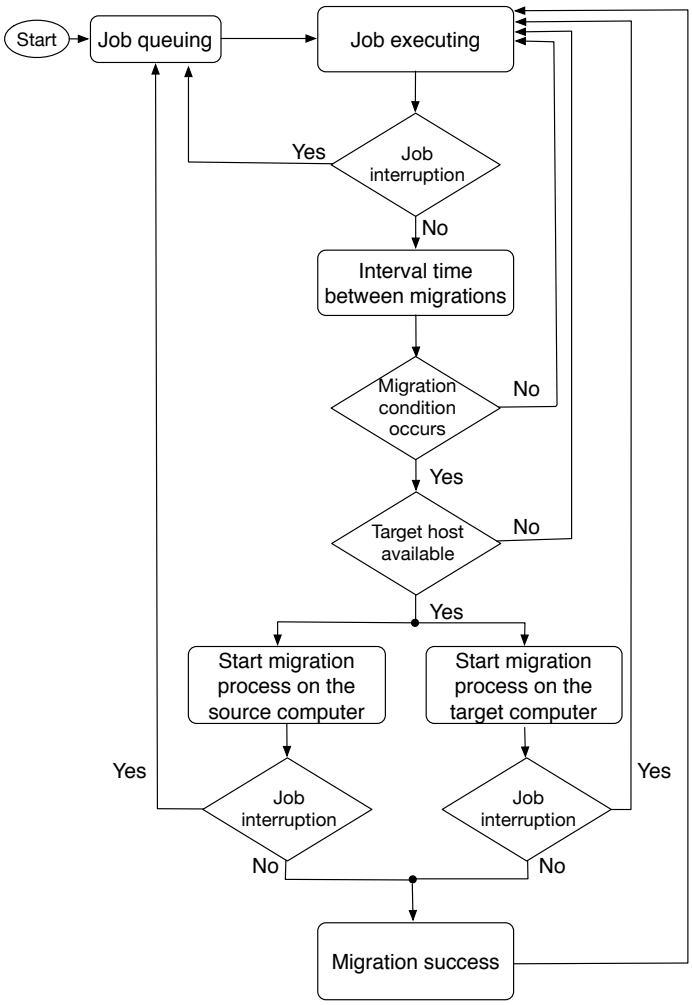


Fig. 5. The flow chart of interval live migration.

The flow chart in Figure 5 illustrates the method of interval migration. Where a set of $k$ jobs $J$=$\{J_1,J_2,J_3,...J_k\}$ presents in the queue. Each $J_i$; $i \in \{1,k\}$ has execution time $ET(J_i)$. The starts time of the job $ST(J_i)$ is given when the job is allocated on a source for execution. Then, the finish time $FT(J_i)$ of the job is calculated as

$$FT(J_i) = ST(J_i) + ET(J_i). \quad (1)$$

When there is an interruption, the job $J_i$ gets a new starting $ST(J_i)$ as well as new finishing time. Moreover, the queuing time can be represented as $QT(J_i)$ for a job $J_i$. After interval time $N$, when the migration condition occurs, the job $J_i$ starts migrating from the source computer $C_s$ to target computer $C_t$. The job migration process is expressed as $J_i : (C_s, \rightarrow C_t)$ and the time of migration is represented as $MT(C_s \rightarrow C_t)$. Since we are using live migration, the job $J_i$ is still running on a source machine $C_s$ during the migration. Then, the starting time of the job $J_i$ on target machine $C_t$ is giving as

$$ST_t(J_i) = ST_s(J_i) + ET_s(J_i) + MT(J_i : (C_s \rightarrow C_t)). \quad (2)$$

The remaining execution time of job $J_i$ on a target machine $C_t$ is calculated as

$$ET_t(J_i) = ET(J_i) - ET_s(J_i) - MT(J_i : (C_s \rightarrow C_t)). \quad (3)$$

### *B. Migration Responsive*

In this method, we migrate the job when an interactive user arrives while HTC job is running. The HTC job and the interactive user share the same machine until the job is migrated to another machine. However, this can affect the performance of the running applications by the interactive user on the source machine. The interactive user will have full CPU and memory capacities of the source machine when the job is migrated.

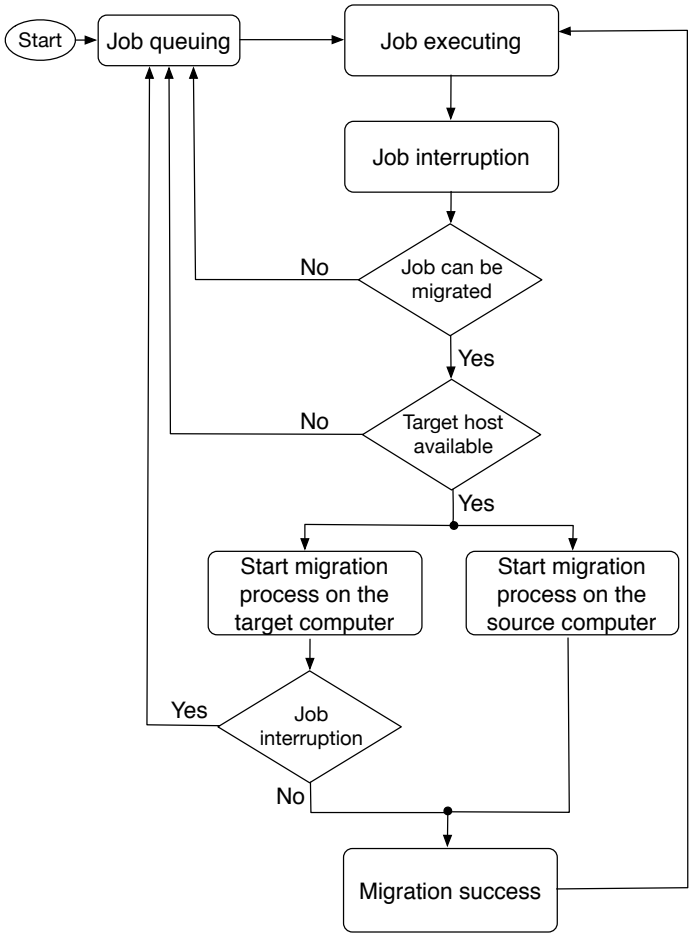


Fig. 6. The flow chart of responsive live migration.

In previous work [21], we measured the live migration time for various workload characteristics on different VMs capacities. Also, we introduced three predictive models to predict the time of live migration. Our results showed that some workload might take a long time to migrate and some cannot be migrated during the workload execution. We have implemented these facts into our simulation tool. When the job cannot be migrated within a short time, the job is removed from the source machine and restart running on another machine.

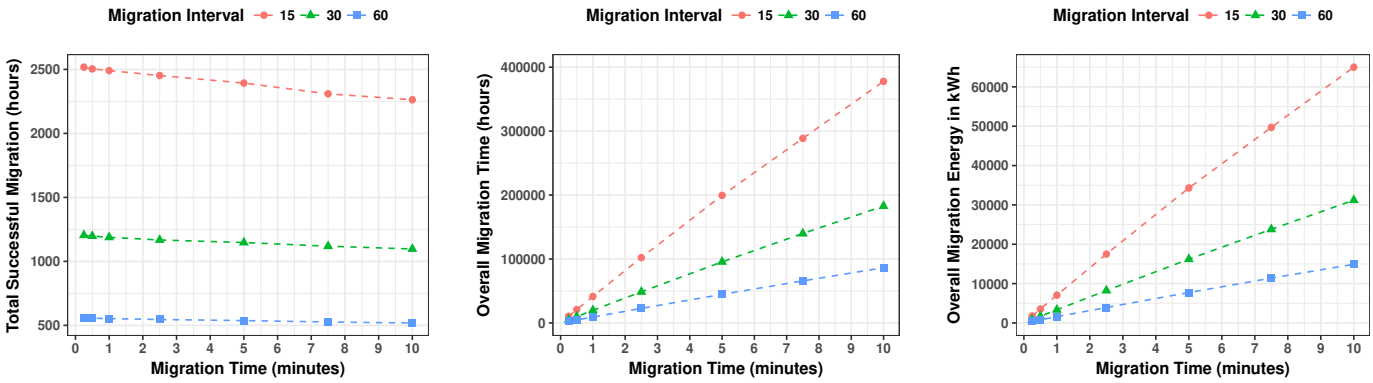


Fig. 7. Total migrations, overall migrations time, and overall energy consumption of successful migrations with different migration durations and intervals.

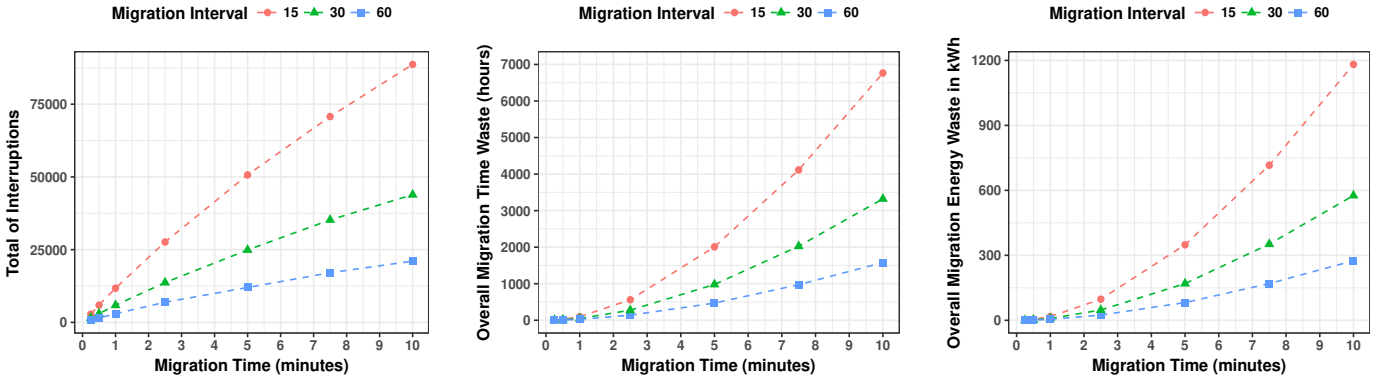


Fig. 8. Total number of interruptions during migration, overall migrations waste time, and overall waste energy of failed migrations with different migration durations and intervals

In this technique, the job rejoins the queue when there is no target machine available in the HTC system pool as well as when an interactive use begins working on the target computer during the migration process. Then, the finish time of the job will be calculated according to Equation 1. Furthermore, Equation 2 and 3 are used to determine the starting time of the job on the target computer and the remaining execution time.

## V. Simulation Outcome

The presented results in this section aim to give a clear understanding of our simulation outcomes. Here, we demonstrate the migration interval technique and we have used the random policy as a baseline to validate our extension to HTSSim. The policy selects the jobs randomly in order to migrate them as well as the selection of target computers. Furthermore, we have not introduced any policy or algorithm to manage the migration process such as when to migrate, which job to migrate, and where to migrate. However, our tool can easily adapt policies and algorithms to reduce system's energy consumption and jobs' makespan.

Figure 7 shows the results of successful migration with different migration durations and intervals. The figure presents the total number of successful job migrations, overall jobs migrations time, and the energy consumption that jobs consumed during the migration process. We see significant reductions in the results when the interval value increases. If the interval time occurs during the job execution time, the system checks its migration policies and starts the migration process if needed. Consequently, a short-running job might have fewer migration attempts than a long-running job which leads to decreases in migration energy consumption and time. Thus, it is crucial to determine when to migrate as well as the reason for migration to avoid inefficient migration.

We acknowledge a significant relation between the migration time and the energy consumption of the migration process. When a job takes a long time to be migrated, it consumes more energy. Furthermore, we see this as an existing problem for many VM consolidation technics where the energy consumption of migration process is not considered. To get over this problem, a policy could be developed, leveraging our observations in [21], to selectively migrate jobs based on estimated runtime and migration time. In doing so, the policy could curtail the costs of migrating jobs, and achieve an overall benefit for the system.

Figure 8 demonstrates the relative impact of jobs interruptions during the migration process where a user logs in to the source or target computer during the migration. We

observe increasing in the number of jobs' interruptions when the migration time of the job increases. As a result, the energy waste of jobs with a long migration time is much higher than jobs associated with short migration time. Similarly, we observed an increase in energy waste of the removed jobs during the migration when the migration time is long. Thus, it worth to check the opening and closing times of student clusters and design policies according to that to decrease the number of users interruption.

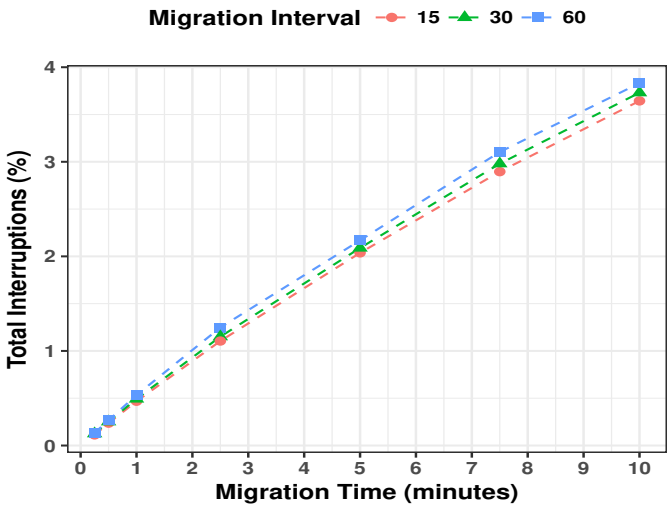


Fig. 9. Proportion of migrations interruption.

Furthermore, the interactive user can login into the source computer or target computer at any time during the migration process regardless the migration interval value. Since we are using random migrations and random target computers to host the migrated job, we cannot predict when the users' login into the system occur. Consequently, the chance of interruptions compares to the number of migrations almost the same between the different migration interval values as illustrated in figure 9. The figure shows the proportion of migrations interrupted in the system. Clearly, the number of interactive users in the system makes an impact on migration success, but there is no immediate link between the interval values and the users' interruptions. As a result, it is crucial to determine when and where to migrate to avoid unsuccessful migrations.

In Figure 10 we explore the impact of the finishing jobs during the migration process. In this situation, the jobs migrations are considered to be worthless migrations. It increases the overall energy consumption of the system. Also, it reduces the number of available machines in the system pool. As a consequence, the jobs might be queued for a long time waiting for an idle computer to become available. Furthermore, number of migration attempts failed due to an unavailable machine for job migration. These failed migration attempts might reduce the overall energy consumption of the system. Our simulation can count these failed attempts during its runtime which gives us the opportunity to understand how many resources usually available for migration when it is needed.

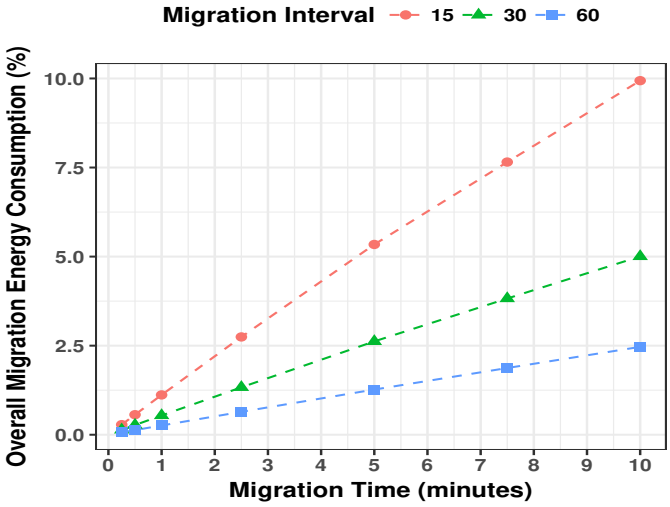


Fig. 11. Impact of jobs migrations on energy.

Figure 11 presents the percentage of total migration energy from the total system energy. When the number of migration increases, the total energy consumption of the system increases as well. Also, the migration time shows a significant increase in total energy consumption when the job takes a long time to be migrated. As the figure shows, jobs migrations might consume up to 10% of total system energy.

## VI. Conclusions

This paper has presented the virtualisation and live migration in HTC systems by extending an existing simulation tool. This allows researchers to evaluate policies and algorithms in order to reduce the energy consumption and jobs makespan by using live migration techniques in HTC environment. A simple random policy has validated our outcomes of the tool. The tool can calculate the overall migration time, the overall energy consumption and waste, and the number of failed migration due to interactive users interruptions or removed jobs. Also, the simulation can count the number of migration attempts when there is not host for migration. Moreover, our results showed that the energy consumption significantly increases when the migration process takes a long time to be finished. Through this paper, we have suggested some approaches to mitigate the energy waste of live migration in a large-scale computing environment. In our ongoing work, we are designing policies and algorithms for migration to reduce energy consumption by avoiding job interruptions. Furthermore, in our current work we consider a single job for each computer; in the future, we hope to assign multiple jobs on one source machine.

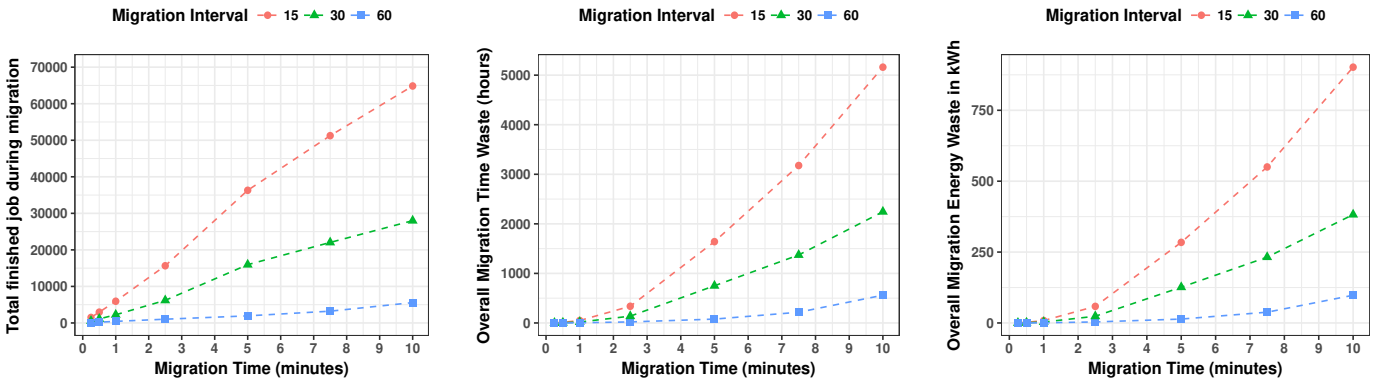


Fig. 10. Total number of jobs finished during migration, overall migrations waste time, and overall waste energy of failed migrations with different migration durations and intervals

## REFERENCES


[1] M. J. Litzkow, M. Livny, and M. W. Mutka, "Condor-a hunter of idle workstations," in *[1988] Proceedings. The 8th International Conference on Distributed*, Jun 1988, pp. 104–111.

[2] D. P. Anderson, "Boinc: a system for public-resource computing and storage," in *Fifth IEEE/ACM International Workshop on Grid Computing*, Nov 2004, pp. 4–10.

[3] W. Gentzsch, "Sun grid engine: towards creating a compute power grid," in *Proceedings First IEEE/ACM International Symposium on Cluster Computing and the Grid*, 2001, pp. 35–36.

[4] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, "Live migration of virtual machines," in *Proceedings of the 2Nd Conference on Symposium on Networked Systems Design & Implementation - Volume 2*, ser. NSDI'05. Berkeley, CA, USA: USENIX Association, 2005, pp. 273–286. [Online]. Available: http://dl.acm.org/citation.cfm?id=1251203.1251223

[5] H. Team, "HTCondor version 8.7.7 manual," Center for High Throughput Computing, University of Wisconsin-Madison, Tech. Rep., 2018. [Online]. Available: https://research.cs.wisc.edu/htcondor/manual

[6] M. Forshaw, A. S. McGough, and N. Thomas, "Energy-efficient checkpointing in high-throughput cycle-stealing distributed systems," *Electronic Notes in Theoretical Computer Science*, vol. 310, pp. 65 – 90, 2015, proceedings of the Seventh International Workshop on the Practical Application of Stochastic Modelling (PASM). [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1571066114000978

[7] G. Aupy, A. Benoit, R. Melhem, P. Renaud-Goud, and Y. Robert, "Energy-aware checkpointing of divisible tasks with soft or hard deadlines," in *2013 International Green Computing Conference Proceedings*, June 2013, pp. 1–8.

[8] X. Ren, R. Eigenmann, and S. Bagchi, "Failure-aware checkpointing in fine-grained cycle sharing systems," in *Proceedings of the 16th International Symposium on High Performance Distributed Computing*, ser. HPDC '07. New York, NY, USA: ACM, 2007, pp. 33–42. [Online]. Available: http://doi.acm.org/10.1145/1272366.1272372

[9] D. Nurmi, J. Brevik, and R. Wolski, "Minimizing the network overhead of checkpointing in cycle-harvesting cluster environments," in *2005 IEEE International Conference on Cluster Computing*, Sept 2005, pp. 1–10.

[10] M. Forshaw, A. McGough, and N. Thomas, "HTC-Sim: A trace-driven simulation framework for energy consumption in high-throughput computing systems," *Concurr. Comput. : Pract. Exper.*, vol. 28, no. 12, pp. 3260–3290, Aug. 2016. [Online]. Available: https://doi.org/10.1002/cpe.3804

[11] H. Casanova, "Simgrid: a toolkit for the simulation of application scheduling," in *Proceedings First IEEE/ACM International Symposium on Cluster Computing and the Grid*, 2001, pp. 430–437.

[12] R. Buyya and M. Murshed, "Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing," *Concurrency and computation: practice and experience*, vol. 14, no. 13-15, pp. 1175–1220, 2002.

[13] W. H. Bell, D. G. Cameron, A. P. Millar, L. Capozza, K. Stockinger, and F. Zini, "Optorsim: A grid simulator for studying dynamic data replication strategies," *The International Journal of High Performance Computing Applications*, vol. 17, no. 4, pp. 403–416, 2003. [Online]. Available: https://doi.org/10.1177/10943420030174005

[14] G. Bisson and F. Hussain, "Chi-sim: A new similarity measure for the co-clustering task," in *2008 Seventh International Conference on Machine Learning and Applications*, Dec 2008, pp. 211–217.

[15] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and experience*, vol. 41, no. 1, pp. 23–50, 2011.

[16] D. Kliazovich, P. Bouvry, and S. U. Khan, "GreenCloud: a packet-level simulator of energy-aware cloud computing data centers," *The Journal of Supercomputing*, vol. 62, no. 3, pp. 1263–1283, Dec 2012. [Online]. Available: https://doi.org/10.1007/s11227-010-0504-1

[17] A. Núñez, J. L. Vázquez-Poletti, A. C. Caminero, G. G. Castañé, J. Carretero, and I. M. Llorente, "iCanCloud: A flexible and scalable cloud infrastructure simulator," *Journal of Grid Computing*, vol. 10, no. 1, pp. 185–209, Mar 2012. [Online]. Available: https://doi.org/10.1007/s10723-012-9208-5

[18] S. H. Lim, B. Sharma, G. Nam, E. K. Kim, and C. R. Das, "Mdcsim: A multi-tier data center simulation, platform," in *2009 IEEE International Conference on Cluster Computing and Workshops*, Aug 2009, pp. 1–9.

[19] A. Beloglazov and R. Buyya, "Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers," *Concurr. Comput. : Pract. Exper.*, vol. 24, no. 13, pp. 1397–1420, Sep. 2012. [Online]. Available: http://dx.doi.org/10.1002/cpe.1867

[20] E. Feller, L. Rilling, and C. Morin, "Energy-aware ant colony based workload placement in clouds," in *2011 IEEE/ACM 12th International Conference on Grid Computing*, Sept 2011, pp. 26–33.

[21] O. Alrajeh, M. Forshaw, and N. Thomas, "Machine learning models for predicting timely virtual machine live migration," in *Computer Performance Engineering*, P. Reinecke and A. Di Marco, Eds. Cham: Springer International Publishing, 2017, pp. 169–183.