Network and Energy-Aware Resource Selection Model for Opportunistic Grids

Izaias de Faria[1], M.A.R. Dantas[1], Miriam A.M. Capretz[2], Wilson A. Higashino[2,3]
Research Laboratory of Distributed Systems (LaPeSD)
[1]Dep. of Informatics and Statistic (INE), Federal University of Santa Catarina (UFSC), Florianópolis, Brazil
izaias.faria@posgrad.ufsc.br, mario.dantas@ufsc.br
[2]Dep. of Electrical and Computer Engineering, Western University, London, Canada
{mcapretz, whigashi}@uwo.ca
[3]Instituto de Computação, Univ. Estadual de Campinas, Campinas, Brazil

***Abstract*—Due to increasing hardware capacity, computing grids have been handling and processing more data. This has led to higher amount of energy being consumed by grids; hence the necessity for strategies to reduce their energy consumption. Scheduling is a process carried out to define in which node tasks will be executed in the grid. This process can significantly impact the global system performance, including energy consumption. This paper focuses on a scheduling model for opportunistic grids that considers network traffic, distance between input files and execution node as well as the execution node status. The model was tested in a simulated environment created using GreenCloud. The simulation results of this model compared to a usual approach show a total power consumption savings of 7.10%.**

Keywords— opportunistic grids; energy consumption; resource selection model.

I. Introduction

Grid systems were proposed as the next generation computing platform and global infrastructure for solving large-scale problems [1]. Grids enable sharing of resources used by scientific researches which have a high cost of ownership [2]. Moreover, they enable on-demand and real-time processing and analysis of data generated by these scientific instruments. This capability significantly enhances the possibilities for scientific and technological research and innovation, industrial and business management, application software service delivery and commercial activities.

Currently, private and public institutions have a large number of computing resources, such as personal computers and workstations, with great capacity for data processing and storage. These computers are idle most of the time and, even when they are in use, usually only a small percentage of their computing capacity is effectively used as confirmed by Beauvisage [3]. *Opportunistic Grids* are computing systems that provide the means to use an installed base of ordinary computers to execute high performance computing applications, thus leveraging their available idle computing power [4]. The focus of opportunistic grid middleware is not on the integration of dedicated computer clusters (e.g. Beowulf [5]) or supercomputing resources, but on taking advantage of the idle computing cycles of ordinary computers and workstations that can be spread across a number of several administrative domains.

As systems scale and energy consumption increases, these technologies have the potential to damage our ecosystems. In the past few years' energy consumption has become one of the main problems that the computer industry has faced due to increasing investment for maintaining the computers [6] and the reduction in performance due to increasing temperatures [7]. Energy consumption is not only determined by hardware efficiency, but it is also dependent on the resource management system (RMS) deployed on the infrastructure.

Scheduling of tasks on multiprocessors (done by the RMS) is usually focused on optimizing common performance criteria such as total completion time and turnaround time. This problem is generally well understood and has been studied for decades. Many research results exist for different variations of this scheduling problem; some of them provide theoretical insights, while others give hints for implementing real systems. This paper presents an approach to improve energy efficiency in opportunistic grids, using a resource-selection model based on the cost of transporting bits across the network and the cost of the actions required for resource selection.

The following sections of the paper are organized as follows: Section II provides the required background knowledge and concepts of scheduling in opportunistic grids. Related work is then discussed in Section III. The proposed model is formally presented in Section IV. The simulation environment and experimental results are presented in Section V. Finally, Section VI concludes the paper and presents future research directions.

II. Background

A. Resource Management System (RMS)

A grid [8] is a very large-scale, generalized distributed network-computing system that can scale to Internet-size environments with machines distributed across multiple organizations and administrative domains. The emergence of a variety of new applications demands that grids support efficient data and resource management mechanisms. The RMS is a module central for the grid operation which is responsible for managing the pool of resources available to the grid, such as processors, network bandwidth, and disk storage. In order to support a variety of applications efficiently, the RMS must address issues such as fault-tolerance and stability [9].

In a grid, the resource pool can include resources from different providers, therefore requiring all resource providers to trust the RMS. Additionally, the RMS is responsible for handling the various resources while adhering to the different usage policies. Fig.1 shows the interaction of the RMS with the resource pool.

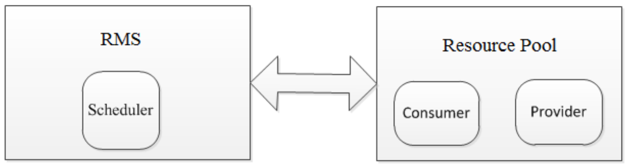

Fig. 1. RMS interaction with Resource Pool

Applications may request resources either directly or indirectly from the grid. Such resource requests are considered as tasks by the grid. In practice, a grid RMS may be required to handle different tasks using different policies. In general, requiring the RMS to support multiple policies can compel the scheduling mechanisms to solve a multi-criteria optimization problem.

Ideally, a scheduler should ensure that tasks are executed making optimum use of available resources and finish in the shortest time possible, while respecting time constraints or other policies applied to the tasks. Due to the heterogeneous and dynamic nature of the grid, the end-user must establish the requirements to be met by the target resources (discovery process) and criteria to rank the matched resources (selection process). The attributes needed for resource discovery and selection must be collected from information services. Usually, resource discovery is based only on static attributes such as operating system and architecture, while resource selection is based on dynamic attributes such as disk space and processor load.

B. *Scheduling strategies*

Ideally, the development of scheduling algorithms should be focused on a specific set of applications, because if the details of the applications to be scheduled are unknown, the algorithm can negatively influence the performance. Because the computational grid environment has certain special features such large number and heterogeneity of resources and dynamic performance, resource allocation in this environment becomes challenging. Taking into account the challenges and known strategies in resource scheduling in computational grids, certain specific algorithms are more widely used in such environments, such as:

Workqueue (WQ): When a resource becomes available, the choice of which task shall be submitted for execution is made at random. The goal is forto assign a greater number of tasks to faster machines, causing slower machines to work with lighter loads [10].

Workqueue with Replication (WQR): WQR is similar to WQ in that tasks are sent to run on the machines that are available. The moment a machine finishes executing a task, it receives a new execution. The difference between WQ and WQR occurs when a machine becomes available and there are no other tasks in line to be executed. At this time WQR initiates replication of tasks that are still running. Once the original task or one of its replicas completes execution, the others are disrupted [10].

Sufferage (Suff) [11]: determines how much each task could be impaired if it were not scheduled on the processor that would run it most efficiently. The *sufferage* value of each task is the difference between its best and second-best Completion Time (CT), considering all processors in the grid. The task with the highest *sufferage* value have execution priority.

XSufferage (XSuff): *XSufferage* is a modification of **Sufferage**, the main difference being the method used to calculate *Sufferage*. *XSufferage* considers the transfer of input data in its calculation of task execution times. In other words, it uses CPU-related information and the estimated task execution time used by *Sufferage* plus the bandwidth available on the network that connects the grid resources.

This paper proposes an algorithm derived from the WQ strategy. The algorithm proposed in Section 4 is focused on resource allocation; as soon as an execution is over, the algorithm allocates resources for the next execution in line. Unlikely WQR the proposed algorithm does not consider execution replicas. Also the strategy of the proposed model considers the cost of the transfer of input files for the execution node similar to the XSuff strategy.

C. *Metrics*

The general purpose of consumption metrics is to provide an overview of the energy efficiency of the infrastructure. In the present article, these metrics seek to prove the validity of the proposed algorithm.

The calculation model used in this work is based on the Baliga *et al.* [12] model, which considers the energy per bit as a fundamental measure of consumption. The energy *Ec* needed to carry a bit over the network can be expressed as:

$$E_C = 3 \times 3 \times \left(\frac{P_{les}}{C_{les}} + \frac{3P_{es}}{C_{es}} + \frac{P_g}{C_g}\right) \qquad (1)$$

where, P_{les}, P_{es} and P_g are respectively the energy consumed by small Ethernet switches, Ethernet switches and routers. C_{les}, C_{es} and C_g represent the capabilities of these pieces of equipment in bits per second.

The first factor 3 in equation 1 takes into account the energy requirement for redundancy (with a numerical value of 2) as well as cooling and other overheads (assumed to have a value of 1.5). The second factor of 3 is related to the combination of the energy consumed by enterprise network switches and the LAN network grid.

D. *Simulation*

Simulation is a practical way to analyze algorithms on large-scale distributed systems of heterogeneous resources. Unlike using the real system in real time, simulation avoids the overhead of coordinating real resources; therefore, it does not add unnecessary complexity to the analytical mechanism. Simulation is also effective in working with very large problems that would otherwise require the involvement of a large number of active users and resources.

Among the available simulators, GridSim [13] and GreenCloud [14] were selected for this research. Some of the main features of GridSim include modelling heterogeneous computational resources, scheduling tasks based on time- or space-sharing policies, differentiated network service, and simulation of workload traces from real systems. This simulator facilitates integrated studies of novel on-demand data replication strategies and task scheduling approaches.

GreenCloud is known for advanced energy-aware studies in the area of cloud computing. GreenCloud extracts, aggregates, and makes available information about the energy consumed by the computing and communication elements of data center. In particular, it focuses on accurate capture of the communication patterns of current and future data center architectures. In the present article GreenCloud has been chosen because it is designed to capture details of the energy consumed by data center components.

III. Related Work

Montero *et al.* [15] analyzed the relevance of resource proximity in the resource-selection process to reduce the cost of file staging. They also studied opportunistic migration when a new resource became available on the grid. In this situation, the performance of the new host, the remaining execution times of applications, and the proximity of the new resource to needed data are considered critical factors in deciding whether task migration is feasible and worthwhile. Along with the opportunistic grid environment and the proximity of the new resource to needed data it is also relevant to consider the resource status because depending on these, the execution migration process can become unfeasible.

Ponciano and Brasileiro [16] investigated energy-aware scheduling, sleeping and wake-up strategies in opportunistic grids. Sleeping strategies are used to reduce grid energy consumption during idle periods; wake-up strategies are used to choose a set of resources to fulfill a workload demand; and scheduling strategies are used to decide which tasks to schedule on the available machines. These strategies could also be taken into account in the resource selection process to enable a wider set of selection options.

Nesmachnow *et al.* [17] introduced a new formulation of the scheduling problem for multicore heterogeneous computational grid systems. In this formulation energy-consumption minimization, along with the *makespan* metric, was considered. The authors also adopted a two-level model, in which a meta-broker agent receives all user tasks and schedules them on the available resources belonging to different local providers. HTCondor [18] used a similar model in which a scheduler chooses resources to execute tasks and has a module that enables users to set machines to a low-power state (hibernation). However the HTCondor scheduler does not consider machines in hibernating state in the selection process, but only idle resources.

Yu *et al.* [19] presented a survey of workflow scheduling algorithms for grid computing. They categorized existing grid workflow scheduling algorithms as either best-effort-based scheduling or QoS-constraint-based. Best-effort scheduling algorithms target community grids to which resource providers provide free access. Because the service-provisioning model of the community grids is based on best effort, quality of service and service availability cannot be guaranteed. Additionally, Yu also discussed several techniques for using scheduling algorithms in dynamic grid environments.

Batista *et al.* [20] presented simulation of an opportunistic computational grid made up of digital receivers. The goal was to distribute different types of applications to run efficiently on the heterogeneous digital receivers that made up this environment. Three scheduling policies were compared: Based on Capacity (BC) and Best Arrangement (MA), proposed in [23], and WorkQueue (WQ). Simulations and analyses showed that an increase in the number of users and the system load required a more efficient scheduling policy, because it generated a greater overhead in the system.

IV. Proposed Network and energy-aware resource selection Model and algorithm

A. *Energy Efficiency Model (EEM)*

The energy consumption of a submission refers to an estimate of energy that will be expended by the environment to initiate the task execution in a grid node. Besides the energy necessary to transport the bits of the input files over the network, the execution machines can be in different states, at different levels in the network, and at different distances from the node with the input files. All these factors must be considered to calculate the energy consumption of a submission. The total consumption *C* of a submission is measured in watts and can be calculated as:

$$C = (E_c * A) + (E_c * E) + P + T + W \qquad (2)$$

where E_c is the cost of transporting a bit over the network [12],; *A* is the size of the input file in bits; *E* is the size of the executable in bits; *P* is the consumption of the computer where the task is executed; *T* is the cost to turn on the machine if it is off; and *W* is the cost to awaken the machine if it is hibernating. Note that Ec is directly proportional to the distance (in levels) from the execution node to the input-file node. In other words, as this distance grows, so does the transportation cost.

The total consumption C must be calculated for each execution node, but not all values are different; in certain cases, it is possible that two or more nodes have the same total energy consumption for a submission. In such cases, network latency is used as a decisive factor to choose the best available resource by assuming weights for energy consumption and network latency:

$$E = (P_c * C) + (P_l * L) \qquad (3)$$

where E is the choice factor, the closer to zero the better, which means less energy consumption added to network latency (considering weights); Pc is the weight associated with energy consumption; C is the energy consumption associated with the change of status and data transfer over the network presented in Equation (2); Pl is the weight assigned to network latency; and L is the average latency between the execution node and the node containing the input files.

B. *Algorithm*

To select the most economically viable resource, an algorithm is proposed here. The proposed algorithm is based on the **WQ** strategy [10], but unlike this strategy, the algorithm uses information about the environment such as energy consumption of resources and node status. Fig. 2 shows the algorithm, which continues to run until there are no more tasks in the execution queue. First, it assembles a list of resources that meet the requirements of the submission, taking into account resources in idle state, turned off and hibernating.

Algorithm 1 Energy-Aware resource selection

```
1:  while jobs left in the scheduler queue:
2:    scheduler queries the available resources
3:    if has resource available
4:       calculate the most economical execution node (list of available resources)
5:       if node status = off
6:          turn on the node
7:       else if node status = Hibernating
8:          wake up the node
9:       allocate resource (selected node)
10:   else
11:      remains in the queue
12: end while
```

Fig. 2. Resource selection algorithm

If resources that satisfy the execution requirements are found, then the selected nodes are input to the proposed model to determine the least costly resource, considering the cost of data transfer to the execution node and its state. If the node is off the cost of turning the node on must be considered; similarly, if the node is hibernating, the cost to awaken the node must be considered. The algorithm then sends an order to the node according to its state, and finally the resource is allocated for execution. If no resources that satisfy the requirements of the execution are available in the pool, the execution is kept in the scheduler queue.

V. Environment and Experimental Results

A. *Simulation environment*

This research used HTCondor [18] in its default configuration as a scheduler for its environment.The network topology is depicted in Fig. 3. It represents an environment where resources from other sub-networks are available to the grid. In this research, 30 personal computers scattered through three sub-networks were used, where PC 1 was the master node and PC 9 was used to store the input files; hence, these were not considered execution nodes. Fig. 3 also shows the network topology by level and the corresponding switches. The environment has eight access switches and one aggregation switch. Table I shows the computers divided by levels where the first digit refers to the Aggregation switch, and the other digits refer to the Access switches; for example, PC 1 is on level 1.1, which means Aggregation switch 1, sub-network 1, and so on.

Table II gives the specifications of the equipment shown in Fig. 3 as well as the power consumption and capacity of the switches. The computers are assumed homogeneous to show the efficiency of the proposed algorithm. The information presented in Table II was taken from equipment specifications and the models chosen were popular models of switches for corporate networks.

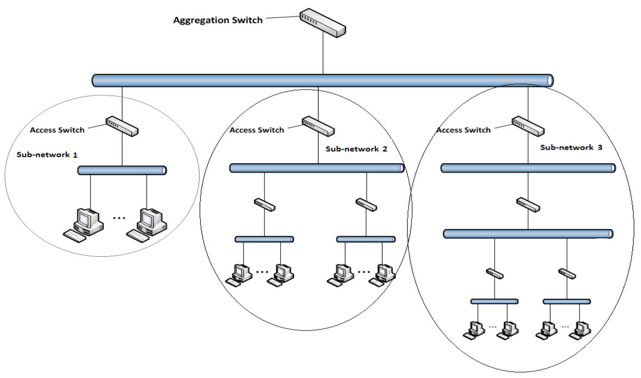

Fig. 3. Network topology

To simulate the study environment, a modified version of GreenCloud [14] was used. This modified version was able to monitor the energy consumption of the proposed model. Besides GreenCloud, GridSim was used to simulate network traffic [13]. These tests did not consider the time for actions such as booting or awakening machines. This simulation focused on finding the best cost/efficiency trade-off considering data traffic in the network, node status, and network latency.

TABLE II. EQUIPMENT SPECIFICATION

Equipment	Brand	Capacity	Consumption (Watts)				
Switches							
Access	HP/ 3COM SWITCH 4210G	128 GB/s	370				
Aggregation	HP/ 3COM SWITCH 6600	176 GB/s	261				
Personal Computer							
PC - Desktop HP	Internal Conguration	-	idle	On Max	Booting	Shutting	Awaken
	Core 2 Duo E6850 @ 3.00 GHz 2Gb DDR2/667 MHz		64	132	100	100	85

B. *Experimental results*

These tests used an environment with random status for the pool machines. The only restriction on status was to have at least one node in hibernating status for each test. Table III shows the default pool. The master node, nodes that store the input files, and the nodes that are in busy state are not considered by the scheduler in the scheduling process.

TABLE I. NETWORK TOPOLOGY BY LEVELS

Network Topology	PC																													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
Level	1.1	1.1	1.2.2	1.2.1	1.3.1.1	1.3.1.2	1.1	1.2.2	1.1	1.3.1.1	1.2.1	1.3.1.1	1.2.1	1.2.2	1.3.1.1	1.3.1.1	1.2.2	1.3.1.2	1.3.1.2	1.1	1.2.1	1.3.1.2	1.3.1.2	1.2.1	1.3.1.2	1.2.2	1.3.1.1	1.3.1.1	1.2.2	1.3.1.2
Aggregation Switches	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Access Switches	1	1	2	2	3	3	1	2	1	3	2	3	2	2	3	3	2	3	3	1	2	3	3	2	3	2	3	3	2	3

TABLE III. DEFAULT POOL

Initial State	PC																													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
Test 1	■	■	□	□	□	■	□	■	■	□	▨	■	□	□	■	□	■	□	□	▨	■	□	□	■	□	■	□	□	▨	■
Test 2	■	□	□	■	■	□	■	□	■	▨	■	□	□	□	□	■	■	□	■	□	□	▨	■	□	□	□	□	■	■	□
Test 3	■	▨	■	□	□	□	■	▨	■	■	□	■	□	▨	■	□	□	□	■	▨	□	■	□	■	□	▨	■	□	□	□

■ Busy
Master
Entry Files
▨ Hibernating
□ Idle

To analyze the effectiveness of the proposed model, two sets of tests were performed. Each test was executed ten times to analyze the allocation strategy of both sets. In the first set, the standard version of HTCondor was used. Three tests with different input-file sizes were carried out, the first with 10 MB, the second with 100 MB and third with 1 GB. As for the weights, the assumption used was 60% for power consumption and 40% for network latency. Table IV shows the resource allocation after the tests in selection order, and Table V shows the energy consumption in watts for each submission.

TABLE IV. SET 1 ALLOCATED RESOURCES

Set 1	PC																													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
Test 1	■	■	1	2	3	■	4	■	■	5	▨	■	6	7	■	8	■	9	10	▨	■	□	□	■	□	■	□	□	▨	■
Test 2	■	1	2	■	■	3	■	4	■	▨	■	5	6	7	8	■	■	9	■	10	□	▨	■	□	□	□	□	■	■	□
Test 3	■	▨	■	1	2	3	■	▨	■	■	4	■	5	▨	■	6	7	8	■	▨	9	■	10	■	□	▨	■	□	□	□

■ Busy
Master
Entry Files
▨ Hibernating
□ Idle

TABLE V. SET 1 ALLOCATED RESOURCE CONSUMPTION

Test	Consumption (Watts)										Total Consumption (Watts)
	1	2	3	4	5	6	7	8	9	10	
1	64.9135	64.9135	65.1736	64.9135	65.1736	64.9135	64.9135	65.1736	65.1736	65.1736	650.4355
2	66.601	73.135	75.736	73.135	75.736	73.135	73.135	75.736	75.736	66.601	728.686
3	155.35	181.36	181.36	155.35	155.35	181.36	155.35	181.36	155.35	181.36	1683.55

As is well known, HTCondor uses a simple matching algorithm, and therefore the first available resource is always selected. Nodes with a status other than idle are not considered for resource allocation; in this case, even when a less costly resource is available, it will not be selected unless it is in idle state.

In the second set of tests, the model proposed in this article was used. Table VI shows the allocation of resources in selection order, and Table VII shows the energy consumption for each submission in the pool for Set 2.

TABLE VI. SET 2 ALLOCATED RESOURCES

Set 2	PC																													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
Test 1	■	■	3	2	□	■	1	■	■	8	▨	■	5	4	■	9	■	6	□	▨	■	10	7	■	□	■	□	□	▨	■
Test 2	■	1	5	■	■	□	■	4	■	▨	■	□	8	6	□	■	■	10	■	2	9	▨	■	3	□	7	□	■	■	□
Test 3	■	▨ 1	■	3	□	□	■	▨ 9	■	■	4	■	7	▨ 10	■	□	5	□	■	▨ 2	8	■	□	■	□	▨	■	□	6	□

■ Busy
Master
Entry Files
▨ Hibernating
□ Idle

TABLE VII. SET 2 ALLOCATED RESOURCE CONSUMPTION

Test	Consumption (Watts)										Total Consumption (Watts)
	1	2	3	4	5	6	7	8	9	10	
1	64.2601	64.9135	64.9135	64.9135	64.9135	65.1736	65.1736	65.1736	65.1736	65.1736	649.7821
2	66.601	66.601	73.135	73.135	73.135	73.135	73.135	73.135	73.135	75.736	720.883
3	111.01	111.01	155.35	155.35	155.35	155.35	155.35	155.35	176.35	176.35	1506.82

As shown in Table VII, the proposed model chose the most energy-efficient resource available in the pool. It is also apparent that , during test 3 the model chose to wake up nodes four times (nodes 2, 8, 14, and 20) instead of sending the submission to more distant nodes. That happened because waking up the nodes was less costly than sending the submission to another sub-network considering that the input file was 1 GB in size. According to the results of Tests 1, 2, and 3, the proposed model obtained respectively 0.10%, 1.10%, and 10.50% energy savings over HTCondor. The total energy-consumption values are shown in Fig. 4.

Fig. 4 shows that in tests 1 and 2, the proposed model achieved a slightly superior performance than the standard HTCondor. By analyzing the values of both tests, it was clear that in the worst-case scenario, the algorithm obtained the same power-consumption performance.

As input files size grows, the performance of the proposed model remains superior to the standard HTCondor. The biggest advantages of the proposed model can be observed during test 3, where HTCondor disregards the machines next to node 9 (input-file storage node) due to their status, although they are less costly to use than machines farther from node 9. In this test, the proposed model obtained a better result because it considered machines in different states, such as node 2, which was hibernating before selection and had to be awakened.

Another relevant conclusion from the results is the influence of network data traffic on resource selection. The greater the distance to be traveled, from the input-file node to the execution node, the higher will be the energy consumption in the grid before the start of execution. This underlines the fact that the scheduler must consider not only the first match from the list of available resources as the HTCondor scheduler does, but should also be concerned about resources in different states and data traffic over the network. In this case, latency can influence resource selection, especially as data size grows.

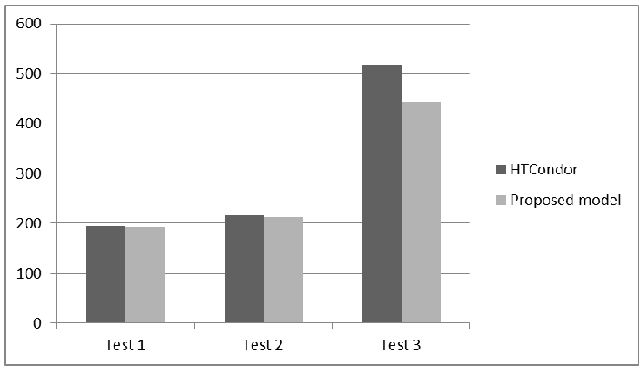

Fig. 4. Total consumption HTCondor vs. Proposed model

The proposed model makes it possible to give weights to power consumption and to network traffic. In the present case, the proposed model was applied assuming 60% of energy consumption and 40% network latency as resource-selection criteria. These values can be changed, which possible implies a change in the resources selected. Comparing the total energy expended in both scenarios, the proposed model attained a total savings of 7.10% compared to the standard HTCondor matching algorithm.

VI. CONCLUSION AND FUTURE WORK

This paper proposed a network and energy-aware resource selection model for an opportunistic grid. To validate the model, an algorithm has been proposed that selects a resource based on the cost associated with transfer of bits over the network, resource status, and network traffic. The main goal of the proposed approach was to reduce the total energy consumption of the grid. The proposed model aims to do so by reducing the cost associated with the process prior to the execution of tasks in the grid, by choosing the least costly resource in the pool. This usually implies choosing the closest execution node to the input-file server considering network traffic and node status.

The proposed model was evaluated through the GreenCloud simulator. The results showed that the model is capable of choosing the best available resource and in the worst case has the same performance than the original HTCondor approach. The model reached successfully a level of saving around 7.10% when compared with the HTCondor scheduling. Due to the influence of network latency in cloud environments, the authors are planning to port the model and algorithm to such environments and to adapt the proposed model to implement migration of executing tasks. Moreover, tests in a real scenario will also be considered in future work.

REFERENCES

[1] A. Luther, R. Buyya, R. Ranjan, and S. Venugopal. High-Performance Computing: Paradigm and Infrastructure. Wiley Series on Parallel and Distributed Computing. John Wiley & Sons, Hoboken, NJ, USA, October 2006. ISBN 978-0-471-65471-1.

[2] The Large Hadron Collider, CERN, January 2004. URL http:==lhc-new-homepage. web.cern.ch/lhc-new-homepage/.

[3] T. Beauvisage, "Computer usage in daily life". In Proc. Intl. Conf. on Human Factors in Computing Systems (Apr. 2009), pp. 575–584.

[4] A. Goldchleger, F. Kon, A. Goldman, M. Finger, and G. C. Bezerra. Integrade: object-oriented grid middleware leveraging idle computing power of desktop machines. Concurrency and Computation: Practice & Experience. vol. 16, pp. 449-459, 2004.

[5] Beowulf, http://www.beowulf.org/. January 2014.

[6] P. Kurp,"Green computing". Communications of the ACM, vol. 51, no. 10, pp. 11-13, 2008.

[7] K. Sharma and S. Aggarwal,"Energy-Aware Scheduling on Desktop Grid Environment with Static Performance Prediction". Proceeding of the 2009 Spring Simulation Multiconference(SpringSim '09), Article No. 105, 2009.

[8] I. Foster, C. Kesselman C (eds.), The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann: San Francisco, CA, 1999.

[9] P.K. Sinha. Distributed Operating Systems: Concepts and Design. IEEE Press: New York, NY, 1997.

[10] D.P. da Silva, W. Cirne, and F.V. Brasileiro, "Trading Cycles for Information: Using replication to Schedule Bag-of-Tasks Applications on Computational Grids". In: Euro-Par Parallel Processing, 2003, pp. 169-180.

[11] T. Casavant and J. Kuhl, "A Taxonomy of Scheduling in General-purpose Distributed Computing Systems". IEEE Transactions on Software Engineering, vol. 14, Fev 1988, pp. 141-154.

[12] J. Baliga, R.W.A. Ayre, K. Hinton and R.S. Tucker, "Green Cloud Computing: Balancing Energy in Processing, Storage, and Transport" Proceedings of the IEEE , vol.99, no.1, pp.149-167, Jan. 2011.

[13] Gridsim, http://www.buyya.com/gridsim/, January 2014.

[14] GreenCloud, http://greencloud.gforge.uni.lu/, January 2014.

[15] R.S. Montero, E. Huedo, and I.M. Llorente. "Grid Resource Selection for Opportunistic Job Migration". 9th International Euro-Par Conference Klagenfurt, Austria, August 26-29, 2003 Proceedings, pp 366-373.

[16] L. Ponciano and F. Brasileiro, "On the impact of energy-saving strategies in opportunistic grids", in Energy Efficient Grids, Clouds and Clusters Workshop (E2GC2), 2010, Brussels, Belgium. Proceedings of the 11th ACM/IEEE International Conference on Grid Computing, 2010. pp. 282–289.

[17] S. Nesmachnow, B. Dorronsoro, J.E. Pecero, P. Bouvry, "Energy-aware Scheduling on Multicore Heterogeneous Grid Computing Systems". Journal of Grid Computing, vol 11, no. 4, pp 653-680, 2013.

[18] HTCondor. http://research.cs.wisc.edu/htcondor/. January 2014.

[19] J. Yu, R. Buyya, K. Ramamohanarao, "Workflow Scheduling Algorithms for Grid Computing", Metaheuristics for Scheduling in Distributed Computing Environments Studies in Computational Intelligence, vol 146, pp. 173-214, 2008.

[20] B.G. Batista, F.C. Teixeira, M.J. Santana, R.H.C. Santana, "Scheduling algorithms for opportunistic computational grid based on television digital receiver.", IEEE International Conference on High-Performance Computing and Simulation (HPCS), 2013, pp. 584-591.