On the Impact of Energy-saving Strategies in Opportunistic Grids

Lesandro Ponciano   Francisco Brasileiro
Universidade Federal de Campina Grande
Departamento de Sistemas e Computação
Campina Grande, PB, Brazil
lesandro@lsd.ufcg.edu.br, fubica@dsc.ufcg.edu.br

***Abstract*—Opportunistic grids are distributed computing infrastructures that harvest the idle computing cycles of computing resources geographically distributed. In these grids, the demand for resources is typically bursty. During bursts of resource demand, many grid resources are required, but on other times they remain idle for long periods. If the resources are kept powered on even when they are neither processing their owners workload nor grid jobs, their exploitation is not efficient in terms of energy consumption. One way to reduce the energy consumed in these idleness periods is to place the computers that form the grid in a "sleeping" mode which consumes less energy. We evaluated two sleeping strategies, denoted: standby and hibernate. Resources that comprise an opportunistic grid are normally very heterogeneous, and differ enormously on their processing power and energy consumption. It opens the possibility of implementing scheduling strategies that take energy-efficiency into account. We consider scheduling in two different levels. Firstly, how to choose which machine should be woken up, if several options are available. Secondly, how to decide which tasks to schedule to the available machines. In summary, our results presented a significant reduction in energy consumption, surpassing 80% in a scenario when the amount of resources in the grid was high. Moreover, this comes with limited impact on the response time of the applications.**

***Keywords*-Bag-of-Tasks Scheduling; Energy-saving Strategies; Opportunistic Grids; Sleeping Modes**

I. Introduction

Opportunistic grids are distributed computing infrastructures that harvest the idle computing cycles of computing resources geographically distributed. This kind of system is suitable for the execution of the so called Bag-of-Tasks (BoT) applications — those parallel applications that can be divided into a large number of independent tasks which do not communicate with each other. Efficient execution of BoT applications over opportunistic grids is achieved because a simple workqueue scheduling strategy (or variations of it) suffices to produce an appropriate mapping of tasks to computing resources [24]. Moreover, fault tolerance is easily achieved by simply assigning a task that has failed to another processing resource that is available. Furthermore, if the execution time of each task is sufficiently small, this strategy is effective even if tasks are not checkpointed regularly, and are simply restarted from the scratch after a failure. Applications with these characteristics are abundant in a variety of settings, including data mining, simulations, fractal calculations, computational biology, computer imaging, among many others. This, together with the current availability of an enormous amount of under utilized processing resources, has made very popular the use of opportunistic grids to run scientific BoT applications.

This kind of grids can be broadly divided in two classes, regarding the motivation that the resource owners have to make their resources available to the grid. In the first class, resource owners have an altruistic behaviour, and donate their resources without expecting any measurable return in exchange. These systems are commonly called *volunteer computing* platforms and were popularized firstly by the pioneer SETI@home project [22], and then by a number of subsequent projects that are powered by the BOINC middleware [5], such as the World Community Grid[1] and Ibercivis[2]. In the second class of grids, resource owners expect to be able to have their donations reciprocated by the recipients of these donations. Peer-to-peer (P2P) grids, such as those powered by the OurGrid middleware [6], are the main representatives of this class of grids. These distinct motivations also lead to a distinct profile of the resources that are available in the two classes of systems, with the personal computers of home users comprising the majority of the resources made available to volunteer computing grids, while the desktops connected by a local area network comprise the resources that are managed by each peer of a P2P grid. The implications that these differences have in the energy efficiency of opportunistic grids is the main issue that we investigate in this paper.

A recent study shows that personal computers are rarely completely idle, since there is always some background service that is active, even when the user is not directly interacting with the computer [21]. On the other hand, most of the time, a substantial fraction of the CPUs that power these resources is not used, creating the adequate conditions to use them opportunistically. Let the baseline energy consumption of these computers be that consumed when they are almost idle. Since the difference between the energy consumed when the computer is fully used and the baseline is small, compared to the baseline consumption,

[1]http://www.worldcommunitygrid.org/
[2]http://www.ibercivis.es/.

the computation that is performed by volunteer computing systems that use the idle cycles of these computers is very efficient in terms of energy consumption.

Unfortunately, this is not necessarily the case for P2P grids. Desktop computers of corporate local area networks, normally used in this kind of grids, spend a lot of time completely idle. Thus, if for one side this also makes them suitable for being explored opportunistically, the lack of a background processing that "pays" for the baseline energy consumption makes their exploitation less energy-efficient than that achieved with personal computers. Moreover, previous studies show that P2P grids are rarely operated at full utilization at all times [10], and the demand for resources is typically bursty [15], [17], [18]. During bursts of resource demand, many grid resources are required, but on other times they remain idle for long periods. If the resources are kept powered on even when they are neither processing their owners workload nor grid jobs, their exploitation is even less energy-efficient. Managing desktop energy consumption is an area of both active research and commercial interest; however, desktops still consume a significant amount of energy when idle [9].

One way to reduce the energy consumed in these idleness periods is to place the computer in a "sleeping" mode that consumes less energy. Several popular operating systems, for example Microsoft Windows [16] and Linux distributions [19], use a standard called Advanced Configuration and Power Interface (ACPI) to implement this kind of power management. In a P2P grid, the agent that manages each machine in the grid could take the decision of switching the machine to a sleeping mode when there are no tasks to be processed, while the peer that manages a particular machine could take the decision to bring it back to operation, provided that there is work to be performed, by sending a message through the network (Wake-on-LAN). In this setting, the only machine that would need to be always on is the one where the peer runs. This machine could be shared with other services, amortising the energy spent to run the peer agent. Two ACPI sleeping states[3] meet those requirements: *standby* (S3), and *hibernate* (S4).

Obviously, this procedure impacts the response time of the applications — also referred to as the applications' *makespan*. Standby, or suspend-to-RAM, is a low-latency sleeping state, since the RAM is kept powered on. On the other hand, hibernate, or suspend-to-disk, consumes less power, but requires a larger latency in order to return the machine to the operational state, since information needs to be retrieved from the disk. This wake up cost introduces delays in the makespan of the jobs submitted to the grid. In this paper we have investigated this trade-off. Our results have shown that both strategies can provide substantial reduction on the energy consumed by the grid to run a given workload. Moreover, this economy comes with no significant increases on the jobs' makespan.

Resources that comprise a P2P grid are normally very heterogeneous, and differ enormously on their processing power and energy consumption [7], [14]. Since not all machines in a P2P grid are used at the same time, this opens the possibility of implementing scheduling strategies that take energy-efficiency into account. We consider scheduling in two different levels. Firstly, the peer agent needs to define a strategy to choose which machine it should wake up, if it has several options available. We consider two possibilities: Most Recently Sleeping (MRS), in which the peer chooses the machine that has most recently changed its state (this is the approach currently used by the OurGrid middleware); or, Energy Aware (EA), in which the peer chooses the most energy-efficient machine first. A second level of scheduling is performed when deciding which task to schedule to the available machines. There are several strategies used to schedule BoT applications, for example: First Come First Served (FCFS), Fastest Processor to Largest Task (FPLT), etc. However, to the best of our knowledge, there is no previous work that provides information about the impact of scheduling strategies on the energy consumption on opportunistic grids. In this paper we analyze this impact for two well-known scheduling strategies (FCFC and FPLT), and for two others that we propose, namely, Most Energy-Efficiency First (MEEF) and Most Energy-Efficient to Largest Task (MEELT).

Our results have shown that both EA and MRS provide similar results in terms of the impact on the jobs' makespan. However, EA showed better energy savings than MRS, in scenarios where the contention for resources in the grid was low. On the other hand, the different task scheduling strategies did not show a meaningful disparity considering both metrics used.

The remainder of this paper is organized as follows. We review related work in Section II. Then, in Section III, we introduce the simulation model that we have used to analyze the energy efficiency of P2P opportunistic grids. The experimental setup chosen is described in Section IV, while the results of the experiments conducted and the analysis of these results are discussed in Section V. Finally, conclusions are presented in Section VI.

II. Related Work

In recent years, several approaches have been proposed to analyze and reduce power consumption in computer systems through better design of both the hardware and the software of these systems, as well as by improving the environment where they are deployed. Regarding the hardware, researchers have tried to develop computers that are more economical in terms of energy consumption. On the software side, one way to save energy is to use more efficient

[3]We highlight only the ACPI states relevant to this study; for information about other states, see [8].

algorithms avoiding excessive processing [4]. Regarding the deployment environment, the entire support infrastructure is considered, mainly including the cooling system of machines.

Some of these studies focus on power down mechanism to save energy in computer systems [3], [12], [20]. Generally, the studies on power down mechanism focus on identifying the optimal threshold times to transition to low-power sleeping modes during an idle period [3], and selecting one state among multiple sleeping states for a single computer [12] or a cluster of computers [20]. These studies provided a framework on the effects of sleeping strategies on the latency and power consumption of these systems. In this paper, we have evaluated the use of these strategies to reduce the energy consumption of computers when they are not in use by a P2P grid.

There are some studies that deal with reduction of energy consumption in grids and clusters using sleeping strategies (or power down modes) [15], [17], [18], [23]. Lammie et al. [15] analyze the effect of automated node scaling (turn on/off), CPU frequency scaling, and job assignment on power consumption and performance of grid workload on larges clusters. Orgirie et al. [17], [18] propose an energy-aware model to reduce the global energy consumption of a large scale experimental grid. The model considers an environment where users need to reserve resources and the resources are dedicated to the user during the reservation period.

Other studies focus on scheduling strategies that are aware of the energy consumption of resources and the behavior of the jobs submitted [7], [13]. Garg and Buyya [7] propose Heterogeneity Aware Meta-Scheduling Algorithm (HUMA) to schedule real-time workloads on resource sites which are more power-efficient. HUMA uses global information about the grid resources to allocate jobs to more energy-efficient resources, and focus on high-performance computing (HPC) applications. Sharma and Aggarwal [23] present a resource prediction tool to predict the job resource usage and an energy-aware scheduler for memory-intensive applications. These studies indicate significant energy savings and little impact on performance. The main difference when compared to the work present in this paper is on the type of grid considered. We analyze energy-efficiency strategies to save energy in opportunist grids. In these grids there is no reservation of resources and resources are not dedicated to the grid.

Kim et al. [13] proposed power-aware scheduling for real-time BoT applications on cluster systems enabled with a Dynamic Voltage Scaling (DVS). The study focuses on a scheduling algorithm that applies DVS to save energy at the same time that guarantees that tasks meet their deadlines. Differently from opportunistic grids, cluster systems provide resource availability guarantees and a homogeneous environment, which are fully exploited to provide energy-efficient of the applications considered. Our focus is on the impact that energy-saving mechanisms may have on the makespan of the applications.

III. Simulation of Energy Management in P2P Opportunistic Grids

Measuring the actual energy consumed by opportunistic grids is challenging, because the computing resources are heterogeneous, dispersed geographically, and used opportunistically. Simulation is a suitable alternative to address these difficulties. We have not found any opportunistic grid simulator that supports simulation of BoT scheduling strategies, sleeping strategies and that measures energy consumption, thus, we have developed our own event-based simulator. Our simulator offers the possibility of reproducing BoT scheduling strategies and sleeping strategies. The simulator design is based on the OurGrid P2P grid [6]. It models the three main components of an OurGrid system, namely: broker, peer and worker.

A P2P grid represents a federation of resources and users that belong to different administrative domains, or sites. Each site manages a number of machines that it offers to the P2P grid. This enables it to give to its local users access not only to its local machines, but also to the machines offered by the other sites. Each site is managed by a *Peer*.

The *Worker* is the component that executes on the grid resources. Workers monitor the utilization of resources and detect when they are idle. Upon detecting that a resource is idle, the worker passes this information on to the peer to which it is connected. At this point, a worker can also decide to switch to an energy-saving mode (sleeping state). A worker that is running a task of a grid job can be claimed back by its owner at any time. When this occurs, we assume that the computation is checkpointed and restarted later at another worker that is available to the grid. In practice, BoT applications are not checkpointed. We decided to assume checkpoints in order to speed up our simulations. Note, however, that checkpoints do not alter the relative performance of the mechanisms studied.

The *Broker* component provides an interface that users use to submit their BoT applications to the P2P grid. It is also responsible for scheduling applications on the grid machines. When a grid user submits a job, the broker that runs at its site requests workers from the local peer. The local peer returns to the broker the local workers that are available, and, if these are not enough to fulfil the broker's request, contact the other peers to find out workers that are available at remote sites and that can be used to fully or partially fulfil the request. When the workers are received, the broker schedules the tasks of the job to execute in these workers, following a simple workqueue approach [24].

For simplicity, we did not model the prioritization mechanism used by the OurGrid peers to decide how to allocate their resources when there is contention for them, i.e. when they are simultaneously requested by more than one peer. This mechanism is crucial to create an incentive for the donation of resources [2], but its functioning affects the several mechanisms studied in a similar way, thus, this simplification should not change our findings substantially. As a consequence, we only consider a single peer and all machines in the grid belong to this peer.

The simulator receives as input two traces and a grid description file. The first trace provides the workload considered and consists of a number of entries, each one providing: job identification, job submission time, task identification, and task computing demand. The same job identification can be associated to one or more tasks that comprise a BoT submission, and each task has a computing demand associated to it. This is expressed as the amount of time that is required to execute the task in the fastest machine of the grid. The second trace describes the availability of all the machines used in the simulation for the whole simulation period. Finally, the system configuration is described by a file that contains one line per machine in the grid, and each line has six fields: machine identification, set of supported CPU speed[4], power consumption associated to each supported CPU speed, set of state transitions supported, latency associated to each state transition supported, and power consumption associated to each state transition supported.

Every machine connected to the grid can be in one of following five states: owner, running, idle, standby or hibernate (see Figure 1). A machine is in the *owner* state when it is in use by its owner. In the owner state the machine is not available to the grid. In this work, we did not measure the energy consumed while the machine is in owner state. The *running* state indicates that the machine is executing a task for a user of the P2P grid. A machine is in the *idle* state when the owner is no longer using the machine, so it is available to be used by the grid, but there are no grid tasks currently allocated to it. In the idle state the machine is active and its power consumption is given by P_i. The latency incurred to change the machine from idle to any other state is negligible and is considered to be zero ($L_i = 0$). Thus, in the idle state, a machine can begin to excute a task as soon as it arrives. *Standby* and *hibernate* are states equivalent to the idle state. The difference is that these sleeping states save energy, when compared to the energy consumption of a machine in the idle state. On the other hand, the latencies associated with the transitions from any of these states to any other state are not negligible.

The power consumptions and latencies associated with the standby and hibernate states are given by, respectively: P_s, L_s, P_h, and L_h. Moreover, $P_h < P_s < P_i$, and $L_h > L_s > L_i$. It is important to notice that state transitions should not be performed too frequently, otherwise the energy

[4]Modern CPUs can operate on different speeds, each with a particular power consumption associated. The current operational speed can be changed at will by the operating system.

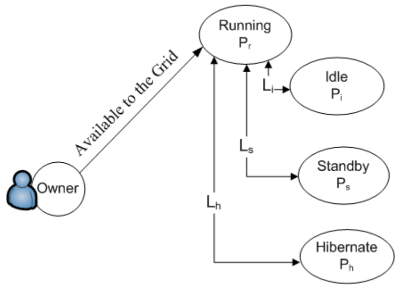


Figure 1. Possible states of a grid machine with a grid woker agent installed. L indicates the latency required to reach and leave the state, and P indicates the power consumption when it operates in this state.

consumption may be even higher than the one in the scenario where the machines are never set to sleep. In this paper we assume that this situation never arises. A state transition from sleeping to operational triggered at a given time t is only immediately made effective if the energy saved since the last time the machine was set to sleep is larger than the energy required to bring the machine back in operation. If not, the transition is postponed to a time $t', t' > t$, at which this is satisfied.

As discussed before, we consider two metrics in our evaluation: makespan and energy consumption. The makespan of a BoT job J_i is defined as the difference between the earliest time of submission (s_i) of any of its tasks, and the latest time of completion of any of its tasks (c_i), more formally $m_{J_i} = c_i - s_i$ [11]. To measure the impact on the makespan when comparing the different strategies expressed in two simulation scenarios, we use the slowdown metric, defined for a job J_i as: $s_{J_i} = m^A_{J_i}/m^B_{J_i}$, where $m^A_{J_i}$ is the makespan for a scenario A, while $m^B_{J_i}$ is the makespan for a scenario B; both scenarios are fed with the same traces and system configuration, varying only on the scheduling strategies used. Slowdowns smaller than 1 indicate that the strategies used in scenario A offer a speedup when compared to those used in scenario B.

The energy consumed in Δt units of time is computed as $E = P * \Delta t$, where P is the power consumption in Watts (W) considered for the machine in the period of time Δt. As discussed above, P varies with the state in which the machine is operating (running, idle, standby and hibernate). In our experiments, we compare several energy saving strategies. Again, we measured the energy savings of the strategies in a scenario A by comparing the energy consumption of this scenario with the energy consumption of a similar scenario where the only difference is that machines are never placed in a sleeping state, i.e. they can only be in the owner, running and idle states. This is expressed as: $\xi_A = (E_A - E_{A,idle})/E_{A,idle}$. Note that both scenarios receive the same traces and system configuration as input.

The dynamics of the event-based simulator is as follows. When a new job demand arrives, if required, the peer follows a wake-up strategy to decide which machines should be brought back to an operational state. There are two wake-up machines strategies implemented: Most Recently Sleeping (MRS) and Energy Aware (EA). MRS wakes up first the machines that were more recently sent to sleep. EA sorts the machines by energy efficiency (speed divided by power consumption) and wakes up first the machines that are more efficient. When the broker receives the machines (workers), it uses a scheduling strategy to allocate the tasks to the workers. We have implemented the following scheduling strategies. First Come First Served (FCFS), which allocates the first task available to the first processor available; Fastest Processor to Largest Task (FPLT), which orders the available tasks by their computing demand and allocate them to the next fastest processor available[5], Most Energy-Efficient First (MEEF), which works similarly to FCFS, but considers the machines ordered by energy-efficiency; and, Most Energy-Efficient to Largest Task (MEELT), which works similarly to FPLT, but considers the machines ordered by energy-efficiency, rather than processing speed. As the workload is processed, the simulator calculates the makespan of each job and the energy consumption of each machine of the P2P grid.

IV. The Experimental Setup

We simulated grids with up to 360 machines. This was enough to allow the simulation of scenarios that span from high to low contention. The machines have the characteristics of the machines currently available in two sites of the OurGrid Community grid (http://status.ourgrid.org/), namely: lsd.ufcg.edu.br and gmf.ufcg.edu.br. These are ordinary machines used in University research labs. Figure 3 and Figure 2 show, respectively, the frequency distribution of CPU speed and power consumption in idle state of the grid machines. We considered desktops with AMD [1] and Intel processors [9]. These manufacturers offer the thermal design power (TDP) of processor models. TDP is the maximum power that a processor can draw for a thermally significant period while running commercially useful software. The power consumption of the machines was attributed based on their model and the power consumption specification provided by their manufactures. We assumed that all jobs submitted to the grid are CPU-bound, which facilitated the calculation of the energy consumption; other studies have also made a similar simplification [15]. To select machines for scenarios with less than 360 machines we did as follows. All the machines were randomly ordered in a list. For a scenario with x machines we have used the first x machines in this list.

For simplicity, we considered that the power consumption and latencies associated with the standby and hibernate states are the same for all machines. In the hibernate state the

[5]Note that a single task is allocated to each processor at a time.

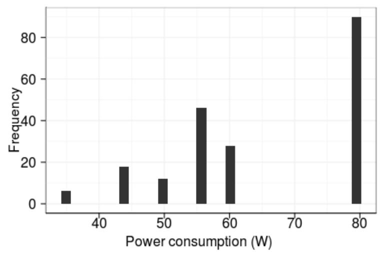


Figure 2. Power consumption of machines in the idle state.

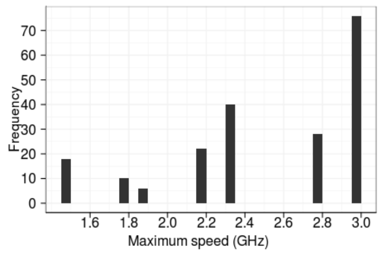


Figure 3. Maximum speed in GHz.

machines consume the power required to enable the wake-on-LAN device, which is typically $P_h = 0.7$ W [9]. In standby the power consumption is approximately $P_s = 3$ W [9]. We consider that the transition of any state to standby takes 2.5 seconds ($L_s = 2.5$), and 55 seconds for the transition from any state to hibernate ($L_h = 55$) [17], [18]. The latency for the transitions to/from the hibernate state can be influenced by some settings or operating system dependencies, however, in this study we considered only the time required to reactivate the system state stored on the hard disk.

To represent the variation in the availability of the machines over time, we used a trace of a desktop grid that displays data on the use of machines, identifying the periods when the machines are idle[6]. The trace used was collected on a non-continuous period of one month, with information about desktop machines belonging to the University of Paris-Sud [14]. As the availability information present in this trace does not form a continuous period, we completed it to generate a complete trace. This was done by identifying periods with information that covered the same times and days of the week of the missing periods and copying this information. Also, for the simulations that lasted for more than one month, the availability information is obtained by using the trace in a circular way.

We have used a trace obtained from the OurGrid Community to simulate the demand for jobs in the grid. This trace contains information about the time of submission and execution time of the tasks of each job. The trace contains

[6]In all traces used in this work, information regarding time is expressed in seconds.

information about $21,705$ tasks executed between December 18, 2008 and November 18, 2009. In the trace there are tasks with different execution times, however, there are some tasks with execution times that are too large, which are beyond the typical execution time of tasks amenable for execution in opportunistic grids. To better analyze these tasks, we calculated the 99-percentile and found that tasks with execution time longer than $41,367$ seconds (27 tasks) were infrequent. These were, therefore, eliminated from the trace used. The filtered trace was then divided in 10 segments of roughly $2,000$ tasks, each, and used as input to the different scenarios evaluated.

The availability of real workloads of BoT applications is rather limited. Thus, we decided to also use synthetic workloads to complement our investigation. To generate the different workloads considered, we used the workload model proposed by Iosup et al. [11]. This model defines distributions for: BoT inter-arrival time (Weibull distribution with $\alpha = 4.25$ and $\beta = 7.86$), BoT Size, i.e, the number of tasks in each job (Weibull distribution with $\alpha = 1.76$ and $\beta = 2.11$), and average task runtime and runtime variability for tasks belonging to the same job (normal distribution with $\mu = 2.73$ and $\sigma = 6.01$, and Weibull distribution with $\alpha = 2.05$ and $\beta = 12.25$, respectively). We generated 10 samples of 200 BoT following these distributions.

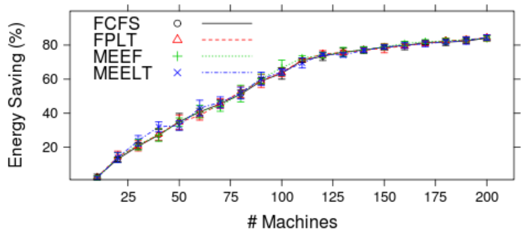
The simulator and all the workload traces used in this work are available for download at: http://redmine.lsd.ufcg.edu.br/wiki/green-grid.

V. Simulation Results

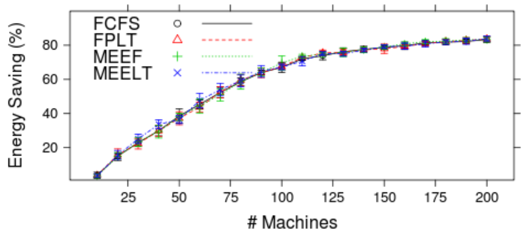
We have conducted simulation experiments considering scenarios that allowed us to investigate the impact of different: i) sleeping strategies; ii) BoT scheduling strategies; and, iii) wake-up strategies. In all cases, we consider different resource contention levels. In all scenarios, we measure the average makespan of the jobs and the average energy consumption of the whole grid, considering a set of different workloads. These values are used to calculate the energy saving achieved and the slowdown associated to the several scenarios simulated.

Our first experiment focused on verifying the impact of policies for scheduling tasks. Figure 4 presents the energy savings achieved with the OurGrid workload. We note that there are no significant differences between the scheduling strategies evaluated, regarding the energy saving aspect. All strategies provide significant energy savings and, as expected, the savings increase as the resource contention diminishes.

Figure 5 shows the slowdown of the jobs for this experiment. The scheduling strategies evaluated have similar impact on the slowdown of jobs. However, the slowdown of jobs varies significantly depending on the sleeping strategy used. Hibernate generates greater slowdown (a maximal value of approximately 1.7) when the amount of resources



(a) Hibernate



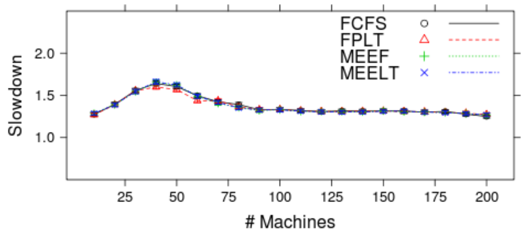
(b) Standby

Figure 4. Energy savings generated by the task scheduling policies with the OurGrid workload. Simulations using the EA wake-up strategy. Relative error bars for a confidence level of 95%.

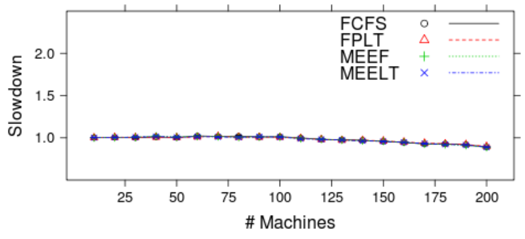
provisioned is between 20 and 90. When the provision of resources is larger than 90, the slowdown reduces and tends to 1. Also when there are only 10 resources, the slowdown is close to 1. This is because in this setting, transitions to the hibernate state are rare, since there are always jobs in the queue. In the intermediate contention levels the strategy is used and the jobs must wait a resource to be awaken, before they can be scheduled to run on them by the broker. On the other hand, when the provision of resources increases, the benefit generated by the increase in the amount of resources available to complete tasks outweighs the cost of waiting for some machines to be awaken, causing a reduction in the slowdown. The same behaviour was also observed when we used the synthetic workload.

Standby presented low slowdown in all scenarios evaluated. Moreover, Figure 5(b) shows that there is a speedup in the scenarios where the amount of resources provisioned is large. This happens only when standby is combined with EA. This is due to the fact that, for the data we used, there is a positive correlation between the energy efficiency and the speed of the processors. The results for MRS are similar, therefore, we do not show them here.

Since both wake-up strategies provided low slowdowns, both when using standby and hibernate as sleeping states, we checked if there was any significant difference between EA and MRS, when considering energy savings. Figure 6 shows that EA performs better than MRS when the number of resources provisioned is larger than 100. When this number is less than 100, both strategies perform equally

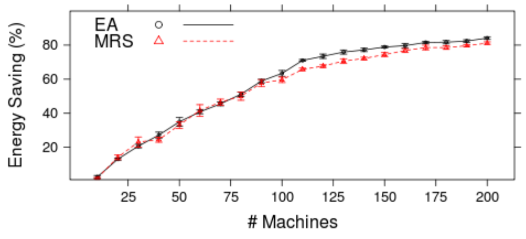


(a) Hibernate

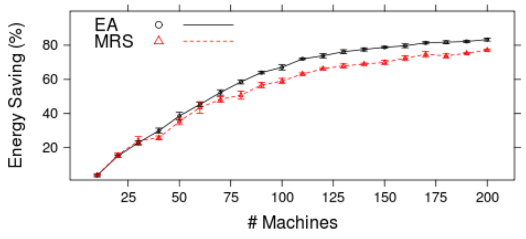


(b) Standby

Figure 5. Slowdown generated by the task scheduling policies with the OurGrid workload. Simulations using the EA wake-up strategy. Relative error bars for a confidence level of 95%.



(a) Hibernate



(b) Standby

Figure 6. Energy savings for both EA and MRS for the two sleeping modes with the OurGrid workload. Relative error bars for a confidence level of 95%.

well. This is because BoT applications submit several tasks simultaneously. Otherwise, when the number of resources provisioned is larger than the number of tasks submitted, the strategy to wake-up becomes relevant. In this context, wake up the most energy-efficient machines first results in greater energy savings.

Figure 7 shows that, when the amount of resources in the grid was low, both strategies were similar in terms of energy consumption. On the other hand, when the amount of available resources is larger than 200, hibernate provides greater energy savings than standby. Hibernate provides greater energy saving than standby when the resource stays idle for a long time. This is because a resource consumes less power in hibernate than in standby.

VI. Concluding Remarks

In this work we have investigated three aspects of an opportunistic grid according to two metrics: energy savings and jobs' makespan. We have evaluated sleeping, wake-up, and scheduling strategies. Sleeping strategies are employed to reduce the energy consumption of the grid during idleness periods; wake-up strategies are employed to choose a set of resources in order to fulfill a workload demand; and scheduling strategies are employed to decide which tasks to schedule to the available machines.

We evaluated two sleeping strategies, denoted: standby and hibernate. Our results have shown that they can both provide substantial savings in the energy consumption of opportunistic grids. In the scenarios we simulated, the

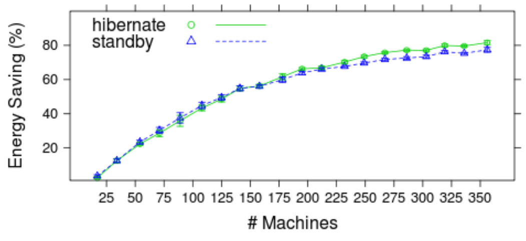


Figure 7. Energy savings for both Hibernate and Standby with the synthetic workload. Relative error bars for a confidence level of 95%.

savings can reach more than 80%, when compared to the energy consumed when the resources are not set to sleep. More importantly, these savings are attained without a huge penalty associated with the increase of the jobs' makespan. We have also evaluated two wake-up strategies: EA and MRS; which have presented similar results in terms of jobs' makespan. However, in scenarios of low contention for resources, EA showed better energy savings results than MRS. Finally, we evaluated four task scheduling policies: FCFS, FPLT, MEEF, and MEELT. These strategies take into account different characteristics of the resources, as well as the workload submitted to the grid. They have not shown significant difference regarding energy savings and makespan when compared with each other.

In summary, we have analyzed different strategies considering several aspects of the grid, and our results presented a significant reduction in energy consumption, surpassing

80% in a scenario when the contention for resources in the grid was low. Moreover, this comes with limited impact on the response time of the applications. As future work, we intend to investigate the performance of these energy-saving strategies in a grid model which considers the existence of multiple peers. Furthermore, we intend to investigate more sophisticated task scheduling policies and other operating features of the grid, such as workload and resources characteristics.

Acknowledgment

Francisco Brasileiro would like to thank the financial support from CNPq/Brazil (grant 300.646/96).

References

[1] Advanced Micro Devices (AMD), "support.amd.com/us/Processor_TechDocs/43375.pdf," Online July 2010.

[2] N. Andrade, F. Brasileiro, W. Cirne, and M. Mowbray, "Automatic grid assembly by promoting collaboration in peer-to-peer grids," *J. Parallel Distrib. Comput.*, vol. 67, no. 8, pp. 957–966, 2007.

[3] J. Augustine, S. Irani, and C. Swamy, "Optimal power-down strategies," in *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science*. Washington, DC, USA: IEEE Computer Society, 2004, pp. 530–539.

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

[5] Berkeley Open Infrastructure for Network Computing, "http://boinc.berkeley.edu/," online July 2010.

[6] W. Cirne, F. Brasileiro, N. Andrade, L. Costa, A. Andrade, R. Novaes, and M. Mowbray, "Labs of the world, unite!!!" *Journal of Grid Computing*, vol. 4, no. 3, pp. 225–246, 2006.

[7] S. K. Garg and R. Buyya, "Exploiting heterogeneity in grid computing for energy-efficient resource allocation," 2009.

[8] Hewlett-Packard Corporation, Intel Corporation, Microsoft Corporation, Phoenix Technologies Ltd. and Toshiba Corporation, "Advanced configuration and power interface specification," http://www.acpi.info/spec.htm Online July 2010.

[9] Intel and U.S. Environmental Protection Agency, "Energy star* system implementation whitepaper," www.intel.com/cd/channel/reseller/asmo-na/eng/339085.htm Online July 2010.

[10] A. Iosup, C. Dumitrescu, D. Epema, H. Li, and L. Wolters, "How are real grids used? the analysis of four grid traces and its implications," in *Proceedings of the 7th IEEE/ACM International Conference on Grid Computing*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 262–269.

[11] A. Iosup, O. Sonmez, S. Anoep, and D. Epema, "The performance of bags-of-tasks in large-scale distributed systems," in *Proceedings of the 17th international symposium on High performance distributed computing*. New York, NY, USA: ACM, 2008, pp. 97–108.

[12] S. Irani, S. Shukla, and R. Gupta, "Online strategies for dynamic power management in systems with multiple power-saving states," *ACM Trans. Embed. Comput. Syst.*, vol. 2, no. 3, pp. 325–346, 2003.

[13] K. H. Kim, R. Buyya, and J. Kim, "Power aware scheduling of bag-of-tasks applications with deadline constraints on dvs-enabled clusters," *IEEE International Symposium on Cluster Computing and the Grid*, pp. 541–548, 2007.

[14] D. Kondo, M. Taufer, C. L. B. III, H. Casanova, Henri, C. Andrew, and A. A. Chien, "Characterizing and evaluating desktop grids: An empirical study," in *In Proceedings of the International Parallel and Distributed Processing Symposium*, 2004.

[15] M. Lammie, D. Thain, and P. Brenner, "Scheduling Grid Workloads on Multicore Clusters to Minimize Energy and Maximize Performance," in *10th IEEE/ACM International Conference on IEEE Grid Computing*, 2009, pp. 145–152.

[16] Microsoft Corporation, "Windows power management," http://www.microsoft.com/whdc/archive/winpowmgmt.mspx Online July 2010.

[17] A. C. Orgerie, L. Lefevre, and J. P. Gelas, "Save watts in your grid: Green strategies for energy-aware framework in large scale distributed systems," in *International Conference on Parallel and Distributed Systems*. Los Alamitos, CA, USA: IEEE Computer Society, 2008, pp. 171–178.

[18] A.-C. Orgerie, L. Lefevre, and J.-P. Gelas, "Chasing gaps between bursts: Towards energy efficient large scale experimental grids," in *International Conference on Parallel and Distributed Computing Applications and Technologies*. Los Alamitos, CA, USA: IEEE Computer Society, 2008, pp. 381–389.

[19] V. Pallipadi and A. Starikovskiy, "The ondemand governor: past, present and future," in *Proceedings of Linux Symposium, vol. 2, pp. 223-238*, 2006.

[20] E. Pinheiro, R. Bianchini, E. V. Carrera, and T. Heath, *Dynamic cluster reconfiguration for power and performance*. Norwell, MA, USA: Kluwer Academic Publishers, 2003, pp. 75–93.

[21] J. Reich, A. Kansal, M. Gorackzo, and J. Padhye, "Sleepless in seattle no longer," in *USENIX Annual Technical Conference*, 2010.

[22] SETI Institute, "Search for extraterrestrial intelligence," http://setiathome.ssl.berkeley.edu/ Online July 2010.

[23] K. Sharma and S. Aggarwal, "Energy aware scheduling on desktop grid environment with static performance prediction," in *Proceedings of the 2009 Spring Simulation Multiconference*. San Diego, CA, USA: Society for Computer Simulation International, 2009, pp. 1–8.

[24] D. P. D. Silva, W. Cirne, F. V. Brasileiro, and C. Grande, "Trading cycles for information: Using replication to schedule bag-of-tasks applications on computational grids," in *Applications on Computational Grids, in Proc of Euro-Par*, 2003, pp. 169–180.