Simulation-Based Performance Evaluation of an Energy-Aware Heuristic for the Scheduling of HPC Applications in Large-Scale Distributed Systems

Georgios L. Stavrinides
Department of Informatics
Aristotle University of Thessaloniki
54124 Thessaloniki, Greece
gstavrin@csd.auth.gr

Helen D. Karatza
Department of Informatics
Aristotle University of Thessaloniki
54124 Thessaloniki, Greece
karatza@csd.auth.gr

ABSTRACT
As the distributed resources required for the processing of High Performance Computing (HPC) applications are becoming larger in scale and computational capacity, their energy consumption has become a major concern. Therefore, there is a growing focus from both the academia and the industry on the minimization of the carbon footprint of the computational resources, especially through the efficient scheduling of the workload. In this paper, a technique is proposed for the energy-aware scheduling of bag-of-tasks applications with time constraints in a large-scale heterogeneous distributed system. Its performance is evaluated by simulation and compared with a baseline algorithm. The simulation results show that the proposed heuristic not only reduces the energy consumption of the system, but also improves its performance.

Keywords
Energy-aware scheduling; bag-of-tasks applications; time constraints; large-scale distributed systems; simulation; performance evaluation

1. INTRODUCTION
The inherent and ever-growing need of High Performance Computing (HPC) applications for effective, highly parallel and scalable processing, requires the employment of distributed computing resources that are becoming larger in scale and computational capacity day-by-day. The energy consumption of these platforms has become a major concern. Consequently, there is a growing focus from both the research community and the industry on the minimization of the carbon footprint of the computational resources, especially through the efficient scheduling of the workload.

HPC applications are usually bags-of-tasks (BoT). That is, they consist of independent, embarrassingly parallel tasks that can be executed in any order, without any intertask communication. Examples of such applications include chromosome mapping, pattern matching and data mining applications [14, 15]. Furthermore, such applications often have time constraints (deadlines) within which they must finish execution. A late result in such cases is commonly considered useless [17, 18, 19, 16].

1.1 Motivation
The nature and time constraints of the workload, as well as the heterogeneity and energy consumption of the distributed computing resources, constitute major challenges that must be addressed during the scheduling of BoT applications. Effective techniques must be employed, in order to minimize the energy consumption, while maintaining good performance. The performance of such algorithms is usually evaluated via simulation, rather than by analytical methods. Analytical modeling is difficult and often requires simplifying assumptions that may have an unpredictable impact on the results.

1.2 Contribution
In this paper, we propose a technique for the energy-aware scheduling of bag-of-tasks applications with time constraints in a large-scale heterogeneous distributed system. Its performance is evaluated by simulation and compared with a baseline algorithm. The rest of the paper is organized as follows: Section 2 presents an overview of related literature, while Section 3 describes the system, workload and energy consumption models, as well as the proposed scheduling technique. Section 4 presents the performance metrics, the experimental setup and the simulation input parameters. Furthermore, it presents and analyzes the simulation results. Finally, Section 5 concludes the paper, providing directions for further research.

2. RELATED WORK
The scheduling of bag-of-tasks applications has been studied extensively in the literature [9, 5, 23]. Two of the most widely used scheduling heuristics are the MinMin and MaxMin policies. They were first introduced in [7] and later implemented in a widely distributed environment in [6]. The MinMin strategy is a two-step iterative process. In the first step, the minimum completion time (MCT) of each unscheduled task is calculated, over all of the processors in the system. In the second step, the task with the minimum MCT is assigned to the corresponding processor. At each
iteration, the current load of the processors is taken into account for the next scheduling decision [25].

The difference between the MaxMin and MinMin policies, is that in the second step of the scheduling process, in the case of MaxMin, the task with the maximum, instead of the minimum, MCT is assigned to the corresponding processor. Consequently, in cases where the application consists of a large number of tasks with small execution times and a few tasks with large execution times (e.g. cases where the computational demands of the tasks are exponentially distributed), the MaxMin heuristic is likely to give a smaller total execution time than the MinMin policy, since it schedules the tasks with larger execution times at earlier iterations [21]. Therefore, there is a higher probability that the tasks with smaller execution times that are subsequently executed, will be executed in parallel with the tasks with larger execution times. Alternative versions of the above algorithms have been proposed in the literature, in an attempt to make them energy-aware [12]. However, the vast majority of them trades off performance for energy savings [10]. The goal of our proposed scheduling heuristic is to achieve energy savings, without compromising the system performance, as this is crucial in the case of HPC applications.

3. PROBLEM FORMULATION

3.1 System Model

The distributed system under study consists of a set $Q = \{p_1, p_2, ..., p_q\}$ of $q$ fully connected heterogeneous processors. Each processor $p_i$ serves its own local queue of tasks (i.e. it has its own memory) and has an execution rate $\mu_i$. A dedicated processor acts as a central scheduler, responsible for scheduling the component tasks of the application jobs to the other processors [26]. The queueing model of the target system is shown in Figure 1.

3.2 Workload Model

In our study, the jobs generated by HPC applications arrive at the central scheduler in a Poisson process with rate $\lambda$ [8, 22, 20]. It is assumed that each job is a bag-of-tasks (BoT), consisting of a set $J = \{t_1, t_2, ..., t_n\}$ of $n$ independent parallel tasks, without any communication or precedence constraints among them. It is assumed that any input data required by a task to start execution are available on all processors. Furthermore, it is assumed that the component tasks of the jobs are not preemptible, as it has been shown that preemption of time-constrained tasks may eventually lead to performance degradation [4]. Each component task $t_i$ has a weight $w_i$, which denotes its computational volume, i.e. the amount of computational operations that must be executed. The computational volume of the tasks is exponentially distributed with mean $\overline{\tau}$.

The computational cost $\text{Comp}$ of a task $t_i$ on a processor $p_j$ is defined as:

$$\text{Comp}(t_i, p_j) = \frac{w_i}{\mu_j}$$

where $\mu_j$ is the execution rate of processor $p_j$. The length $L$ of a job is equal to the largest average computational cost of its component tasks. That is:

$$L = \max_{t_i \in J} \left\{ \text{Comp} (t_i) \right\}$$

where $\text{Comp}(t_i)$ is the average computational cost of task $t_i$ over all of the processors in the system.

It is assumed that each job has an end-to-end deadline $D$, which is given by:

$$D = AT + RD$$

where $AT$ is the arrival time of the job. $RD$ is the relative deadline of the job and is uniformly distributed in the range $[2L, 4L]$. When a job fails to complete its execution within its deadline, it is considered lost. An example of a BoT job is illustrated in Figure 2.

3.3 Energy Consumption Model

In a large-scale distributed system, typically the processors consume the greatest amount of energy, compared to the other system resources [24]. Consequently, in this paper we focus on the energy consumption of the processors. Specifically, it is assumed that the power $P^i$ consumed by a processor $p_i$ in the system, is given by:

$$P^i(u_i(\tau)) = P^i_{\text{idle}} + \left( P^i_{\text{max}} - P^i_{\text{idle}} \right) \cdot u_i(\tau)$$

where $P^i_{\text{idle}}$ is the power consumed by the processor when idle, whereas $P^i_{\text{max}}$ is the power consumed by the processor at 100% utilization. $u_i(\tau)$ is the utilization of the processor and it is a function of time $[3, 2, 13]$. Therefore, the energy consumption $E^i$ of a processor $p_i$ over a period of time $[\tau_0, \tau_1]$
can be defined as an integral of its power consumption $P^i$:

$$E^i = \int_{t_0}^{t_1} P^i (u_i(\tau)) \, d\tau$$ (5)

### 3.4 Scheduling Algorithms

#### 3.4.1 Baseline Policy

As a baseline algorithm, the MaxMin policy is employed, since it provides better results for jobs with tasks that have exponential computational demands, as described in Section 2. We adapted this policy, in order to take into account the deadlines of the jobs during the scheduling process. Specifically, when a job arrives at the central scheduler, it enters a global waiting queue and the scheduler is invoked. In the first step of the first iteration of the scheduling process, the estimated completion time $CT(t_i,p_j)$ of each component task $t_i$ of the job is calculated on each processor $p_j$ in the system:

$$CT(t_i,p_j) = \tau_{idle}(t_i,p_j) + \text{Comp}(t_i,p_j)$$ (6)

where $\tau_{idle}(t_i,p_j)$ is the time at which processor $p_j$ will be able to execute task $t_i$.

For the calculation of the $\tau_{idle}(t_i,p_j)$ parameter, we first find the potential position of task $t_i$ in $p_j$’s queue, according to the deadline $D$ of $t_i$’s job, so that the task with the earliest deadline is placed at the head of $p_j$’s queue. Consequently, the local waiting queues of the processors are arranged according to the Earliest Deadline First (EDF) policy [11]. Subsequently, the minimum estimated completion time $MCT$ of each task $t_i$ over all of the processors in the system is calculated, as follows:

$$MCT(t_i) = \min_{p_j \in Q} \{CT(t_i,p_j)\}$$ (7)

In the second step of the first iteration of the scheduling process, we select the task with the maximum $MCT$ and we assign it to the processor that can provide it with it.

Subsequently, a second iteration of the scheduling process is performed, taking into account the current load of the processors, as resulted by the scheduling decision of the previous iteration. The iterations continue, until all of the component tasks of the newly-arrived job are scheduled. The pseudocode for the baseline MaxMin policy is shown in Algorithm 1.

#### 3.4.2 Energy-Aware Scheduling Heuristic

Our proposed energy-aware scheduling heuristic is based on the MaxMin policy, which was modified so that the energy consumption of the processors is taken into account during the scheduling process. Specifically, during the first step of the process, along with the estimated completion time $CT$ of each component task $t_i$ on each processor $p_j$, the estimated energy consumption $E^j$ of each processor is also calculated, as follows:

$$E^j = P_{max}^j \cdot (CT(t_i,p_j) - \tau)$$ (8)

where $P_{max}^j$ is the power consumption of processor $p_j$ at 100% utilization, whereas $\tau$ is the current time. That is, we calculate the estimated amount of energy required to finish the execution of all of $p_j$’s assigned tasks, including task $t_i$. Subsequently, after calculating the minimum estimated completion time $MCT$ of task $t_i$ as in the case of MaxMin, we find the set of processors that can provide the task with estimated completion time that falls in the range $[MCT(t_i), MCT(t_i) \cdot (1 + SM)]$, where $SM$ is the selection margin of the algorithm. From this set, we choose the processor $p_j$ that has the minimum estimated energy consumption. The minimum estimated completion time of the task is now $MCT'(t_i) = CT(t_i,p_j)$.

The second step of the iteration is performed in the same manner as in the case of MaxMin. That is, we choose the task with the maximum $MCT'$ and we assign it to the processor that can provide it with it. The pseudocode for the proposed ENRG-MaxMin policy is shown in Algorithm 2.

---

**Algorithm 1** The baseline MaxMin policy.

**Input:** A bag-of-tasks job $J = \{t_1, t_2, ..., t_n\}$ of $n$ independent parallel tasks and a set $Q = \{p_1, p_2, ..., p_q\}$ of $q$ heterogeneous processors.

**Output:** A schedule of $J$ onto $Q$.

```
1: repeat
2: /* 1st Step */
3: for each task $t_i \in J$ do
4: for each processor $p_j \in Q$ do
5: $\text{CT}(t_i,p_j) \leftarrow \tau_{idle}(t_i,p_j) + \text{Comp}(t_i,p_j)$$
6: end for
7: $\text{MCT}(t_i) \leftarrow \min_{p_j \in Q} \{\text{CT}(t_i,p_j)\}$
8: end for
9: /* 2nd Step */
10: select $t_i \in J$ so that
11: $\text{MCT}(t_i) = \max_{t_i \in J} \{\text{MCT}(t_i)\}$
12: $\text{assign } t_i \text{ to } p_j \text{ where } \text{MCT}'(t_i) = \text{CT}(t_i,p_j)$
13: mark $t_i$ as scheduled and remove it from $J$
14: until all tasks $t_i \in J$ are scheduled
```

**Algorithm 2** The proposed ENRG-MaxMin policy.

**Input:** A bag-of-tasks job $J = \{t_1, t_2, ..., t_n\}$ of $n$ independent parallel tasks and a set $Q = \{p_1, p_2, ..., p_q\}$ of $q$ heterogeneous processors.

**Output:** A schedule of $J$ onto $Q$.

```
1: repeat
2: /* 1st Step */
3: for each task $t_i \in J$ do
4: for each processor $p_j \in Q$ do
5: $\text{CT}(t_i,p_j) \leftarrow \tau_{idle}(t_i,p_j) + \text{Comp}(t_i,p_j)$$
6: $E^j \leftarrow P_{max}^j \cdot (\text{CT}(t_i,p_j) - \tau)$
7: end for
8: $\text{MCT}(t_i) \leftarrow \min_{p_j \in Q} \{\text{CT}(t_i,p_j)\}$
9: for each processor $p_j \in Q$ do
10: if $\text{CT}(t_i,p_j) \in [\text{MCT}(t_i), \text{MCT}(t_i) \cdot (1 + SM)]$
then
11: insert $p_j \in Q'\$
12: end if
13: end for
14: select $p_j \in Q'$ so that $E^j = \min_{p_j \in Q'} \{E^n\}$
15: $\text{MCT}'(t_i) \leftarrow \text{CT}(t_i,p_j)$
16: end for
17: /* 2nd Step */
18: select $t_i \in J$ so that
19: $\text{MCT}'(t_i) = \max_{t_i \in J} \{\text{MCT}'(t_i)\}$
20: $\text{assign } t_i \text{ to } p_j \text{ where } \text{MCT}'(t_i) = \text{CT}(t_i,p_j)$
21: mark $t_i$ as scheduled and remove it from $J$
22: until all tasks $t_i \in J$ are scheduled
```
4. SIMULATION-BASED PERFORMANCE EVALUATION

4.1 Performance Metrics

In order to compare the performance of the proposed energy-aware scheduling heuristic with the baseline algorithm, the following metrics were used:

- **Job Guarantee Ratio**, which is the ratio of the number of jobs that completed their execution within their deadline, over the number of all of the jobs that arrived at the central scheduler, during the observed time period.

- **Total Energy Consumption**, which is the total energy consumed by all of the processors in the system, during the observed time period. It is measured in kilowatt-hours (kWh).

4.2 Simulation Setup

In order to evaluate the performance of the proposed scheduling heuristic in the framework under study, we implemented our own discrete-event simulator in C++, tailored to the specific requirements of the particular problem. Furthermore, no specific workload traces were used, but synthetic workload instead, in order to obtain unbiased, general results, not applicable only to particular workload traces. The simulation input parameters are shown in Table 1. The ENRG-MaxMin strategy was compared to the baseline policy, MaxMin, under different values of the selection margin SM. Specifically, in the case of the ENRG-MaxMin heuristic, the selection margin in the first step of the scheduling process was chosen to be SM = 5%, 15%, 25%, 35% and 45%.

In this study, it is assumed that half of the processors in the system are small, in terms of computational capacity, with identical execution rate and power consumption characteristics. Correspondingly, the other half of the processors are considered to be large. Specifically, the execution rate and the power consumption characteristics of the small and large processors were modeled after the Intel® Xeon® E3-1206Lv5 and Intel® Xeon® E5-4669 v3 processors, respectively, according to the SPECpower_ssj® 2008 benchmark test that ran for 1000 hours (kWh).

The execution rate of a small processor is $\mu_s = 2.6 \cdot 10^7$ SSJ_3OPS (Server-Side Java Operations Per Second), whereas the execution rate of a large processor is $\mu_l = 7.9 \cdot 10^7$ SSJ_3OPS. The power consumption of a small processor is $P_{idle}^s = 13.8$ W when idle and $P_{max}^s = 46.2$ W at 100% utilization. On the other hand, the power consumption of a large processor is $P_{idle}^l = 22.97$ W and $P_{max}^l = 150.14$ W, respectively. The mean computational volume of the component tasks of the jobs was chosen to be $\tau = 10^8$ SSJ_3OPs (Server-Side Java Operations), so that on average, the computational cost of a task on a large processor is approximately equal to 2 minutes. The value of the arrival rate $\lambda$ of the jobs was chosen to be $\lambda = 0.02$, so that the system is stable.

We ran 30 replications of the simulation with different seeds of random numbers, for each set of input parameters.

4.3 Simulation Results

The simulation results of the comparison of the proposed scheduling heuristic, ENRG-MaxMin, with the baseline policy, MaxMin, are shown in Figures 3 and 4. It can be observed that the proposed technique outperforms the baseline algorithm, in terms of both performance metrics, Job Guarantee Ratio and Total Energy Consumption. Furthermore, it can be observed that the ENRG-MaxMin policy yields the highest guarantee ratio and the lowest energy consumption, when the selection margin SM used in its first scheduling step is set to 25%. Specifically, as it is shown in Table 2, the increase in guarantee ratio is 0.1671%, whereas the decrease in energy consumption is 0.2084%, compared to the baseline strategy. Even though the improvement is relatively small, it can still have a significant impact on the long-term performance and energy consumption of the system.

For higher selection margin values, the performance of the proposed policy starts to deteriorate. This is due to the fact that for higher values, more processors are considered for selection. These processors can provide a higher (i.e. worse) estimated completion time for the task to be scheduled, than the processor that can provide the task with the minimum estimated completion time MCT. From the set of these processors, the algorithm selects the one that requires the minimum estimated amount of energy to finish the execution of all of its assigned tasks, including the task to be scheduled, without taking into account the estimated completion time that it can provide. Consequently, there is a higher probability that a processor with a much higher estimated completion time than the MCT will be selected, ultimately leading to a higher number of jobs missing their deadline. This also results in higher energy consumption, since the energy consumed for the processing of jobs that eventually miss their deadline (i.e. jobs that are eventually lost) is essentially wasted. However, despite the minor degradation in the performance of the ENRG-MaxMin policy for higher selection margin values, it still outperforms the baseline policy, MaxMin.

5. CONCLUSIONS

In this paper, the ENRG-MaxMin heuristic was proposed for the energy-aware scheduling of bag-of-tasks applications with time constraints in a large-scale heterogeneous dis-
Table 1: Simulation model input parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of completed jobs</td>
<td>$10^6$</td>
</tr>
<tr>
<td>Total number of processors</td>
<td>$q = 128$</td>
</tr>
<tr>
<td>Number of small processors</td>
<td>$q_s = 64$</td>
</tr>
<tr>
<td>Number of large processors</td>
<td>$q_l = 64$</td>
</tr>
<tr>
<td>Small processor execution rate</td>
<td>$\mu_s = 2.6 \cdot 10^5$ SSJ_OPS</td>
</tr>
<tr>
<td>Large processor execution rate</td>
<td>$\mu_l = 7.9 \cdot 10^5$ SSJ_OPS</td>
</tr>
<tr>
<td>Number of tasks per job</td>
<td>$n \sim U[1, 64]$</td>
</tr>
<tr>
<td>Job arrival rate</td>
<td>$\lambda = 0.02$</td>
</tr>
<tr>
<td>Job relative deadline</td>
<td>$RD \sim U[2L, 4L]$</td>
</tr>
<tr>
<td>Mean task computational volume</td>
<td>$\overline{w} = 10^8$ SSJ_OPS</td>
</tr>
<tr>
<td>Small processor power consumption when idle</td>
<td>$P_{s_idle} = 13.8$ W</td>
</tr>
<tr>
<td>Small processor power consumption @ 100% utilization</td>
<td>$P_{s_max} = 46.2$ W</td>
</tr>
<tr>
<td>Large processor power consumption when idle</td>
<td>$P_{l_idle} = 23.97$ W</td>
</tr>
<tr>
<td>Large processor power consumption @ 100% utilization</td>
<td>$P_{l_max} = 150.14$ W</td>
</tr>
<tr>
<td>Scheduling method</td>
<td>MaxMin, ENRG-MaxMin</td>
</tr>
<tr>
<td>Selection margin for ENRG-MaxMin</td>
<td>$SM = {5%, 15%, 25%, 35%, 45%}$</td>
</tr>
</tbody>
</table>

Figure 3: Job Guarantee Ratio in the case of ENRG-MaxMin and MaxMin policies. In the case of the ENRG-MaxMin heuristic, the selection margin in the first step of the scheduling process was chosen to be $SM = 5\%, 15\%, 25\%, 35\%$ and $45\%$.

Figure 4: Total Energy Consumption in the case of ENRG-MaxMin and MaxMin policies. In the case of the ENRG-MaxMin heuristic, the selection margin in the first step of the scheduling process was chosen to be $SM = 5\%, 15\%, 25\%, 35\%$ and $45\%$.

6. REFERENCES


