2013 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing

Multi-Objective Workflow Scheduling: An Analysis of the Energy Efficiency and Makespan Tradeoff

Juan J. Durillo and Vlad Nae and Radu Prodan
University of Innsbruck, 6020-Innsbruck, Austria
Email: juan, vlad, radu @dps.uibk.ac.at

***Abstract*—While in the past scheduling algorithms were almost exclusively targeted at optimizing applications' makespan, today they must simultaneously optimise several goals. Among these goals, energy efficiency is receiving increasing attention for environmental and financial reasons. In contrast to related work that optimises energy consumption as a single objective function, we reformulate in this paper the problem as a bi-objective optimisation by considering both makespan and energy as goals. We study the potential benefits of using a Pareto-based workflow scheduling algorithm called MOHEFT using realistic energy consumption and performance models for task executions. We analyse the tradeoff solutions computed by MOHET for different workflows (different in shapes and sizes) in different execution scenarios (different resources in terms of energy consumption). The obtained results show that our bi-objective approach found in some cases schedules that reduce the energy consumption up to 85% with only 3.3% of makespan concessions.**

I. Introduction

Scheduling the activities composing a scientific workflow for optimizing execution time is a challenging task, belonging to the class of NP-complete optimisation problems. In real distributed systems, schedulers must simultaneously optimize other goals too, making the problem even harder. Among these goals, energy efficiency is receiving increasing attention due to two important factors: environmental and financial. For example, [1] reports that about 50% of Amazon's data centre management budget represents costs for powering and cooling the machines.

In these circumstances, workflow scheduling has to be formulated as a multi-objective optimisation problem (MOP) aiming at optimising two possibly conflicting criteria: *makespan* and *energy consumption* entailed by the workflow's execution. The main characteristic of MOPs is that no single solution exists that is optimal with respect to all objectives, but a set of tradeoff solutions known as *Pareto front*. Solutions within this set cannot be further improved in any of the considered objectives without causing the degradation of at least another problem's criterion.

Computing such a set of tradeoff solutions provides the user valuable information about different aspects of workflow execution that cannot be seen otherwise. Concretely, a Pareto front can answer important questions such as: *"are my optimisation goals conflicting?"*, *"can I always simultaneously improve the execution time and the energy efficiency?"* and *"how big is the tradeoff between my optimisation criteria?"*, or it can ease the selection process for the best fitting solution to the user's particular requirements. For example, it may happen that by conceding up to 1% increase in the makespan, the associated energy consumption is halved. This kind of information will be reflected in the Pareto front and is not easily extracted without computing it. While Pareto-based approaches are readily applicable for these situations, most related work formulates the problem as a single objective optimisation to which only a single optimal solution exists, thus depriving the user of this relevant information.

In addition to this, existing energy-efficient approaches to workflow scheduling present two other disadvantages which hinder or even prevent their wider application. Firstly, most of them are based on the dynamic voltage-frequency scaling (DVFS) technology, which enables controlled changes in the clock frequency and voltage of a CPU with the goal of reducing its energy consumption [2], [3], [4]. This technology currently presents an essential limitation preventing its proper employment in the area of scheduling on heterogeneous resources: it has to be applied to the whole CPU[1], thus prohibiting the parallel execution of tasks requiring different DVFS levels on the same multi-core machine. Secondly, most existing models are based on a simplified view of the machine's energy consumption, constrained to two distinct levels corresponding to the machine's idle and fully loaded states. In reality, however, the energy consumption varies in a non-linear fashion with the level of CPU utilisation which, in turn, depends on the executed tasks' characteristics. This behaviour is even more evident in multi-core CPUs with cores having different levels of utilisation.

In this paper, we tackle the problem of energy-efficient workflow scheduling using for the first time a truly multi-objective Pareto-based approach and relying on realistic energy consumption and performance models, able to capture the behaviour of real multi-core CPU systems at different utilisation levels. We model the energy consumption and execution time of workflow tasks using an empirical model

[1]Intel scheduled the release of their newest Haswell architecture in 2013 which promises "more fine grained control" of the CPU sub-components for energy efficiency, but no concrete details have been released so far.

based on knowledge extracted from historical executions of real workflow tasks. For optimising the makespan and energy consumption of workflows' execution, we employ the MOHEFT multi-objective scheduler developed by us in previous work [5].

The rest of this paper is structured as follows. Section II formally describes the scheduling problem tackled in this work. Section III reviews existing models for energy consumption and execution time of tasks, and introduces our novel modelling approach. Following, the MOHEFT algorithm is introduced in Section IV. We present in Section V an evaluation of our algorithm and analyse the obtained results. Section VI reviews existing work on multi-objective workflow and energy-efficient scheduling. Finally, we present in Section VII the principal conclusions and future work ideas.

II. Formalism

Our problem consists in scheduling the workflow tasks on the available resources in such a way that the makespan and the energy consumption of that workflow execution are minimized. We introduce in the remainder of this section a simple but realistic formalism that defines the workflow, resource environment, and the metrics targeted.

A. Workflow Application

We model a *workflow application* as a directed acyclic graph (DAG), $W = (A, D)$ consisting of n tasks or activities (in the remainder we use both terms interchangeably): $A = \bigcup_{i=1}^{n} \{A_i\}$, interconnected through control flow and data flow dependencies, D, defined as:

$$D = \{(A_i, A_j, Data_{ij}) \mid (A_i, A_j) \in A \times A\},$$

where $Data_{ij}$ represents the size of the data needed to be transferred from activity A_i to activity A_j. We use $pred(A_i) = \{A_k \mid (A_k, A_i, Data_{ki}) \in D\}$ to denote the set of *predecessors* of activity A_i (i.e. activities to be completed before starting A_i). Every activity $A_i \in A$ is characterised by its length (or workload) measured for example in total number of instructions. The execution time and the energy consumption entailed by an activity execution depend on the tasks's length and the resource on which it executes.

B. Resource Environment

We assume that our hardware platform consists of a set of m heterogeneous resources $R = \cup_{j=1}^{m} R_j$. Each resource $R_j \in R$ is described by a set of nine different characteristics influencing the number of machine instructions per second that it executes, and the energy it consumes during this. A brief description of these nine characteristics is summarised in Table I. We use $sched(A_i)$ to denote the resource on which activity A_i is scheduled to be executed.

C. Makespan

For computing the workflow makespan, we first define the *completion time* $T_c^{(A_i)}$ of an activity A_i on resource

Table I
Resource characteristics.

Characteristic	*Description*
Technology [nm]	Dimension of the CPU lithography (e.g. 32nm)
Architecture [bits]	CPU architecture (e.g. 32 or 64 bits)
Min frequency [GHz]	Minimum processor frequency
Frequency [GHz]	Nominal processor frequency
Cache size [KB]	Level of cache memory size
Cache sharing	Number of cores sharing the last level cache
Cores	Number of cores on the CPU
Threads	Number of concurrent hardware threads[i]
TDP	CPU Thermal Design Point[ii]

[i] Twice the number of physical cores for hyper-threading. [ii] Theoretical maximum heat dissipation requirement for not exceeding the maximum junction temperature; also a rough indicator of the power consumption class of the CPU.

$R_j = sched(A_i)$ as the maximum completion time of its predecessors, including their data transfers to R_j, plus its own total execution time $t_{(A_i, R_j)}$:

$$T_c^{(A_i)} = \begin{cases} t_{(A_i,R_j)}, & pred(A_i) = \emptyset; \\ \max\limits_{A_p \in pred(A_i)} \left\{ T_c^{(A_p)} + \frac{Data_{pi}}{b_{pj}} \right\} + t_{(A_i,R_j)}, & pred(A_i) \neq \emptyset, \end{cases} \tag{1}$$

where $t_{(A_i,R_j)}$ is the *computation time* of activity A_i on R_j, $Data_{pi}$ is the number of bytes transferred between A_p and A_i, and b_{pj} is the bandwidth of one TCP stream between $sched(A_p)$ and R_j ($b_{pj} = \infty$, if $sched(A_p) = R_j$).

Finally, we compute the workflow makespan as the maximum completion time of all its n activities:

$$T_W = \max_{i \in [1,n]} \left\{ T_c^{(A_i, sched(A_i))} \right\}. \tag{2}$$

D. Energy Consumption

We represent the total energy E_W consumed by a workflow execution as the sum of the energy E_D consumed for all data transfers and the energy E_C consumed for executing all its computational activities:

$$E_W = E_D + E_C. \tag{3}$$

Given a workflow schedule, we define E_D as follows:

$$E_D = \sum_{\substack{(A_i, A_j, Data_{ij}) \in D \wedge \\ sched(A_i) \neq sched(A_j)}} \mathcal{E}_{ij} \cdot Data_{ij}, \tag{4}$$

where $Data_{ij}$ is the number of bytes transferred between A_i and A_j, and $\mathcal{E}_{ij}$ is a characteristic value representing the energy expended for transferring one byte of data between $sched(A_i)$ and $sched(A_j)$, which neglects the energy consumed by the external networking equipment.

Similarly, we define the energy E_C consumed by the computational activities of a workflow as the sum of the energy $E_{R_j}^{(A_i)}$ consumed by resource $R_j = sched(A_i)$ for executing activity A_i:

$$E_C = \sum_{i=1}^{n} E_{R_j}^{(A_i)}, \tag{5}$$

where n is the total number of workflow activities.

We further define the energy $E_{R_j}^{(A_i)}$ consumed by resource $R_j = sched\,(A_i)$ executing activity A_i as follows:

$$E_{R_j}^{(A_i)} = P_{R_j}^{(s)} \cdot t_{(A_i,R_j)} + \int_{T_s^{(A_i)}}^{T_c^{(A_i)}} P_{R_j}^{(d)}(t)\ \mathrm{d}t, \qquad (6)$$

where $P_{R_j}^{(s)}$ refers to the static power consumption of the resource R_j in idle state, $P_{R_j}^{(d)}(t)$ represents the additional dynamic power consumption of the same resource stemming from the computations scheduled at time instance t, $t_{(A_i,R_j)}$ is the computation time of A_i on R_j, $T_c^{(A_i)}$ its completion time, and $T_s^{(A_i)} = T_c^{(A_i)} - t_{(A_i,R_j)}$ its start time.

E. Problem Definition

Given a workflow $W = (A, D)$, our goal is to approximate the *Pareto-optimal workflow schedules* $sched(W) = \bigcup_{i=1}^{n} sched\,(A_i)$ with respect to makespan and energy consumption, where n is the total number of activities.

III. Energy and Execution Time Modelling

A. Background

Our problem formulation relies on two models: one for the number of instructions a resource is able to process every second, and another for the energy consumed by a resource at an instant t. This section provides a concise background overview of existing models and analyses their main weaknesses.

Existing works so far considered multi-core CPU as having as many independent resources as cores within the chip. This fact is reflected by ignoring the overhead of using different cores in the same CPU for computing the makespan [6], [7], [8], [9], [10], [11] and by considering only two different levels of energy consumption: idle or fully loaded [12], [13], [14], [15], [16], [17]. Here, however, we prove thorough an extensive experimentation that: (1) the performance of each core is impacted by the workload of the other cores; (2) the energy consumption of a multi-core CPU may vary among a multitude of different levels at any instant t; and (3) the energy consumption overhead inherent to powering on a core decreases with the number of cores already powered on.

To substantiate our claims, we measured the execution time and the associated energy consumption of a workflow activity called `povray`, belonging to a the Persistence Of Vision Raytracer (POV-Ray), a real workflow application [18]. We focus our analysis on this activity which renders a set of frames (i.e. images) from a three-dimensional scene descriptor file as it is highly parallelisable and the most time-consuming part of the workflow. We measured the execution time and energy consumption of the `povray` activity using different workload sizes on a diverse pool of resources, described in Table II. We obtained these measurements through a tool we developed in Python and C which retrieves online measurements from multiple LAN-enabled Voltech PM1000+[2] power measurement devices connected to the machines.

Table II
Resource configuration for modelling time and energy consumption of the `povray` workflow task (all machines are 64 bit architectures and the minimum operational frequency is omitted).

CPU Model	*Tech. [nm]*	*Freq. [GHz]*	*Cache size [KB]*	*Cache sharing*	*Cores*	*Threads*	*TDP [W]*	*CPU no.*	*Inst.*
Opteron 8356	65	2.3	2048	1	4	4	95	8	2
Opteron 6168	45	1.9	12288	6	12	12	115	2	2
Xeon X5650	32	2.66	12288	6	6	12	95	2	2
Xeon E7-4870	32	2.4	30720	1	10	20	130	4	1
Opteron 880	90	2.4	2048	1	2	2	95	4	1
Opteron 885	90	2.6	2048	1	2	2	95	8	1

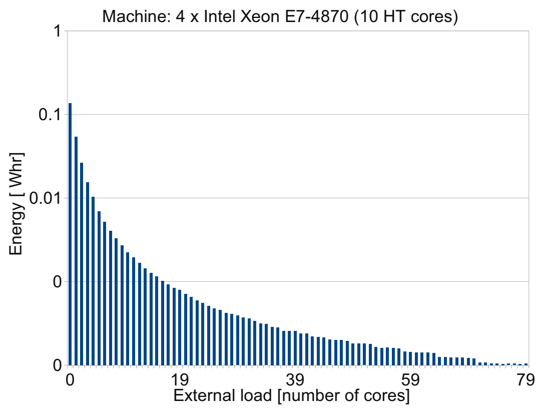

Figure 1. Energy consumption of a single core within a multi-core machine (Intel with Hyper-Threading) with increasing external load until reaching machine saturation.

Figure 1 presents the variation in energy consumption of one CPU core with increasing external load (number of running threads) on the remaining CPU cores. The decrease in energy consumption per core with increasing external load is because different cores share a set of common subsystems (e.g. cache memory, memory bus, memory controller) that consume relatively the same amount of energy when utilised by any number of cores, as they quickly reach saturation state. Regarding to the performance, our experiments have revealed that the time of computing one frame considerably increases when other cores are concurrently utilised. For example, in the same machine used in Figure 1, we have observed a slowdown of 34.2% in case of 39 threads of external load compared to no external load.

In light of these findings, we affirm that existing models for performance and energy consumption of multi-core machines must be adjusted or, alternatively, new models accurately reflecting the real behaviour of these types of resources need to be researched.

[2]http://www.voltech.com/products/poweranalyzers/PM1000.aspx

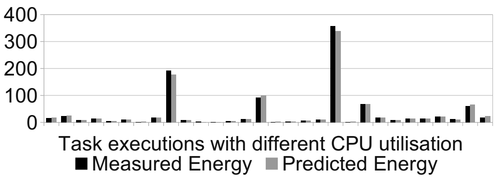

Figure 2. Neural network prediction vs. measured energy consumption.

B. Proposed Models

In this paper we investigate the use of neural network predictors [19] for designing models reflecting the execution time and energy consumption of workflow activities in different resources. To each activity type we associate two neural networks: one for estimating its execution time on any activity configuration – input size – resource combination, and the other providing the corresponding energy consumption estimations. We consider this approach to be a valid alternative in the context of scientific workflows, where the type and number of activities is limited and a priori known.

Each of our predictors consists of a Multi-Layer Perceptron (MLPs) with one hidden layer, which requires as input the different characteristics of the machine (as described in Table I) the activity will be executed on, along with the input size (e.g. the number of frames to render in the case of the `povray` task analysed before) and the external load on the resource. To validate this approach, we use the data collected from executions of the real `povray` workflow task on our diverse set of resources (shown in Table II) for neural network training and, eventually, for cross-validation.

We conducted 20 experiments consisting of network training and testing cycles, using in each instance a different selection of the training and testing sets, and measured an average prediction accuracy of 97% and a minimum of 95%. Figure 2 presents a sample of a validation trace comprising real measured energy consumption values and the corresponding estimation values by the trained neural network predictor. The values represent the energy expended for rendering $1,000$ frames for different configurations consisting of various machines with different external loads (in no particular order).

IV. MOHEFT: Multi-Objective Heterogeneous Earliest Finish Time Algorithm

Here we describe MOHEFT, the algorithm used for computing optimal solutions for makespan and energy consumption. This algorithm extends HEFT [6], a list based-heuristic for optimizing makespan.

HEFT is a greedy constructive algorithm which builds a solution by iteratively mapping tasks on resources. This mapping aims at minimising the completion time of every task; therefore, in every iteration it assigns the current task to the resource which minimises this goal. When multiple optimisation goals are considered, we must create several solutions at the same time because the goal is to approximate the whole Pareto front instead of a single solution. This could be achieved by scheduling each task on all resources offering a tradeoff between the considered objectives, and this is the idea MOHEFT is based on (see Algorithm 1).

Algorithm 1 MOHEFT algorithm.

```
Require: $W = (A, D), A = \bigcup_{i=1}^{n} A_i$          ▷ Workflow application
Require: $R = \bigcup_{i=1}^{m} R_i$                      ▷ Set of resources
Require: $K$                                              ▷ Number of tradeoff solutions
Ensure: $S = \bigcup_{i=1}^{K} sched(W)$                  ▷ Set of $K$ tradeoff workflow schedules
 1: function MOHEFT($W$,$R$,$K$)
 2:     $B \leftarrow$ B-RANK($A$)                        ▷ Order the tasks according to B-rank
 3:     for $k \leftarrow 1, K$ do                        ▷ Create $K$ empty workflow schedules
 4:         $S_k \leftarrow \emptyset$
 5:     end for
 6:     for $i \leftarrow 1, n$ do                        ▷ Iterate over the $n$ ranked tasks
 7:         $S' \leftarrow \emptyset$
 8:         for $j \leftarrow 1, m$ do                    ▷ Iterate over all $m$ resources
 9:             for $k \leftarrow 1, K$ do                ▷ Iterate over all $K$ tradeoff schedules
10:                 $S' \leftarrow S' \cup \{S_k \cup (B_i, R_j)\}$    ▷ Add new mapping to all intermediate schedules
11:             end for
12:         end for
13:         $S' \leftarrow$ SORTCROWDDIST($S'$, $K$)▷ Sort according to crowding distance
14:         $S \leftarrow$ FIRST($S'$, $K$)              ▷ Choose $K$ schedules with highest crowding distance
15:     end for
16: return $S$
17: end function
```

The only additional input parameter of MOHEFT with respect to HEFT is the size of the set of tradeoff solutions, K. MOHEFT first orders the tasks using the B-rank algorithm (line 2, see details of the algorithm in [6]). Then it creates a set S of K empty solutions (lines 3–5). Afterwards, the mapping phase begins (lines 6–15) in which MOHEFT iterates first over the list of ranked tasks (line 6). The idea is to extend every solution in S by mapping the next task onto all possible resources, creating m new solutions that are stored in a temporal set S' (initially empty (line 7)). For creating these new solutions, it iterates over the set of resources (line 8) and the set S (line 9), and adds the new extended intermediate schedules to the new set S' (line 10). This strategy results in an exhaustive search if we do not include any restrictions. We therefore only save the best K tradeoffs solutions from the temporary set S' to the set S (lines 13–14). Our criterion is to prefer solutions increasing the diversity of the tradeoff (the K most different optimal solutions). To this end, we have employed a concept borrowed from the multi-objective optimisation theory: the crowding distance defined in [20].

In [5], we showed that MOHEFT computed solutions of the same quality of the ones computed by HEFT in terms of makespan, and in some cases of higher quality.

V. Experiments on Improving the Energy Efficiency of Workflow Schedules

In this section we present an analysis of the tradeoff possibilities between makespan and energy consumption when scheduling workflows with MOHEFT.

Table III
ANALYSED WORKFLOW SHAPES; EACH WORKFLOW SHAPE IS ASSIGNED TO A DIFFERENT WORKFLOW TYPE (FROM 1 TO 4).

Type	*Description*
Type-1	Many independent activities sharing a successor and predecessor
Type-2	Many independent activities with different successors and predecessors
Type-3	Few independent activities, where at most two activities run in parallel within any workflow region
Type-4	Alternating workflow regions with many independent activities (parallel regions) and regions with few independent activities (sequential regions)

Our experiments focus on two main aspects: workflows and resources. First, we analyse in Section V-A the potential for energy improvement of four workflow classes, distinct by their shape and number of tasks. Second, we analyse the impact of the resources' static (Section V-B) and dynamic (Section V-C) power consumption components (introduced in Equation 6) on the energy efficiency of the workflow schedules.

Workflows: The four workflow types analysed here, distinct in shape and size (i.e., number of activities), are created using a generator of synthetic workflows based on the one described in [21]. A brief description of the generated workflow types is presented in Table III. We generated the activities composing these workflows using the real `povray` activity as template, introduced in Section III. The reasons for selecting this activity are twofold: on the one hand, it is an activity belonging to a real application and, on the other hand, we can directly apply the prediction models for activity performance and energy consumption developed in Section III-B.

Resources: We generated the resources considered in our experiments using a synthetic resource generator. This generator takes as input the description of realistic commercial CPUs and generates complete resource setups consisting of several clusters comprising multiple nodes, each node being a multi-core machine. Each machine is defined by the nine characteristics described in Table I, along with other characteristics such as its power consumption in idle state $P^{(s)}_{R_j}$, or the amount of installed memory.

Evaluation: We evaluate the potential for energy efficiency improvement of workflow schedules by plotting the tradeoff function between their makespan and energy consumption. We employ MOHEFT for computing the optimal tradeoff scheduling solutions. The presented graphs start with the most makespan-efficient schedule and continue along the Pareto-front towards the most energy-efficient solution. The two plotted metrics, the energy improvement and the makespan deterioration, are presented as percentages relative to the most makespan-efficient solution computed by HEFT.

A. Impact of the Workflow Shape and Size

This experiment analyses the impact of the workflows' shape and size on the computed tradeoff solutions. The considered workflows are composed of 20–200 distinct activities and the resource pool is heterogeneous, with a much higher number of resources than the workflows' sizes. Each workflow type's energy savings potential is depicted in Figure 3.

In the case of *Type-1* workflows, the maximum energy savings are between 58% (workflow size 50) and 90% (workflow size 200) depending on the number of activities composing the workflow. These improvements, however, incur a serious makespan deterioration, between 43% (50 activity workflow) and 428% (200 activity workflow). Even though this workflow type requires a significant concession in execution time for reaching the most energy-efficient solution, we also must emphasise that, interestingly, considerable energy can be saved, up to 50%, with only a makespan tradeoff of below 20%. For this workflow type, a higher number of activities indicates, in general, more opportunities for increasing energy gains.

Type-2 and *Type-3* workflows exhibit nearly identical behaviour in terms of energy savings potential. With a merely 8% concession in makespan these workflows' execution will consume 78% less energy. Regarding the workflow size, we observe that workflows with more activities exhibit a more regular, easily quantifiable energy-makespan tradeoff behaviour rather than their smaller sized counterparts.

Type-4 workflows' energy-makespan tradeoff characteristic is significantly influenced by their size. Considerable energy savings can be made, but we observe less makespan deterioration for bigger sized workflows: only 3.3% increase in makespan resulting in 85% less energy being consumed for workflows with 100 activities versus an 8.2% makespan increase for an 80% energy gain for workflows with 20 activities. On the other hand, we also observe that very big workflows might not offer enough energy optimisation potential (e.g. maximum 40% energy savings for workflows of size 200).

Based on this analysis we conclude that both workflow shape and size strongly impact then energy efficiency of its execution and that, regardless of the workflow type, significant energy can be saved (between 75% an 85%) with minimal makespan deterioration (8% in most cases).

B. Impact of the Resources' Static Energy Consumption

The aim of this experiment is to analyse whether the resources' static energy consumption components influence the energy-makespan tradeoff. To this end, we utilise three resource pools, each composed of heterogeneous machine types, but with different characteristic levels of static energy consumption (i.e. energy consumed while in idle state, see Equation 6): (1) the *low* static energy consumption, with values distributed between 20Whr and 400Whr, and centred around 20Whr, (2) the *medium*, centred around 120Whr, and (3) *high*, centred around 400Whr. We consider all workflow types, but keep the size constant at 200 activities.

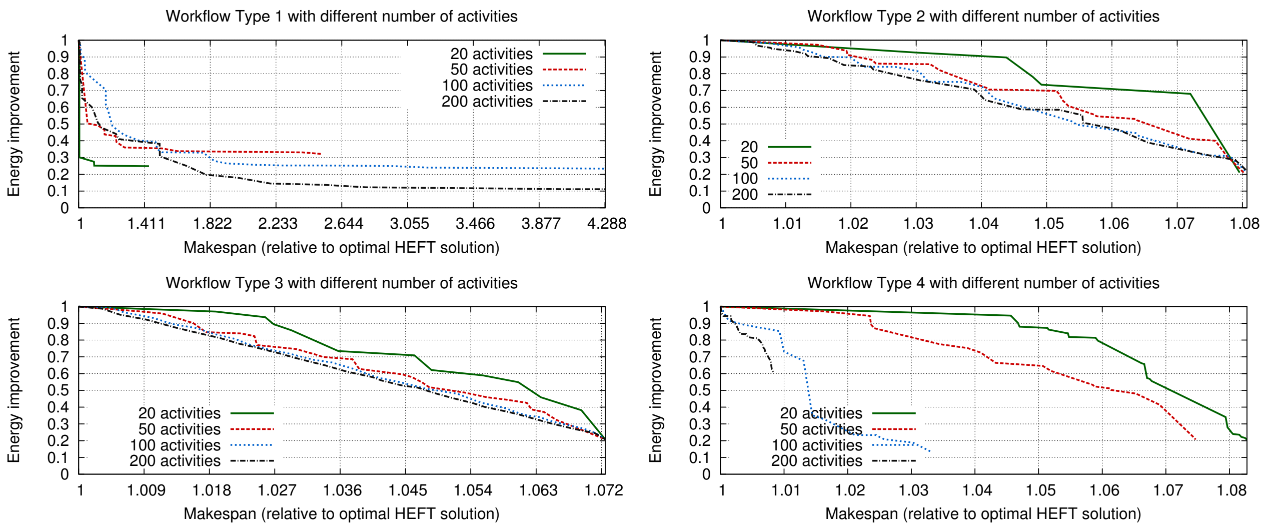

Figure 3. *Energy savings vs makespan deterioration* of the tradeoff solutions computed by MOHEFT for workflows with different number of activities.

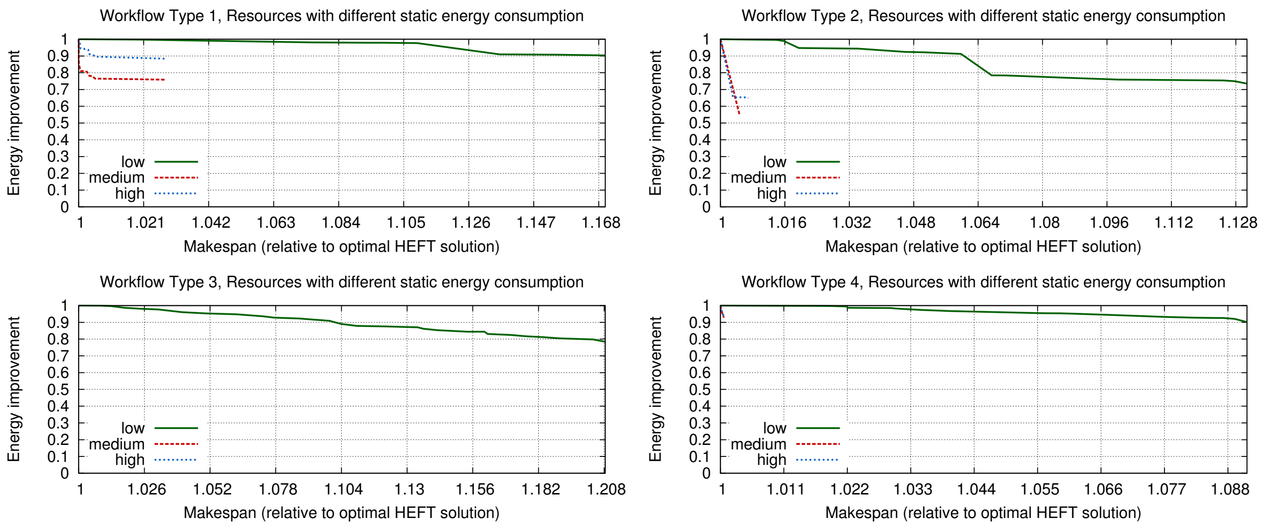

Figure 4. *Energy savings vs makespan deterioration* of the tradeoff schedules computed by MOHEFT for resources with different static energy consumption.

Figure 4 shows a clearer relationship between the tradeoff components than in the previous analysis. The *low* static energy resources offer a wide range of compromise solutions, but the energy savings are limited to a maximum of 25% (*Type-2* workflows). On the other hand, the *medium* and *high* static energy resources exhibit a higher potential for energy savings (up to 45% for *Type-2* workflows), but seem to favour highly parallel workflows, as no tradeoff solutions were found for *Type-3* workflows and only marginal improvements are reached for *Type-4* workflows.

C. Experiment-3: Impact of CPU TDP

This last part of our evaluation complements the study of the previous section. We investigate here the impact of the dynamic component of the resources' energy consumption (see Equation 6) on the energy-makespan tradeoff potential. While most of a resource's subcomponents contribute to its dynamic energy consumption, here we study only the impact of its dominant component, the CPU's energy consumption. We classify CPUs based on their thermal design power (TDP) rating, which provides a good approximation of their characteristic energy consumption. Similarly to the idle study, we define three classes of dynamic energy consumption, *low* with CPU TDPs between 35W and 55W, *medium* between 55W and 90W, and *high*, between 90W and 130W. The workflows in this experiment are the same as in the previous section.

Figure 5 shows the tradeoff fronts for the first three types of workflows. *Type-4* workflows are not presented since MOHEFT produced only one schedule (optimal for

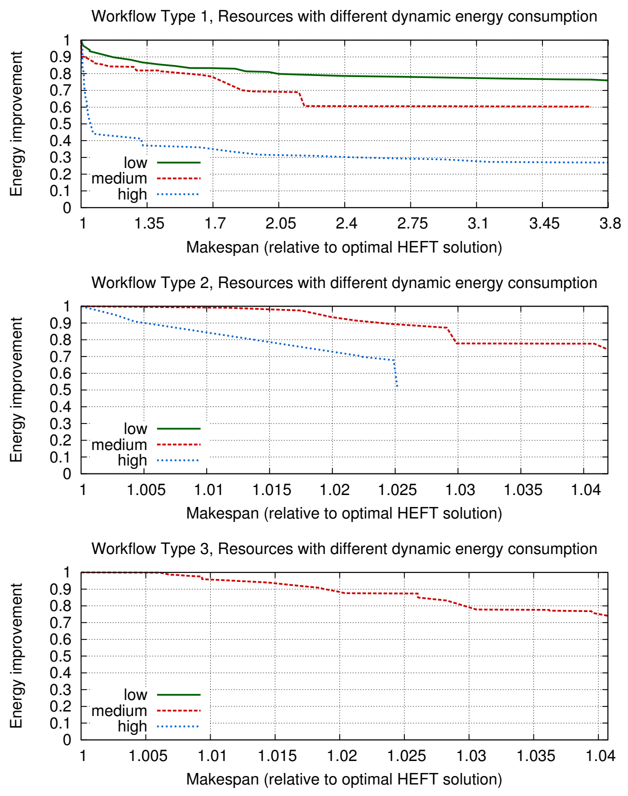

Figure 5. *Energy savings vs makespan deterioration* of the tradeoff schedules computed by MOHEFT for resources with different dynamic energy consumption.

energy and makespan). We observe that, in contrast to the static energy case, the energy-makespan tradeoff front is highly correlated to the dynamic energy component of the resources. Firstly, considerable energy savings can be obtained in most cases: up to 55% with only a 3.7% increase in makespan for *Type-1*, up to 48% with only a 2.5% makespan increase for *Type-2*, and up to 25% with 4% makespan increase for *Type-3*. Secondly, and more significantly, there is a high correlation between the resources' dynamic energy component and the energy-makespan tradeoff potential. For example, in workflows of *Type-1* and *Type-2* we observe that the higher the resource pool's TDP the higher the possibility for energy optimisation with a lower hit to the makespan. This latter statement can be visually confirmed by the graphs presented in Figure 5, as all tradeoff trends appear to preserve the same order as their TDP classes and do not intersect (e.g. the *high* TDP trace is below the *medium* trace and does not intersect it).

We conclude that the energy characteristic of the resources highly impacts the energy efficiency improvement potential. Both, static and dynamic energy components, influence the tradeoff front, with a more significant contribution from the dynamic component. Finally, we also note that any eventual tradeoff opportunity highly depend on the type of the workflow being scheduled, with the highly parallel ones offering the most tradeoff solutions and very often the highest energy savings opportunities.

VI. Related Work

Most of the existing multi-objective workflow schedulers reduce the problem to a single optimisation one. Two different approaches have been used so far: aggregation of the objective functions in a single analytical function and optimisation of a single objective while keeping the other within predefined constraints. In any of these two cases, however, only a single solution is computed instead of a Pareto front, thus depriving the user of the information offered by this last. Related works doing so have targeted the optimisation of makespan and reliability [7], [22], [10] or makespan and cost utility [23], with no attention to energy consumption.

Among existing approaches optimising energy efficiency and makespan, we can find genetic algorithms [13], [14] and heuristic techniques [15], [16], [17]. Most of the work in this area incorporate DVFS tuning for reducing the energy consumption. Among the techniques which do no use DVFS, a common strategy is to power-down or turn off unused resources to sleep mode [24], aiming at reducing the energy consumed at idle state.

Only few works computing the Pareto front for workflow scheduling have been proposed so far. Some of these works are based on the use of genetic algorithms for seeking tradeoff solutions between makespan and financial cost in Cloud systems [11] or for optimal scheduling considering makespan and energy consumption using DVFS techniques [12]. The main drawback of genetic algorithms is the required time to converge towards solutions of high quality. To overcome this issue, a few multi-objective list-based schedulers have been proposed [25], [5] but they have never been applied in the context of energy-efficient scheduling.

To the best of our knowledge, our work is the first truly multi-objective approach able to compute a set of tradeoff solutions optimising makespan and energy consumption in distributed systems not relying on DVFS techniques. Our method also incorporates power-down techniques by accounting only for the energy consumed by the machines doing computations. Although not used, our method can be also combined with DVFS in a hybrid approach.

VII. Conclusions and Future Work

In this paper we have tackled the problem of energy-efficient workflow scheduling in distributed heterogeneous systems. Our emphasis is to analyse the tradeoff solutions computed by a multi-objective optimisation algorithm called MOHEFT in terms of energy consumption and makespan.

In our experiments, MOHEFT computed workflow schedules that reduce the energy consumption in up to 85% by incurring only a 3.3% makespan deterioration in some cases. In general, MOHEFT provides the user with different tradeoff makespan-energy schedules from which a user can choose the solution of his interest. The obtained results for the workflows evaluated in this paper show that considerable energy savings can be obtained with small makespan trade-offs. Additionally, we observed that opportunities for energy optimisation depend on the workflow shape and number of activities, and are highly correlated with the dynamic energy consumed by the resources.

In future work, we plan to extend our multi-objective space with more optimisation goals, such as economic cost of workflow executions, a criterion of particular interest in the context of the emerging commercial Clouds.

AKNOWLEDGMENT

Austrian Science Fund project TRP 72-N23 and Standortagentur Tirol project RainCloud funded this research.

REFERENCES

[1] J. Hamilton, "Cooperative expendable micro-slice servers (cems): Low cost, low power servers for int. et-scale services."

[2] E. Humenay, D. Tarjan, and K. Skadron, "Impact of process variations on multicore performance symmetry," in *Proc. of the conf. on Design, automation and test in Europe*, ser. DATE '07. San Jose, CA, USA: EDA Consortium, 2007, pp. 1653–1658.

[3] T. Burd, T. Pering, A. Stratakos, and R. Brodersen, "A dynamic voltage scaled microprocessor system," in *Solid-State Circuits Conference, 2000. Digest of Technical Papers. ISSCC. 2000 IEEE Int. .*, 2000, pp. 294 –295, 466.

[4] A. Chandrakasan, S. Sheng, and R. Brodersen, "Low-power cmos digital design," *Solid-State Circuits, IEEE Journal of*, vol. 27, no. 4, pp. 473 –484, apr 1992.

[5] J. Durillo, H. Fard, and R. Prodan, "Moheft: A multi-objective lilst-based method for workflow scheduling," in *4th IEEE Inter. Conference on Cloud Computing Technology and Science*, December 2012.

[6] H. Topcuoglu, S. Hariri, and M.-Y. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," *Parallel and Distributed Systems, IEEE Trans. on*, vol. 13, no. 3, pp. 260 –274, mar 2002.

[7] S.-K. Garg, R. Buyya, and H. J. Siegel, "Scheduling parallel applications on utility grids: time and cost trade-off management," in *Proc. of the Thirty-Second Australasian Conf. on Computer Science - Volume 91*, ser. ACSC '09. Darlinghurst, Australia: Australian Computer Society, Inc., 2009, pp. 151–160.

[8] E. Ilavarsan and P. Thambidurai, "Low complexity performance effective task scheduling algorithm for heterogeneous computing environments," *Journal of Computer Science*, vol. 3, no. 2, pp. 94–103, 2007.

[9] X. Wang, R. Buyya, and J. Su, "Reliability-oriented genetic algorithm for workflow applications using max-min strategy," in *Proc. of the 2009 9th IEEE/ACM Int. . Symposium on Cluster Computing and the Grid*, ser. CCGRID '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 108–115.

[10] I. Assayad, A. Girault, and H. Kalla, "A bi-criteria scheduling heuristics for distributed embedded systems under reliability and real-time constraints," in *Int. . Conf. on Dependable Systems and Networks, DSN'04*. Firenze, Italy: IEEE, Jun. 2003.

[11] J. Yu, M. Kirley, and R. Buyya, "Multi-objective planning for workflow execution on grids," in *Proc. of the 8th IEEE/ACM Int. Conf. on Grid Computing*, ser. GRID '07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 10–17.

[12] M. Mezmaz, N. Melab, Y. Kessaci, Y. Lee, E.-G. Talbi, A.Y.Zomaya, and D. Tuyttens, "A parallel bi-objective hybrid metaheuristic for energy-aware scheduling for cloud computing systems," *Journal of Parallel and Distributed Computing*, no. 71, pp. 1497–1508, 2011.

[13] J. Kolodziej, S. U. Khan, and F. Xhafa, "Genetic algorithms for energy-aware scheduling in computational grids," in *2011 Int. . Conf. on P2P, Parallel, Grid, Cloud and Int. et Computing*, 2011.

[14] J. Kolodziej, S. Khan, L. Wang, and A. Zomaya, "Energy efficient genetic-based schedulers in computational grids," *Concurrency and Computation: Practice and Experience*, pp. 1497–1508, 2012.

[15] C. Diaz, M. Guzek, J. Pecero, G. Danoy, P. Bouvry, and S. Khan, "Energy-aware fast scheduling heuristics in heterogeneous computing systems," in *High Performance Computing and Simulation (HPCS), 2011 Inter. Conference on*, july 2011, pp. 478 –484.

[16] P. Lindberg, J. Leingang, D. Lysaker, S. U. Khan, and J. Li, "Comparison and analysis of eight scheduling heuristics for the optimization of energy consumption and makespan in large-scale distributed systems," *Journal Of SuperComputing*, pp. 478 –484, April 2010.

[17] N. Min-Allah, H. Hussain, S. Khan, and A. Y. Zomaya, "Power efficient rate monotici scheduling for multi-core systems," *Journal of Parallel and Distributed Computing*, pp. 48–57, January 2012.

[18] http://www.povray.org/.

[19] C.-M. Bishop, *Neural Networks for Pattern Recognition*, 1st ed. Oxford University Press, USA, Jan. 1996.

[20] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast elitist multi-objective genetic algorithm: Nsga-ii," *IEEE Trans. on Evol. Computation*, vol. 6, pp. 182–197, 2000.

[21] J. Yu, R. Buyya, and K. Ramamohanarao, "Workflow scheduling algorithms for grid computing," in *Metaheuristics for Scheduling in Distributed Computing Environments*, F. Xhafa and A. Abraham, Eds. Springer Berlin, 2008, pp. 109–153.

[22] M. Hakem and F. Butelle, "Reliability and scheduling on systems subject to failures," in *Proc. of the 2007 Inter. Conference on Parallel Processing*, ser. ICPP '07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 38–.

[23] R. Sakellariou, H. Zhao, E. Tsiakkouri, and M.-D. Dikaiakos, "Scheduling workflows with budget constraints," in *in Integrated Research in Grid Computing, S. Gorlatch and M. Danelutto, Eds.: CoreGrid series*. Springer-Verlag, 2007.

[24] S. Khuller, J. Li, and B. Saha, "Energy efficient scheduling via partial shutdown," in *Proc. of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA '10. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2010, pp. 1360–1372.

[25] L. Canon and E., "Mo-greedy: an extended beam-search approach for solving a multi-criteria scheduling problem on heterogeneous machines," ser. Inter. Heterogeneity in Computing, 2011.