Evaluating energy-aware scheduling algorithms for I/O-intensive scientific workflows

Tainã Coleman[1], Henri Casanova[2], Ty Gwartney[2], and Rafael Ferreira da Silva[1]

[1] USC Information Sciences Institute, Marina del Rey, CA, USA
[2] Information and Computer Sciences, University of Hawaii, Honolulu, HI, USA
{tcoleman,rafsilva}@isi.edu, {henric,tyg}@hawaii.edu

Abstract. Improving energy efficiency has become necessary to enable sustainable computational science. At the same time, scientific workflows are key in facilitating distributed computing in virtually all domain sciences. As data and computational requirements increase, I/O-intensive workflows have become prevalent. In this work, we evaluate the ability of two popular energy-aware workflow scheduling algorithms to provide effective schedules for this class of workflow applications, that is, schedules that strike a good compromise between workflow execution time and energy consumption. These two algorithms make decisions based on a widely used power consumption model that simply assumes linear correlation to CPU usage. Previous work has shown this model to be inaccurate, in particular for modeling power consumption of I/O-intensive workflow executions, and has proposed an accurate model. We evaluate the effectiveness of the two aforementioned algorithms based on this accurate model. We find that, when making their decisions, these algorithms can underestimate power consumption by up to 360%, which makes it unclear how well these algorithm would fare in practice. To evaluate the benefit of using the more accurate power consumption model in practice, we propose a simple scheduling algorithm that relies on this model to balance the I/O load across the available compute resources. Experimental results show that this algorithm achieves more desirable compromises between energy consumption and workflow execution time than the two popular algorithms.

Keywords: Scientific workflows · Energy-aware computing · Workflow scheduling · Workflow simulation.

1 Introduction

Scientific workflows have become mainstream for the automated execution of computational workloads on parallel and distributed computing platforms. Over the past two decades, workflow applications have become more complex as the volume of data and processing needs have substantially increased. At the same time, computational platforms have increased their processing capacity and developed mechanisms for efficient workload management. An integral aspect of

the workload processing lifecycle is energy management by which the system makes scheduling and resource provisioning decisions so as to maximize workload throughput while reducing or bounding energy consumption [9]. In the past few years, energy management for scientific workflows has gained traction, especially due to their singular contributions to major scientific discoveries. As a result, several works have proposed energy-aware workflow scheduling algorithms for a range of computing environments, and in particular cloud computing platforms [8, 11, 12, 22].

These scheduling algorithms have all been designed and evaluated using the traditional model for application power consumption [5, 7, 10, 12, 13, 21]: power consumption is linear in CPU utilization and does not account for I/O operations. In previous work [15, 17], we have experimentally quantified the accuracy of this model and have identified sources of inaccuracy. A key finding was that power consumption on multi-socket, multi-core compute nodes is not linearly related to CPU utilization. Another finding was that, unsurprisingly, I/O operations (i.e., disk reads and writes) significantly impact power consumption. Since many workflows are I/O-intensive, accounting for the power consumption of I/O is crucial for power management of their executions. In that same work, we then proposed a power consumption model that accounts for (i) computations that execute on multi-socket, multi-core compute nodes; and (ii) I/O operations and the idle power consumption caused by waiting for these operations to complete. Experimental results show that this model, unlike the traditional simpler model, has high accuracy with respect to real-world workflow executions on production platforms.

Given the inaccuracies of the traditional power consumption model, it is unclear whether previously proposed energy-aware workflow scheduling algorithms that use this model can be effective in practice. Published evaluations show good results, but these results report on power consumption computed based on that very same, inaccurate, model. One of the goals of this work is to assess the effectiveness on these algorithms in practice. To this end, we evaluate two popular (i.e., most cited according to Google Scholar) energy-aware workflow scheduling algorithms for cloud platforms [11, 22]. We perform this evaluation both when assuming the traditional power consumption model and when assuming the accurate model in [15, 17]. We then propose a simple I/O-aware workflow scheduling algorithm that uses the accurate model, and compare this algorithm to the algorithms in [11, 22]. Specifically, this work makes the following contributions:

1. An evaluation of two popular energy-aware workflow scheduling algorithms [11, 22] when used for executing I/O-intensive workflow applications, using the traditional power model in the literature;
2. An analysis of how the results from the above evaluation differ when the energy consumption of the schedules produced by the two algorithms is estimated using the accurate power model proposed in [15, 17].
3. A simple I/O- and energy-aware workflow scheduling algorithm that leverages the accurate power consumption model proposed in [15, 17] to judi-

ciously allocate threads to cores in multi-socket, multi-core platforms and to reduce the power consumption due to I/O operations.
4. An evaluation of this algorithm, which shows that it achieves a preferable compromise between energy consumption and workflow execution time when compared to the two algorithms in [11, 22].

2 Background

2.1 Scientific Workflows

Scientific workflows are a cornerstone of modern scientific computing, and they have underpinned some of the most significant discoveries of the last decade [4,6]. They are used to describe complex computational applications that require efficient and robust management of large volumes of data, which are typically stored/processed on heterogeneous, distributed resources. In many cases, a workflow can be described as a directed acyclic graph, where the vertices represent tasks and the edges represent data or control dependencies. As workflows continue to be adopted by scientific projects and user communities, they are becoming more complex. Today's production workflows can be composed of millions of individual tasks that execute for milliseconds to hours, and that can be single-threaded programs, multi-threaded programs, tightly coupled parallel programs (e.g., MPI programs), or loosely coupled parallel programs (e.g., MapReduce jobs), all within a single workflow [14]. Many of these workflows are used to analyze terabyte-scale datasets obtained from streams, from files, or object stores. As a result, most workflows comprise I/O-intensive tasks, and many workflows are mostly composed of such tasks [3]. These are the workflows we specifically target in this work.

2.2 Power and Energy Consumption

In [15, 17], we have investigated the impact of CPU utilization and disk I/O operations on the energy usage of I/O-intensive workflow executions on platforms that comprises multi-socket, multi-core compute nodes. In contrast to the traditional power consumption model used in the energy-aware workflow scheduling literature, we find that power consumption is impacted non-linearly by CPU utilization and depends on the way in which workflow tasks are allocated to cores and sockets. Our experimental results also show that I/O operations, as well as the idling due to waiting for these operations to complete, have significant impact on overall power consumption. Based on these results, we proposed a power consumption model for I/O-intensive workflows that accounts for the above phenomena. Experimental evaluation of this model showed that it accurately captures real-world behavior, whereas the traditional model used in the literature can be inaccurate by up to two orders of magnitudes. Below, we briefly describe both models, which are used for the experiments conducted in this work. (Note that neither model accounts for energy consumption due to RAM usage.)

Traditional power and energy model. Energy-aware workflow scheduling studies [5, 7, 10, 12, 13, 21, 22] typically assume that the dynamic power (i.e., not accounting for the host's idle power consumption) consumed by the execution of a task at time t, $P(t)$, is linearly related to the task's CPU utilization, $u(t)$, as:

$$P(t) = (P_{\max} - P_{\min}) \cdot u(t) \cdot \tfrac{1}{n}, \tag{1}$$

where $P_{\max}$ is the power consumption when the compute node is at its maximum utilization, $P_{\min}$ is the idle power consumption (i.e., when there is no or only background activity), and n is the number of cores. Note that $u(t)$ can be computed by benchmarking the task on a dedicated compute node.

Given this power consumption model, the energy consumption of a task, E, is modeled as follows:

$$E = r \cdot P_{\min} + \int_0^r P(t)\mathrm{d}t, \tag{2}$$

where r denotes the task's execution time.

I/O-aware power consumption model. The model proposed in [15,17] models $P(t)$, the power consumption of a compute node at time t, as:

$$P(t) = P_{\mathrm{CPU}}(t) + P_{\mathrm{I/O}}(t), \tag{3}$$

where $P_{\mathrm{CPU}}(t)$, resp. $P_{\mathrm{I/O}}(t)$, is the power consumption due to CPU utilization, resp. I/O operations.

Let s denote the number of sockets on the compute node, and n the number of cores per socket, so that the total number of cores on the compute node is $s \cdot n$. Let K denote the set of tasks that use at least one core on the compute node. $P_{\mathrm{CPU}}(t)$ is then defined as follows:

$$P_{\mathrm{CPU}}(t) = \sum_{k,i,j} P_{\mathrm{CPU}}(k, i, j, t), \tag{4}$$

where $P_{\mathrm{CPU}}(k, i, j, t)$ is the power consumption of CPU utilization at time t due to the execution of task k ($k \in K$) on socket i ($0 \leq i < s$) at core j ($0 \leq j < n$) on the compute node. Experiments on real-world systems show that power consumption does not linearly increase as cores on sockets are allocated to workflow tasks, and that the behavior depends on the scheme used to allocate each additional core on a socket. We consider two such schemes: (i) *unpaired* – cores are allocated to tasks in sequence on a single socket until all cores on that socket are allocated, and then cores on the next socket are allocated in sequence; and (ii) *pairwise* – cores are allocated to tasks in round-robin fashion across sockets (i.e., each core is allocated on a different socket than the previously allocated core). Both core allocation schemes can be supported by configuring the hardware/software infrastructure accordingly. Based on experimental results, the following model for $P_{\mathrm{CPU}}(k, i, j, t)$ is derived:

$$P_{\mathrm{CPU}}(k,i,j,t) = \begin{cases} (P_{\max} - P_{\min}) \cdot \frac{u(t)}{s \cdot n} & \text{if } j = 0 \text{ (first core on a socket)} \\ 0.881 \cdot P_{\mathrm{CPU}}(k,i,j-1,t) & \text{if } j > 0 \text{ and } \textit{pairwise} \\ 0.900 \cdot P_{\mathrm{CPU}}(k,i,j-1,t) & \text{if } j > 0 \text{ and } \textit{unpaired} \end{cases} \tag{5}$$

where $u(t)$ is the task's CPU utilization at time t. The model is written recursively as the power consumption due to allocating a task to a core on a socket depends on the power consumption due to previously allocated cores on that socket. The 0.881 and 0.900 coefficients above are obtained from linear regressions based on measurements obtained on real-world platforms [15, 17].

Similarly to the definition of P_{CPU}, we have:

$$P_{\mathrm{I/O}}(t) = \sum_{k,i,j} P_{\mathrm{I/O}}(k,i,j,t), \tag{6}$$

where $P_{\mathrm{I/O}}(k,i,j,t)$ is the power consumption of I/O operations at time t due to the execution of task k ($k \in K$) on socket i ($0 \leq i < s$) at core j ($0 \leq j < n$) on the compute node. $P_{\mathrm{I/O}}(k,i,j,t)$ is modeled as follows:

$$P_{\mathrm{I/O}}(k,i,j,t) = \begin{cases} 0.486 \cdot (1+0.317 \cdot \omega(t)) \cdot P_{\mathrm{CPU}}(k,i,j,t) & \text{if } \textit{pairwise} \\ 0.213 \cdot (1+0.317 \cdot \omega(t)) \cdot P_{\mathrm{CPU}}(k,i,j,t) & \text{otherwise} \end{cases} \tag{7}$$

where the 0.486 and 0.213 values above come from linear regressions [15, 17], and $\omega(t)$ is 0 if I/O resources are not saturated at time t, or 1 if they are (i.e., idle time due to IOWait). More precisely, $\omega(t)$ is equal to 1 whenever the volume of I/O requests placed by concurrently running tasks exceeds some platform-dependent maximum I/O throughput. In Eq. 7, $\omega(t)$ is weighted by an application-independent single factor (0.317).

A detailed description and evaluation of the above model is available in [15, 17]. In this work, we limit our analysis to the *unpaired* scheme, as it yields the lowest energy consumption of the two schemes. For simplicity, in the rest of this paper we denote the above model for the unpaired scheme as the ***realistic*** model (in contrast to the *traditional* model described earlier).

3 Analysis of Energy-aware Workflow Scheduling Algorithms

In this section, we describe two widely-used energy-aware workflow scheduling algorithms that leverage the traditional power consumption model for making scheduling decisions described in the previous section. We then evaluate the energy consumption for schedules computed by these two algorithms using both the traditional and the realistic models. We do so by using a simulator that can simulate the power consumption of a workflow execution on a compute platform for either model. We perform these simulations based on real-world execution traces of three I/O-intensive workflow applications. The specific scheduling problem that these algorithms aim to solve is as follows.

Scheduling problem statement. Consider a workflow that consist of single-threaded tasks. This workflow must be executed on a cloud platform that comprises homogeneous, multi-core compute nodes. Initially, all compute nodes are powered off. A compute node can be powered on at any time. Virtual machine

(VM) instances can be created at any time on a node that is powered on. Each VM instance is started for an integral number of hours. After this time expires, the VM is shutdown. A node is automatically powered off if it is not running any VM instance. The cores on a node are never oversubscribed (i.e., a node runs at most as many VM instances as it has cores). A VM runs a single workflow task at a time, which runs uninterrupted from its start until its completion. The metrics to minimize are the workflow execution time, or makespan, and the total energy consumption of the workflow execution.

3.1 SPSS-EB

The Static Provisioning-Static Scheduling under Energy and Budget Constraints (SPSS-EB) algorithm in [11] computes a static resource provisioning and task schedule at the onset of application execution. It considers tasks in topological order (i.e., respecting task dependencies). For each task to be scheduled, with earliest start time t (computed based on the completion times of its already scheduled parent tasks), the algorithm considers the three options below in sequence:

1. If possible, schedule the task to run at the earliest time $t' \geq t$ on a VM instance that is already scheduled to be running at time t' and that will be able to complete the task before this VM instance expires.
2. Otherwise, if possible, schedule a new VM instance to start at time t on a node that has already been scheduled to be powered on and will have at least one idle core at that time, and schedule the task on that instance at time t.
3. Otherwise schedule a new node to be powered on at time t, schedule a new VM instance to be started on that node at time t, and schedule the task to execute on that VM instance at time t.

For each option above, if multiple VM instances or nodes are possible, pick the one that will complete the task the earliest, breaking ties by picking the VM instance or node that will lead to the highest energy saving.

We refer the reader to [11] for a more detailed description of and pseudo-code for the algorithm. Note that the algorithm therein also considers the monetary cost of running VM instances (as charged by the cloud provider), which is used to break ties and also used to evaluate the goodness of the schedule. In this work, we ignore monetary cost and only consider energy consumption and execution time.

3.2 EnReal

Like SPSS-EB, the Energy-aware Resource Allocation (EnReal) algorithm [22] computes a static schedule by considering tasks in topological order. For each task to be scheduled, with earliest start time t (computed based on the completion times of its already scheduled parent tasks), the algorithm follows the following steps:

1. If possible, schedule the task to run at the earliest time $t' \geq t$ on a VM instance that is already scheduled to be running at time t' or shortly after time t' (defined by "time partitions" – computed based on the number of overlapping tasks) and that will be able to complete the task before this VM instance expires. Ties are broken by picking the VM instance that would consume the least amount of energy, and then by picking the VM instance that would lead to a more balanced allocation of tasks on compute nodes.
2. Otherwise, if possible, start a new VM at time $t' \geq t$ on a node that is already scheduled to be on at time t'. Ties are broken by picking the node with the highest number of already scheduled VM instances.
3. Otherwise, schedule a new node to be powered on at time t, schedule a new VM instance to be started on that node at time t, and schedule the task to execute on that VM instance at time t.

We refer the reader to [22] for a more detailed description of and pseudo-code for the algorithm. Note that the algorithm therein also considers migration, which relocates a VM instance from a compute node to another. The objective is to save energy by co-locating computations so as to reduce the number of nodes that are powered-on. In this work, we ignore migration as the energy savings it provides for relatively short-running tasks is marginal [20].

3.3 Workflow Energy Consumption Analysis

The analysis presented in this work is based on the simulated execution of real-world execution traces of scientific workflow applications executed on the Chameleon Cloud [2] platform, an academic cloud testbed. These traces are distributed as part of the WfCommons project [18] and represent a number of different configurations, in which the number of tasks and their characteristics (e.g., input data size, number of I/O operations, flops) vary. Therefore, we argue that these traces form a representative set of small- and large-scale workflow configurations. Specifically, we consider three I/O-intensive workflows:

- *1000Genome* – A bioinformatics workflow that identifies mutational overlaps using data from the 1000 genomes project in order to provide a null distribution for rigorous statistical evaluation of potential disease-related mutations. We consider 15 instances of 1000Genome, with between 260 and 902 tasks.
- *Montage* – An astronomy workflow for generating custom mosaics of the sky. The workflow re-projects images to correct orientation, and rectifying a common flux scale and background level. We consider 9 instances of Montage, with between 59 and 2122 tasks.
- *SoyKB* – A bioinformatics workflow that resequences soybean germplasm lines selected for desirable traits such as oil, protein, soybean cyst nematode resistance, stress resistance, and root system architecture. We consider 9 instances of SoyKB, with between 96 and 676 tasks.

Simulator. We have developed a simulator for our experimental evaluation and validation purposes. The simulator is based on WRENCH [1], a framework for implementing simulators of workflow management systems, with the goal of producing simulators that are accurate and can run scalably on a single computer

while requiring minimal software development effort. In [1], we have demonstrated that WRENCH achieves these objectives, and provides high simulation accuracy for workflow executions using state-of-the-art workflow systems. To ensure accurate and coherent results, the simulations conducted here use the same platform description as for the evaluation of the power model developed in our previous work [15, 17]: 4 compute nodes each with 2 hexacore processors. The simulator code and experimental scenarios used in the rest of this paper are all publicly available online [19].

Evaluation results. Fig. 1-*top* shows the simulated energy consumption of the schedules computed by the SPSS-EB and EnReal algorithms, as computed with both the *traditional* and the *realistic* models, for all Montage, SoyKB, and 1000Genome workflow application instances. Recall that both algorithms make scheduling decisions assuming that the traditional model holds in practice. Energy consumption does not necessarily increase monotonically with the number of workflow tasks due to irregular workflows structures. Comparing the "SPSS-EB/traditional" to the "EnReal/traditional" results would thus correspond to comparisons traditionally done in the literature. Instead, comparing the "SPSS-EB/realistic" to the "EnReal/realistic" results corresponds to a realistic comparison. We can see that, in some cases, results vary significantly. For instance, for the 364-task 1000Genome workflow, the traditional comparison gives a clear advantage to SPSS-EB, while the realistic comparison gives a larger advantage to EnReal. In total, such "reversals" are observed for 6 of the 9 Montage executions, none of the SoyKB executions, and 6 of the 15 1000Genome executions. When no reversals occur, the magnitude of the advantage of one algorithm over the other can be largely overestimated when assuming that the traditional model holds. For instance, consider the 1000Genome execution with 820 and 920 tasks. A traditional comparison would indicate that EnReal consumes marginally less energy than SPSS-EB, while a realistic comparison shows that, in fact, SPSS-EB consumes about twice as much energy as EnReal. Overall, the traditional model can lead to misleading results. We conclude that published results evaluating these and other energy-aware workflow scheduling algorithms do not allow for an accurate quantitative comparison of how algorithms would perform in practice, and in particular for I/O-intensive workflows.

The results in Fig. 1-*top* show smaller discrepancies between the traditional and the realistic models for EnReal than for SPSS-EB. We term the absolute difference between the energy consumption computed based on the two models the "error". Table 1 summarizes the results in Fig. 1 and shows the maximum error, the mean error, and the standard deviation of the error for both algorithms computed for all workflow instances for each application. The maximum error is up to three orders of magnitude for SPSS-EB, but only up to half an order of magnitude for EnReal. The mean error for SPSS-EB can also be much larger than that of EnReal, especially for the 1000Genome workflows. One may wonder how come, for some workflows, EnReal results are much less sensitive to the choice of the power consumption model. We analyzed the schedules computed by EnReal. When multiple options are possible for scheduling a task,

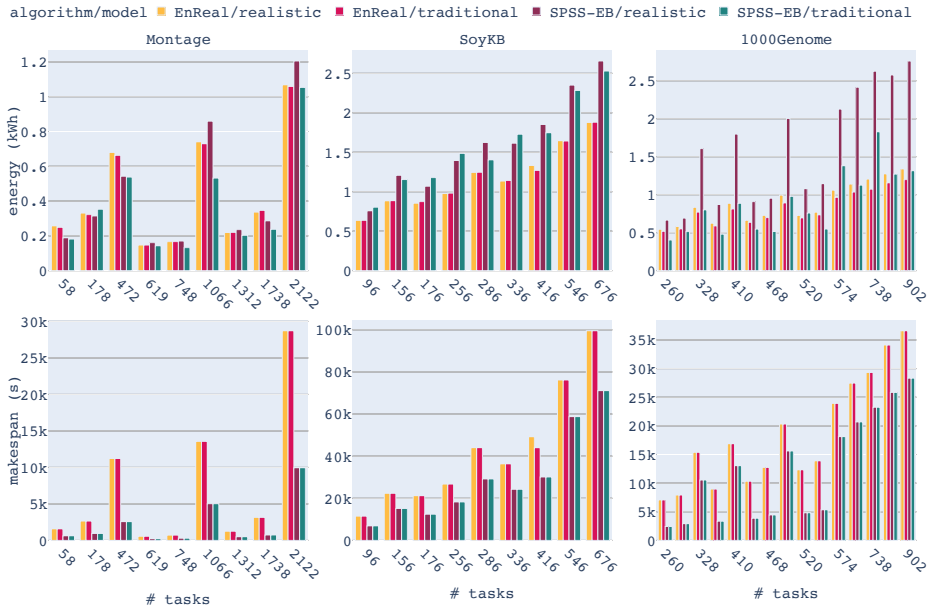

Fig. 1. Simulated workflow energy consumption (top) and makespan (bottom) for each workflow application instance for the SPSS-EB and EnReal algorithms. Results are shown when simulating energy consumption with the traditional power model and the realistic power model.

Workflow	Algorithm	Energy Error (KWh) Max		Mean		Stand. Deviation	
Montage	SPSS-EB	0.45	(160.71%)	0.09	(39.31%)	0.17	(52.07%)
	EnReal	0.22	(17.01%)	0.06	(7.81%)	0.01	(1.89%)
SoyKB	SPSS-EB	0.66	(33.76%)	0.13	(5.37%)	0.37	(15.72%)
	EnReal	0.12	(10.46%)	0.11	(9.17%)	0.02	(1.65%)
1000Genome	SPSS-EB	5.88	(360.57%)	1.95	(187.46%)	1.80	(120.50%)
	EnReal	0.35	(26.29%)	0.11	(14.65%)	0.10	(7.08%)

Table 1. Energy consumption error (maximum, mean, and standard deviation) for both algorithms and for each set of workflow instances.

EnReal balances the distribution of tasks among cores and computing nodes. As a result, it "involuntarily" also distributes the I/O load, which saves energy. By contrast, SPSS-EB favors early task completions, thus leading to I/O contention, which can translate to much higher energy consumption than would occur if the traditional model held in practice (e.g., for 8 of the 15 1000Genome executions).

When considering only the results obtained with the realistic model in Fig. 1-*top*, there are significant reductions in energy consumption for most cases when using the EnReal algorithm relative to using the SPSS-EB algorithm. However, these energy savings come at the cost of higher makespans. Makespan results are shown in Fig. 1-*bottom*. EnReal consistently leads to higher makespans than

SPSS-EB (on average higher by 4.35x for Montage, 1.71x for SoyKB, and 3.81x for 1000Genome). This is because EnReal fosters resource re-use. More precisely, and unlike SPSS-EB, it does not create a new VM instance if an already running instance will become idle in the near future.

The results in this section make it possible to evaluate and compare algorithms using our realistic power model, but these algorithms make their scheduling decisions based on the traditional model. In practice, they can make very suboptimal decisions, such as execute I/O-intensive tasks concurrently on the same compute node in an attempt to save energy. Such decisions can be particularly harmful for the overall energy consumption since the time waiting for I/O operations to complete, as seen in [15, 17], can significantly increase idle power consumption. In the next section, we investigate whether is possible to design a simple algorithm that makes good decisions based on the realistic model.

4 Energy-aware Scheduling of I/O-intensive Workflows

In this section, we present an energy-aware workflow scheduling algorithm that accounts for the energy cost of I/O by using the power consumption model described in Section 2.2 as a basis for making scheduling decisions. Our goal is to show that it is possible, and in fact straightforward, to design an algorithm that compares favorably to previously proposed algorithms that rely on the traditional power model.

4.1 I/O- and Energy-aware Scheduling

We propose IOBalance, an I/O- and energy-aware workflow scheduling algorithm that aims at minimizing energy consumption of I/O-intensive workflows by reducing I/O contention and data movement operations. Contention is lessened by distributing tasks that perform high number of I/O operations among available (running) nodes; data movement reduction is achieved by assigning tasks to nodes in which most of the tasks' input data are available (i.e., has been produced by a previous task).

Like SPSS-EB and EnReal, IOBalance computes a static schedule, deciding when to power hosts on and when to start VM instances. Tasks are scheduled in topological order, marking a task ready whenever all its parent tasks have been scheduled (initially all entry tasks are marked ready). Among all ready tasks to be scheduled, the algorithm first schedules the task with the highest volume of I/O operations, breaking ties based on the task's earliest start time (i.e., picking the task that can be started the earliest). Given a task to be scheduled with earliest start time t, IOBalance considers three options below in sequence:

1. If possible, schedule the task on a VM instance that is already scheduled to be running at time t or later, and that can complete the task before expiration. For each such VM instance the algorithm determines: (i) the energy cost of the task's execution (computed using the power consumption model in Eq. 3); and (ii) the earliest time at which the task could start on this VM

instance. The algorithm picks the VM instance with the lowest energy cost. If multiple instances have the same energy cost, then it picks the one that can start the task the earliest.
2. Otherwise, if possible, schedule a new VM instance to start at time t on a host that has already been scheduled to be powered on and will have at least one idle core at that time, and schedule the task on that instance at time t. If multiple such hosts exist, pick the host that already stores the largest amount of data needed by the task (so as to reduce data movements). VM instances on the same host are allocated to cores in round-robin fashion.
3. Otherwise, schedule a new host to be powered on at time t, schedule a new VM instance to be started on that host at time t, and schedule the task to execute on that VM instance at time t. If multiple such hosts exist, pick the host that already stores the largest amount of data needed by the task.

4.2 Experimental Evaluation

To evaluate the effectiveness of our algorithm and compare it to SPSS-EB and EnReal, we implemented it in the simulator used for the experiments in Section 3.3. Hereafter, we only show results for the realistic power consumption model. That is, the power consumed by the execution of the workflow on the compute platform is simulated based on the realistic model. However, recall that SPSS-EB and EnReal make scheduling decisions assuming the traditional model.

Fig. 2-*top* shows the simulated energy consumption of the schedules computed by the SPSS-EB, EnReal, and IOBalance algorithms, for all Montage, SoyKB, and 1000Genome workflow application instances. Overall, IOBalance saves significant energy when compared to SPSS-EB for all Montage and SoyKB workflow instances. Energy savings are up to 53% and on average 32% for Montage, up to 44% and on average 18% for SoyKB, and up to 64% and on average 36% for 1000Genome. When compared to EnReal, our proposed algorithm leads to schedules that consume more energy for most workflow instances. Specifically, consumed energy is higher than that of EnReal by up to 52% and on average 30% for SoyKB workflows, and up to 48% and on average 18% for 1000Genome workflows. For Montage workflows, however, IOBalance leads to lower energy consumption than EnReal for 6 of the 9 workflow instances, by up to 47%.

These energy results must be put in perspective with the makespan results shown in Fig. 2-*bottom*. For 30 of the 33 workflow instances IOBalance leads to makespan that is within 5% of the makespan achieved by SPSS-EB (the exceptions are the 58- and 178-task Montage workflows, and the 176-task SoyKB workflow). IOBalance schedules tasks in a way that reduces potential I/O contention, thus saving on energy, while rarely increasing the makespan when compared to SPSS-EB. For instance, the 1000Genome workflow is composed of a large number of I/O-intensive tasks that run for a couple of minutes and process high volumes of data (O(GB)) [16]. These tasks, if all scheduled on the same node (or a small number of nodes), suffer from local and remote I/O contention, which increases power consumption as defined by $\omega(t)$ in $P_{\text{I/O}}$ (Eq. 7). The distribution of I/O-intensive tasks among cores/nodes counterbalanced by

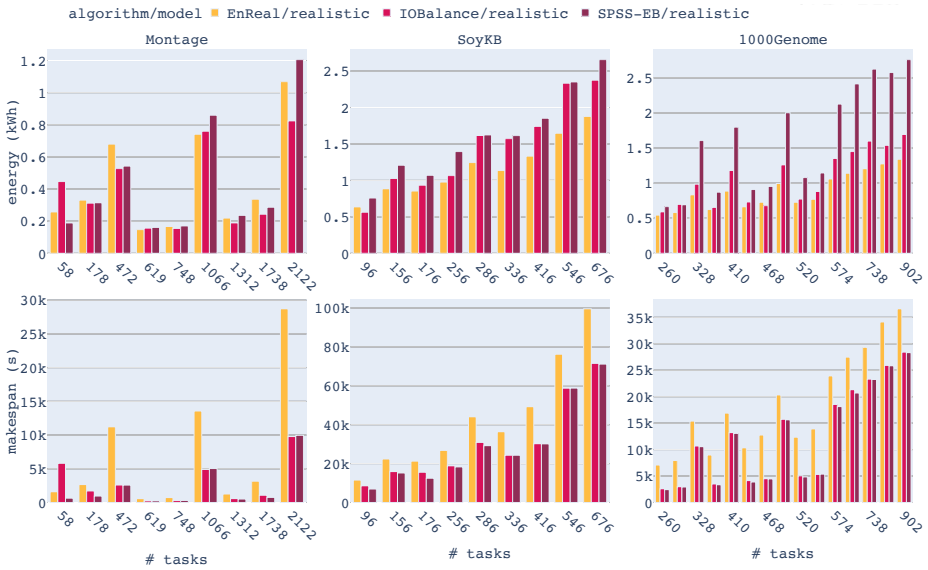

Fig. 2. Simulated workflow energy consumption (top) and makespan (bottom) for each workflow application instance for the SPSS-EB, EnReal, and IOBalance algorithms. Results are shown when simulating energy consumption with the realistic power model.

CPU-bound tasks prevents waiting for I/O operations to complete, which saves on energy by avoiding idle power consumption, but allows CPU-intensive tasks to benefit from the idle CPU cycles, which reduces makespan. IOBalance leads to makespan shorter, by up to 333% and on average by 94%, than EnReal for all workflow instances. So although EnReal schedules consume less energy, as noted when comparing SPSS-EB and EnReal in Section 3.3, this energy saving comes at the cost of 2x longer makespan on average.

We conclude that even a simple algorithm like IOBalance can improve on the state of the art because it makes scheduling decisions based on the realistic power consumption model. SPSS-EB and EnReal achieve different compromises between energy and makespan, with SPSS-EB consuming more energy but leading to shorter makespans and EnReal consuming less energy but leading to much longer makespans. IOBalance achieves a compromise that is strictly preferable to that achieved by SPSS-EB, saving significant energy while achieving similar makespans. For most workflow instances it leads to higher energy consumption than EnReal but achieves much shorter makespans.

5 Related Work

The need to manage energy consumption in large cloud data-centers has received significant attention in this past decade [9]. At the application and system levels, researchers have investigated techniques and algorithms to enable energy-

efficient executions. In the scientific workflows literature, a range of energy-aware workflow task scheduling or resource provisioning algorithms [5, 7, 10, 12, 13, 21] have been proposed. In [7], only dynamic energy consumption is considered and an algorithm composed of five sub-algorithms (VM selection, sequence tasks merging, parallel tasks merging, VM reuse, and task slacking algorithm) is proposed. Similarly, [12] also only considers dynamic energy consumption, but uses reinforcement learning for a more budget oriented analysis. Other works consider both static and dynamic energy consumption [5,10,13,21]. All these works make the strong assumption that power consumption is linearly correlated with CPU utilization and equally divided among virtual machines (or CPU cores within a compute node), and ignore power consumption due to I/O. The work in [15,17] shows that, at least for I/O-intensive workflow executions, these assumptions do not hold in practice, and proposes an accurate, if more complex, power consumption model. We use this model in this work.

Dynamic Voltage and Frequency Scaling (DVFS) is a well-known power management technique [7, 10, 12, 21]. It is used to decrease processor frequency to save energy and is often paired with slack time optimization, a method that takes advantage of idle slots resulting from early-completing tasks. Algorithms that implement this combined approach generally succeed in reducing power consumption (see the comparison to EnReal in [5]). This reduction comes at a sharp increase in workflow makespan. The algorithms considered in this work (the two algorithms in [11,22] and the algorithm proposed in Section 4.1) do not use DVFS, but could conceivably be augmented to do so.

6 Conclusion and Future Work

In this work, we evaluate two popular energy-aware workflow scheduling algorithms when used for executing I/O-intensive workflow applications. We quantify the energy consumption of the schedules they compute using both the traditional, but inaccurate, power consumption model used in the literature and the accurate power model developed in [15,17]. We show that comparing these algorithms under the traditional model leads to misleading results. Furthermore, as both algorithms make scheduling decisions based on the inaccurate traditional model, it is unclear how effective they can be in practice. For this reason, we propose a simple I/O-aware workflow scheduling algorithm that uses the accurate power consumption model, and compare this algorithm to the above two algorithms. Experimental results show that this algorithm achieves a strictly preferable tradeoff between makespan and energy than one of its two competitors. Although it often leads to higher energy consumption than the other competitor, it also achieves a 2x shorter makespan on average. In future work, we plan to broaden our analysis to consider the use of DVFS.

Acknowledgments. This work is funded by NSF contracts #1923539, #1923621, #2016610, and #2016619. Results presented in this paper were obtained using the Chameleon testbed supported by the National Science Foundation.

References

1. Casanova, H., et al.: Developing accurate and scalable simulators of production workflow management systems with wrench. Future Gener. Comp. Sy. **112** (2020)
2. Chameleon cloud. `https://chameleoncloud.org` (2021)
3. De Oliveira, D.C., et al.: Data-intensive workflow management: for clouds and data-intensive and scalable computing environments. Synthesis Lectures on Data Management **14**(4) (2019)
4. Deelman, E., et al.: Pegasus, a workflow management system for science automation. Future Generation Computer Systems **46**, 17–35 (2015)
5. Ghose, M., et al.: Energy efficient scheduling of scientific workflows in cloud environment. In: IEEE HPCC (2017)
6. Klimentov, A., et al.: Next generation workload management system for big data on heterogeneous distributed computing. In: J. Phys. Conf. Ser. vol. 608 (2015)
7. Li, Z., et al.: Cost and energy aware scheduling algorithm for scientific workflows with deadline constraint in clouds. IEEE Trans. Serv. Comput. **11**(4) (2018)
8. Ma, X., et al.: An iot-based task scheduling optimization scheme considering the deadline and cost-aware scientific workflow for cloud computing. EURASIP Journal on Wireless Communications and Networking **2019**(1), 1–19 (2019)
9. Orgerie, A.C., et al.: A survey on techniques for improving the energy efficiency of large-scale distributed systems. ACM Computing Surveys (CSUR) **46**(4) (2014)
10. Pietri, I., Sakellariou, R.: Energy-aware workflow scheduling using frequency scaling. In: International Conference on Parallel Processing Workshops (2014)
11. Pietri, I., et al.: Energy-constrained provisioning for scientific workflow ensembles. In: International Conference on Cloud and Green Computing (CGC) (2013)
12. Qin, Y., et al.: An energy-aware scheduling algorithm for budget-constrained scientific workflows based on multi-objective reinforcement learning. The Journal of Supercomputing **76**(1), 455–480 (2020)
13. Shepherd, D., et al.: Workflow scheduling on power constrained vms. In: IEEE/ACM 8th International Conference on Utility and Cloud Computing (2015)
14. Ferreira da Silva, R., et al.: A characterization of workflow management systems for extreme-scale applications. Future Generation Computer Systems **75** (2017)
15. Ferreira da Silva, R., et al.: Accurately simulating energy consumption of i/o-intensive scientific workflows. In: Computational Science – ICCS 2019 (2019)
16. Ferreira da Silva, R., et al.: Using simple pid-inspired controllers for online resilient resource management of distributed scientific workflows. Future Gener. Comp. Sy. **95** (2019)
17. Ferreira da Silva, R., et al.: Characterizing, modeling, and accurately simulating power and energy consumption of i/o-intensive scientific workflows. Journal of Computational Science **44**, 101157 (2020)
18. Ferreira da Silva, R., et al.: Workflowhub: Community framework for enabling scientific workflow research and development. In: IEEE WORKS Workshop (2020)
19. Energy-aware simulator. `https://github.com/wrench-project/energy-aware-simulator` (2021)
20. Wang, X., et al.: Delay-cost tradeoff for virtual machine migration in cloud data centers. Journal of Network and Computer Applications **78** (2017)
21. Wu, T., et al.: Soft error-aware energy-efficient task scheduling for workflow applications in DVFS-enabled cloud. Journal of Systems Architecture **84**, 12–27 (2018)
22. Xu, X., et al.: Enreal: An energy-aware resource allocation method for scientific workflow executions in cloud environment. IEEE Trans. Cloud Comput. **4**(2) (2015)