Energy-Constrained Provisioning for Scientific Workflow Ensembles

Ilia Pietri*, Maciej Malawski †,Gideon Juve ‡, Ewa Deelman ‡, Jarek Nabrzyski §, Rizos Sakellariou*
*University of Manchester, School of Computer Science, U.K
†AGH University of Science and Technology, Dept. of Computer Science, Poland
‡ USC Information Sciences Institute, USA
§ University of Notre Dame, Center for Research Computing, USA

***Abstract*—Large computational problems may often be modelled using multiple scientific workflows with similar structure. These workflows can be grouped into ensembles, which may be executed on distributed platforms such as the Cloud. In this paper, we focus on the provisioning of resources for scientific workflow ensembles and address the problem of meeting energy constraints along with either budget or deadline constraints. We propose and evaluate two energy-aware algorithms that can be used for resource provisioning and task scheduling. Experimental evaluation is based on simulations using synthetic data based on parameters of real scientific workflow applications. The results show that our proposed algorithms can meet constraints and minimize energy consumption without compromising the number of completed workflows in an ensemble.**

***Keywords*—cloud computing; energy efficiency; workflow scheduling; resource provisioning**

I. Introduction

Cloud computing offers a wide range of options to users to provision on-demand resources that meet the computational requirements of their applications. As users have more flexibility in controlling the execution of their applications by specifying QoS requirements and cost characteristics, many challenges arise from the provider's point of view whose goals may be conflicting. For example, minimizing energy consumption is an important goal from the provider's perspective. The amount of allocated resources affects the makespan of the applications and the cost incurred by the user, but at the same time it also affects the energy consumed by the cloud infrastructure. The provider may postpone the processing of some jobs in order to avoid low resource utilization or increased number of Virtual Machines (VMs) and hosts used, when user requirements, such as deadlines, can still be met. In other cases, more VMs can be used when host utilization is increased without requiring new hosts to be powered on, even increasing the cost incurred by the user (assuming that this is within budget). Overall, determining the optimal amount of resources in order to achieve a trade-off between the user requirements and the provider's optimization goals is a challenging problem.

Scientific workflows [1] are an important category of cloud computing applications under study, as the cloud computing infrastructure offers a dynamic environment to address the changing resource requirements of workflow-based applications. In many scientific problems workflows can be grouped into ensembles. In this paper, we address the problem of energy-efficient resource provisioning and task scheduling with energy constraints for ensembles of scientific workflows. In addition to energy, budget or deadline constraints are also taken into account. To the best of our knowledge, this is the first paper in the literature that describes algorithms to meet energy constraints along with budget or deadline when planning scientific workflow ensembles. More specifically, the contributions of this work are:

- The development of two algorithms, SPSS-EB and SPSS-ED, which can be used for scheduling workflows in ensembles so that energy and deadline or energy and budget constraints are met.
- An experimental evaluation of the performance of the algorithms and their behaviour in meeting constraints at the same time executing as many workflows from the ensemble as possible.

The rest of the paper is organized as follows. In Section 2 related work is presented. In Section 3 the problem and the assumptions made are described. The proposed algorithms are described in Section 4, while the experimental evaluation follows in Section 5. Finally, Section 6 concludes the paper.

II. Related Work

A significant amount of work has been done on task scheduling and provisioning of resources for workflows for both grid [2],[3],[4] and cloud infrastructures [5],[6],[7],[8]. Most of the proposed algorithms focus on optimizing application performance, such as reducing the application cost or makespan. Elastic provisioning of resources for workflows with deadline requirements is the focus in [6], which extends Balanced Time Scheduling (BTS), a heuristic algorithm that achieves the minimum resource capacity required in order to minimize application cost. Huu and Montagnat [7] propose cost-based algorithms for workflow-based applications aiming at minimizing the number of allocated resources and reducing application makespan. Applications with budget and deadline constraints are also the focus in [9]. A scalable environment for the user is provided by choosing proper VM instance types that allow jobs to be completed within the deadline and reduce application cost for a particular user request. Algorithms for workflow ensembles that meet budget and deadline constraints are proposed in [5]. This paper extends the work in [5] by including energy constraints. A multi-objective scheduling framework for workflows is proposed in [3] with energy consumption being one of the objectives. However, in their model the energy consumed by a machine is related to its computational speed.

Existing work in energy efficiency in clouds and clusters aims at improving load balancing [10] and server consolidation [11], reducing energy consumption using DVFS techniques [12] or focusing on thermal management [13]. In our work we focus on energy savings when planning the execution of scientific workflows in ensembles, a problem which has not been considered before.

III. Problem Description and Assumptions

In our model for energy constrained provisioning, the user has some knowledge of the energy required for the execution of their applications based on information by the provider and submits a workflow ensemble for execution specifying the energy constraints. Provisioning is made so that the total energy consumption stays within an energy budget limit. More specifically, a scheduling plan for the ensemble is developed to execute as many workflows as possible given the energy constraints. A workflow from the ensemble is accepted and added to the plan only when the total energy consumption will not exceed the available energy budget. Different scheduling schemes can require different amount of energy to be spent for the execution of the workflows. Note that energy consumption consists of static and dynamic energy [12]. The static energy is the energy consumed by the system resources when idle and the dynamic energy is the amount of energy consumed by the applications when running. The dynamic energy required by a workflow can be modelled depending on the resource needs of the tasks.

a) Application Model: The work focuses on workflow-based applications where each workflow consists of inter-related tasks and can be represented as a Directed Acyclic Graph (DAG) with the nodes being the tasks and the edges representing data dependencies between them. Hence, for two tasks i and j, the edge (i, j) shows that the task j can only be executed when the processing of i has finished. All workflows in the ensemble are submitted at the same time and scheduling aims at completing as many workflows from the ensemble as possible under the given constraints. A workflow is admitted only when all the tasks can be completed without exceeding the constraints. Finally, the time required for data transfer is not included in the estimation of task runtime, assuming that a shared cloud storage system such as Amazon S3 is used. Hence, task runtime is not affected by the placement decision on different VMs.

b) Cloud Resources: A model similar to Amazon's Elastic Compute Cloud (EC2) is assumed with VMs being provisioned on demand in a per-hour billing basis with partial hours being rounded to the full hour. We also assume that jobs do not run concurrently on a VM, but have exclusive access to it. Although cloud computing infrastructures offer different VM instance types to users with various characteristics including CPU, memory and disk sizes, a single VM instance type is used for simplicity, while the hosts are homogeneous with fixed capacity in VMs. Finally, static scheduling takes place and migration is not supported.

c) Model for Energy Consumption: Energy consumption is calculated as the integral of the consumed power [14]. The model to measure energy consumption is based on [15] where the power consumed by a host is linearly related with host utilization and is given by:

$$P_i = (P_{max} - P_{min}) * U_i + P_{min}$$

where U_i is the CPU utilization of host i at that time, P_{max} is the power consumed when the host is fully utilized and P_{min} is the minimum power consumed by an active host. Without loss of generality, P_{max} and P_{min} can be considered constant, like in [14], although they may vary in real systems depending on the application characteristics and resource needs. As the hosts are assumed to be homogeneous with a fixed capacity of VMs per host (n) and tasks have exclusive access to VMs consuming all the capacity, the fraction of the energy consumed by a running VM can be defined as:

$$E_{VM} = (P_{max} - P_{min}) * \frac{1}{n} * r$$

where r is the runtime of the VM. Other system resources, like memory and disk, consume energy, but in most of the current models energy consumption is mainly determined by the CPU [14]. Thus, the energy required for data transfer can be ignored. For each host the time that is required to be active for the execution of the workflows is determined by the period between the time the first task assigned to the host starts until the processing of the last task being scheduled to it finishes. It is assumed that hosts are exclusively used for the execution of the workflows in the ensemble for that period; hence, slots to which no tasks have been assigned remain idle. In order to increase host utilization the energy consumed for each host switching on/off can be taken into account. Many energy-efficient approaches that aim at minimizing the number of active hosts and reduce energy consumption also use power on/off operations to shut down under-loaded hosts [10], [11], [16]. In our work, host switching on/off operations are considered in the calculation of the total energy consumption by adding the energy consumption of booting and switching off each host used.

IV. Scheduling Algorithms

In this section two energy-aware algorithms, SPSS-EB (Static Provisioning-Static Scheduling under Energy and Budget Constraints) and SPSS-ED (Static Provisioning-Static Scheduling under Energy and Deadline Constraints) are proposed. The two algorithms take into account energy constraints to schedule the tasks and provision cloud resources for workflow ensembles in addition to budget or deadline constraints, respectively. Resource provisioning is planned in two phases: an ensemble planning algorithm, PlanEnsemble, creates a plan for the task scheduling of the workflows. For every workflow, PlanEnsemble calls the workflow planning algorithm, PlanWorkflow, to build a new plan on top of the current one for each workflow in the ensemble. The returned plan is accepted when the constraints are met; otherwise, it is rejected and the procedure continues with the next workflow in the ensemble. This two-phase approach was initially used in [5] to develop an algorithm that takes into account only budget and deadline constraints, which provided an inspiration for the current work.

A. SPSS under Energy and Budget Constraints (SPSS-EB)

The aim of SPSS-EB is to maximize the number of completed workflows under energy and budget constraints, by

Algorithm 1 SPSS-EB ensemble planning algorithm

Require: W: workflow ensemble, e: energy budget, b: budget
Ensure: : Schedule as many workflows of W as possible given e and b
1: **procedure** PLANENSEMBLE(W, e, b)
2: $P \leftarrow \emptyset$, $A \leftarrow \emptyset$ ▷ Current plan, set of admitted DAGs
3: **for** w in W in priority order **do**
4: $P' \leftarrow$ PlanWorkflow(w, P)
5: **if** EnergyConsumption(P') $\leq e$ && Cost(P') $\leq b$ **then**
6: $P \leftarrow P'$, $A \leftarrow A + w$ ▷ Accept plan, admit w
7: **end if**
8: **end for**
9: **return** P, A
10: **end procedure**

Algorithm 2 SPSS-EB workflow planning algorithm

Require: w: workflow, P: current plan
Ensure: Create plan for w that minimizes energy consumption
1: **procedure** PLANWORKFLOW(w, P)
2: $P' \leftarrow$ copy of P
3: **for** t in w in topological order **do**
4: $v \leftarrow$ VM that minimizes energy consumption and start time of t
5: **if** v already provisioned **then**
6: Schedule(t, v)
7: **else**
8: Provision a new VM v and schedule(t, v)
9: **end if**
10: **end for**
11: return P'
12: **end procedure**

Algorithm 3 basic SPSS-EB workflow planning algorithm

same as Alg.2, but line 4 changes as follows:
4: $v \leftarrow$ VM that minimizes application cost and start time of t

Algorithm 4 SPSS-ED ensemble planning algorithm

Require: W: workflow ensemble, e: energy budget, d: deadline
Ensure: Schedule as many workflows of W as possible given e and d
1: **procedure** PLANENSEMBLE(W, e, d)
2: $P \leftarrow \emptyset$, $A \leftarrow \emptyset$ ▷ Current plan, set of admitted DAGs
3: **for** w in W in priority order **do**
4: $P' \leftarrow$ PlanWorkflow(w, P, d)
5: **if** EnergyConsumption(P') $\leq e$ && Makespan(P') $\leq d$ **then**
6: $P \leftarrow P'$, $A \leftarrow A + w$ ▷ Accept new plan, admit w
7: **end if**
8: **end for**
9: **return** P, A
10: **end procedure**

Algorithm 5 SPSS-ED workflow planning algorithm

Require: w: workflow, P: current plan, d: deadline
Ensure: Create plan for w that minimizes energy consumption and meets deadline d
1: **procedure** PLANWORKFLOW(w, P, d)
2: $P' \leftarrow$ copy of P
3: DEADLINEDISTRIBUTION(w, d)
4: **for** t in w sorted by DL(t) **do**
5: $v \leftarrow$ VM that minimizes energy consumption and start time of t
6: **if** FinishTime(t, v) $<$ DL(t) **then**
7: Schedule(t, v)
8: **else**
9: Provision a new VM v and schedule(t, v)
10: **end if**
11: **end for**
12: **return** P'
13: **end procedure**

Algorithm 6 basic SPSS-ED workflow planning algorithm

same as Alg.5, but line 5 changes as follows:
5: $v \leftarrow$ VM that minimizes application cost and start time of t

minimizing energy consumption. In the ensemble planning algorithm (Alg.1) the procedure $PlanEnsemble$ calls the workflow planning algorithm (Alg.2) for each workflow in the ensemble to build a new plan on top of the current one. The new plan is accepted in case the energy and cost constraints are fulfilled and the workflow is admitted. Otherwise, the new plan is rejected and the algorithm continues to the next workflow. More specifically, in line 3 of Alg.1 the workflows are sorted in order to prioritize workflows that consume less energy. Sorting the workflows by the energy (the estimated dynamic energy) they consume and choosing the plan that minimizes energy consumption is a possible way to ensure that as many workflows from the ensemble as possible will be executed.

When planning a workflow (Alg.2), SPSS-EB schedules each task of the workflow so that the total energy consumed is the minimum. Hence, the slot that impacts the energy consumed the least is chosen. Between slots that affect energy consumption the same, VMs on running hosts are preferred to avoid booting a new host. This is because booting up a host is energy consuming. Otherwise, the slot with the earliest start is chosen, even if a new VM on a running host is needed to be switched on. The next criterion is choosing the slot with the lowest cost. Finally, existing resources are preferred.

In addition to the above algorithm, a basic version of SPSS-EB was used for comparison purposes. In this basic version we try to consider application cost first. The algorithm is the same as above, but workflows are not prioritized in terms of energy and the key difference is that the decision (line 4 in Alg. 3) simply prefers slots that minimize application cost. If two VMs have the same cost, existing VMs are preferred. Otherwise, slots with earlier start are preferred. This basic version can be considered as a baseline version that is based on a budget-deadline heuristic [5], which does not attempt to allocate for energy, but simply checks if the energy constraint is met.

B. SPSS under energy and deadline constraints (SPSS-ED)

The aim of SPSS-ED is to maximize the number of completed workflows under energy and deadline constraints, by minimizing energy consumption. In line 3 (Alg.4) the workflows are sorted in priority order depending on the energy consumption they require.

When planning a workflow (Alg.5), SPSS-ED schedules each task of the workflow so that the energy consumed is the minimum. Firstly, each task is given a sub-deadline, $DL(t)$, being a fraction of the slack time of the workflow, $ST(w)$, the difference between the critical path from the deadline, as in [5]. More specifically in line 4, the tasks are divided into levels according to the length of their longest path from the entry task of the workflow and a portion of the slack time of the workflow, $ST(l)$, is given to each level depending on the number of tasks, $N(l)$, and the total task runtime of the level, $RT(l)$:

$$ST(l) = ST(w)\left[\left(\alpha\frac{N(l)}{N(w)}\right) + \left((1-\alpha)\frac{RT(l)}{RT(w)}\right)\right]$$

Levels with many tasks are given a larger portion of the slack time ($\alpha = 0.7$) to avoid processing many tasks in parallel that would require a large amount of resources. The sub-deadline of the task is

$$DL(t) = LST(t) + RT(t) + ST(Level(t)),$$

where $LST(t)$ is determined by the sub-deadline of its predecessors:

$$LST(t) = \begin{cases} 0, & \text{if } Pred(t) = \emptyset \\ \max_{p \in Pred(t)} DL(p), & \text{otherwise.} \end{cases}$$

When the deadline is smaller than the critical path, the workflow cannot finish before the deadline and is rejected. For each task sorted by deadline, the decision (line 5) is made so that VMs that impact energy consumption at the minimum are chosen, with VMs on running hosts being preferred. In case two slots have the same effect on energy consumption, the slot with the earliest start is preferred in order to minimize application makespan if possible. Otherwise, the slot with the minimum cost is chosen so that application cost is reduced. The last criterion in the decision making is that existing VMs are preferred so that new VMs are not switched on.

Same as before, a basic version of SPSS-ED was used. This is the same as above, but the decision (line 5 in Alg.6) is made, as in [5], without taking into account energy consumption; slots that minimize application cost are preferred. If two VMs have the same cost, existing VMs are preferred. Otherwise, slots with earliest start are preferred.

V. Experimental Evaluation

A. Methodology

The simulator [17] that is based on CloudSim [18] was used in order to implement and evaluate the proposed algorithms. The simulator, also used in other studies [5], was modified in order to incorporate the model for the energy consumption and develop the proposed algorithms. Static VM allocation without migration is assumed for simplicity with hosts being powered on when a new VM has to be switched on and there is no space in the active hosts. Each host is assumed to be switched on when the execution of the first assigned task starts and it is switched off when the last task running on it finishes. Single-core VMs were used in the implementation so that only one task is running on it each time. One instance type of VMs is used with CPU capacity of 1 MIPs and hourly price of $1. Hosts are homogeneous with a capacity of 4 VMs. For the calculation of the consumed energy, the parameters used in the experiments were based on [19]: $P_{max} = 255W$, $P_{min} = 217W$, power for host booting-up $P_{boot} = 288W$ and switching-off $P_{switchoff} = 213W$ while it is assumed that 170 secs and 20 secs are needed to boot up and power off a host, respectively.

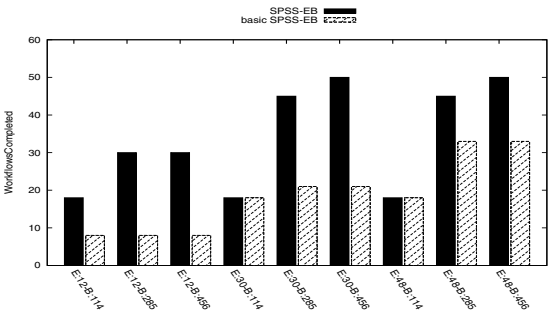
Data from three real scientific applications, namely LIGO [20], SIPHT [5], [20] and Montage [5] were used in order to assess the performance of the algorithms. LIGO is an example of a scientific workflow-based application, where gravitational waves in the universe are detected. LIGO reads and writes a significant amount of data while the most computationally intensive jobs consume most of its total runtime. One of the job categories can be executed in parallel [21]. SIPHT is an application used to search for sRNA encoding genes for bacterial replicons. SIPHT can be characterized as a CPU-intensive workflow with most of the jobs having high CPU utilization [21]. Finally, Montage is another case of workflow-based application that generates image mosaics of the sky, in which several jobs spend most of their time on I/O operations [21]. Synthetic data were generated using information from real scenarios to create the ensemble of the workflows. The code for the workflow generator can be found in [22]. An ensemble of 50 workflows was generated for each of these applications. Each workflow had a fixed size of 100 tasks.

For the experiments, three values were used for each of the three constraints (deadline D in sec, budget B in $, energy E in kWh) to create a total of 9 scenarios. In each case, the energy and budget or energy and deadline values represent requirements that must be met by the algorithms. The values used for each workflow were the following. In the case of LIGO: E is 12, 30 or 48 kWh, B is $114, 285 or 456 and D is 2573, 11579 or 20584 secs. In the case of SIPHT: E is 9, 23, 36 kWh, B: $83, 208, 332 and D: 7865, 35393, 62920 secs. In the case of Montage: E is 2, 5, or 8 kWh, B is $6, 15, or 24 and D is 139, 626, 1112 secs. The values used in each scenario are shown on the x-axis in the graphs that follow. For example, in Fig. 1, E:12 - B:114 represents the scenario in which the total energy consumption should not exceed 12kWh and the total cost should be less than $114. In Fig. 1b and 1d these constraints are met. In the case of energy and deadline constraints, the 9 scenarios represent combinations of energy and deadline requirements to be met. For example, in Fig. 4, E:12 - D:2573 represents the scenario in which the total energy consumption should be less than 12kWh and the deadline is 2573 secs. In Fig. 4b and 4c these constraints are met.

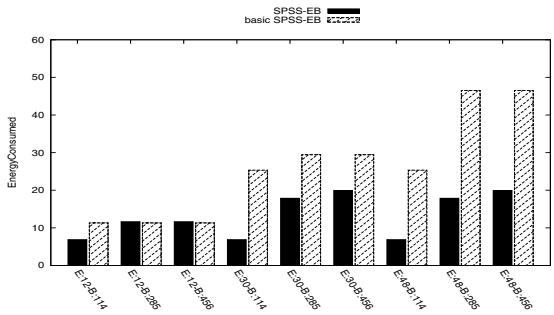
B. Results

Energy-Budget Constraints: The results for the performance of the algorithms for energy and budget constraints are presented in Fig.1, 2, 3 for LIGO, SIPHT and Montage, respectively. When the constraints are tight a smaller number of workflows is completed so that the energy and budget requirements of the ensemble can be met. SPSS-EB reduces significantly the total energy consumption in comparison with the basic version. This can be explained by the fact that the total makespan decreases while host utilization is increased by using all the available VMs. As more VMs in the host are used, the total cost incurred by the user is increased but remains within the budget limits. In the case of LIGO, the reduction in energy consumption is larger than in SIPHT and Montage, as the makespan of the ensemble is reduced more. This may be explained by the different characteristics and structure of the applications. Overall, SPSS-EB increases server utilization by exploiting all the VMs of the host to execute tasks in parallel, reducing total makespan and energy consumption. On the other hand, the basic algorithm uses a smaller number of VMs to reduce application cost resulting in longer makespans. However, it is less energy-efficient as it has low host utilization.

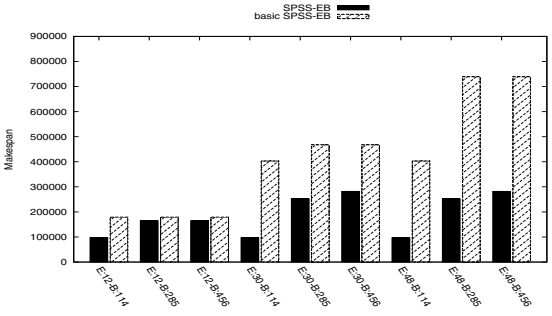
Energy-Deadline Constraints: In the case of energy and deadline constraints, SPSS-ED performs better than basic SPSS-ED as far as energy efficiency is concerned (Fig.4b, 5b, 6b). In the case of LIGO (Fig.4), the makespan of the ensemble is the same for both of the compared algorithms. Energy consumption is reduced in the case of the SPSS-ED algorithm by increasing host utilization and reducing the average number of hosts required over time. In the case of SIPHT (Fig.5),



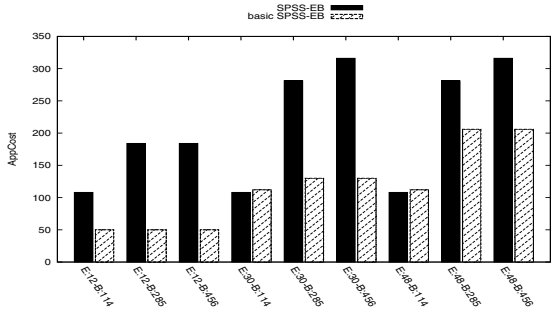
(a) WorkflowsCompleted



(b) Energy

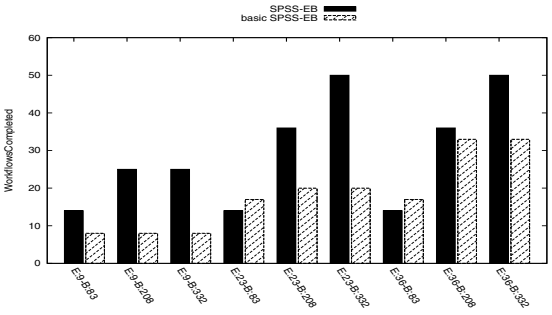


(c) Makespan

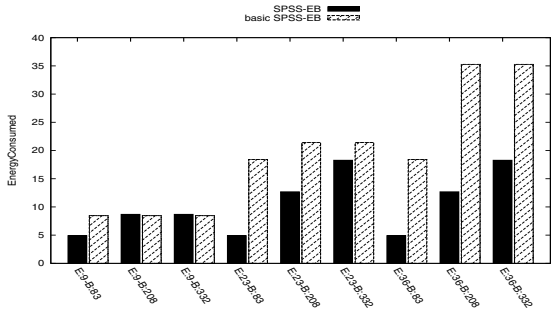


(d) AppCost

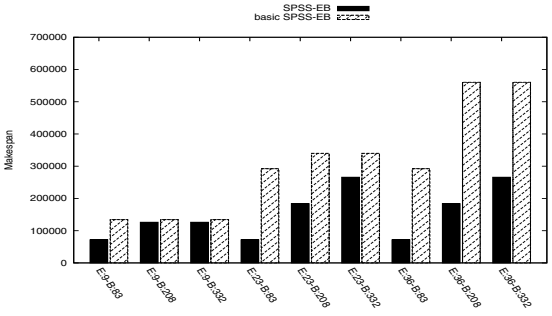
Fig. 1: Results for LIGO under energy and budget constraints



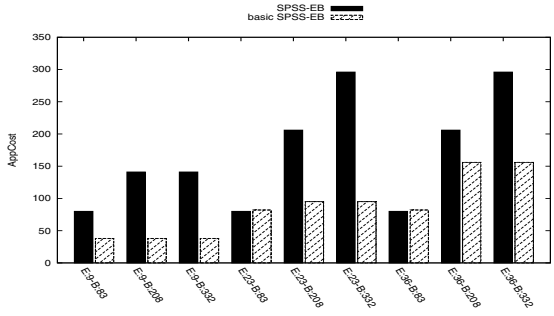
(a) WorkflowsCompleted



(b) Energy

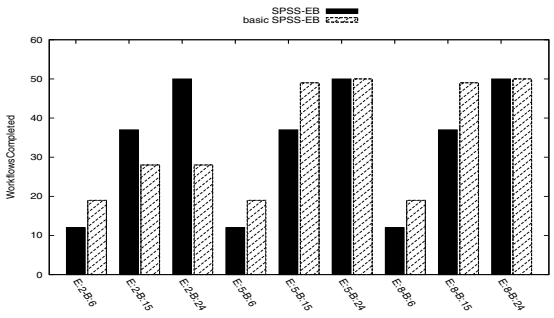


(c) Makespan

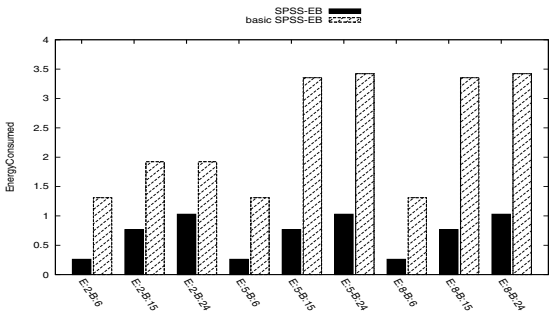


(d) AppCost

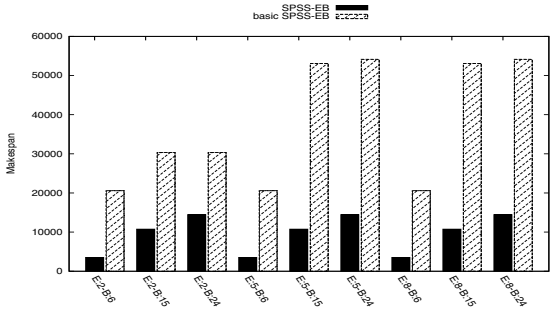
Fig. 2: Results for SIPHT under energy and budget constraints



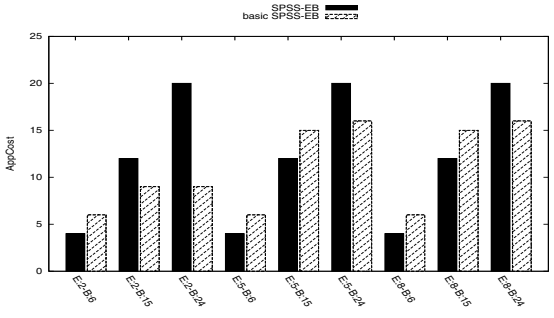
(a) WorkflowsCompleted



(b) Energy

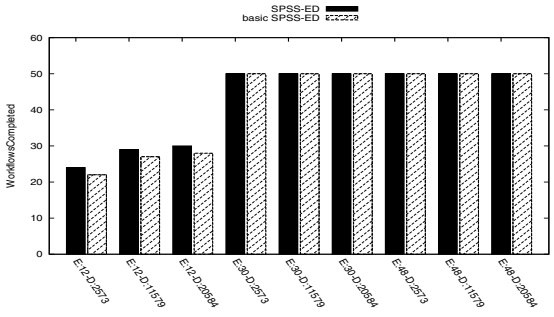


(c) Makespan

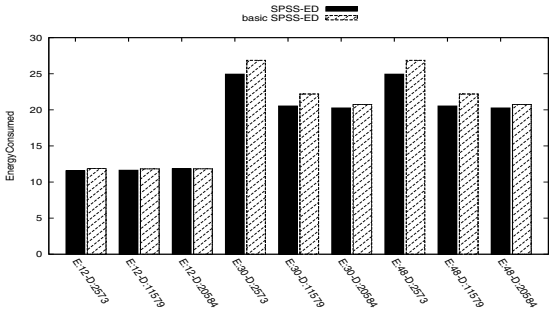


(d) AppCost

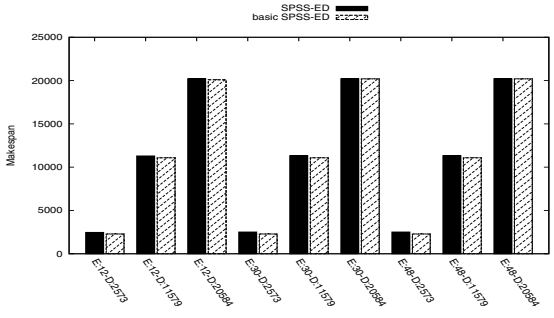
Fig. 3: Results for MONTAGE under energy and budget constraints



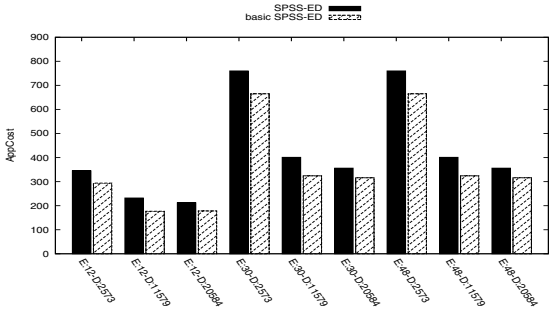
(a) WorkflowsCompleted



(b) Energy

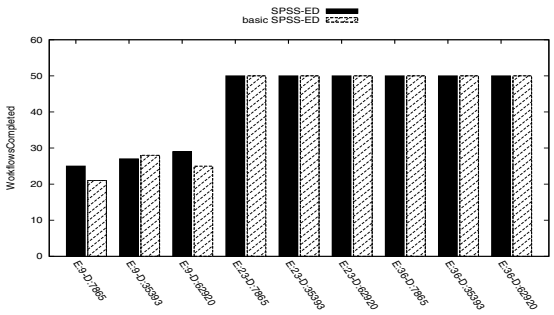


(c) Makespan

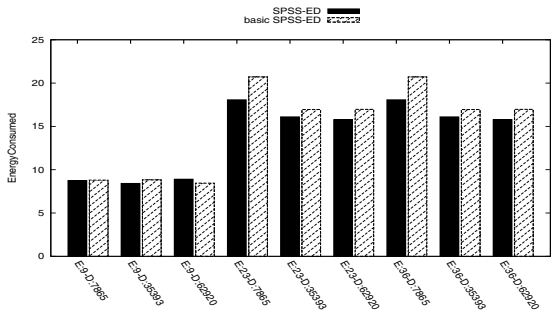


(d) AppCost

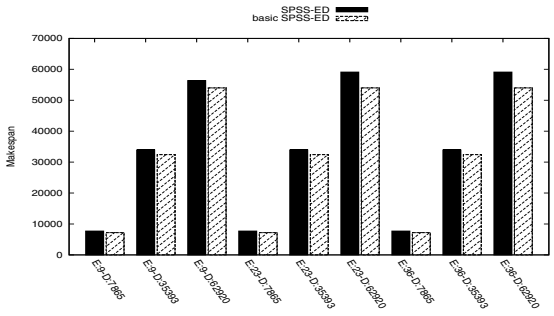
Fig. 4: Results for LIGO under energy and deadline constraints



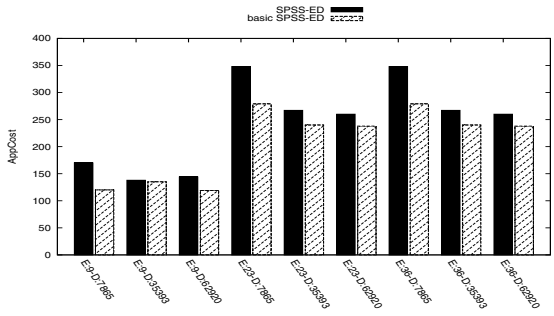
(a) WorkflowsCompleted



(b) Energy

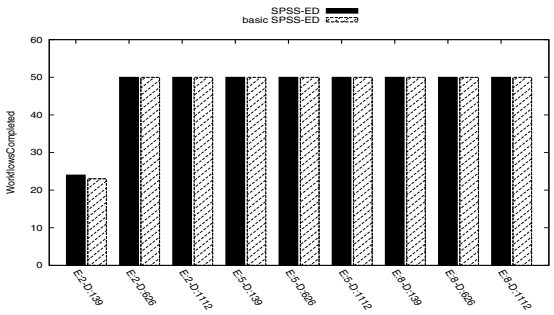


(c) Makespan

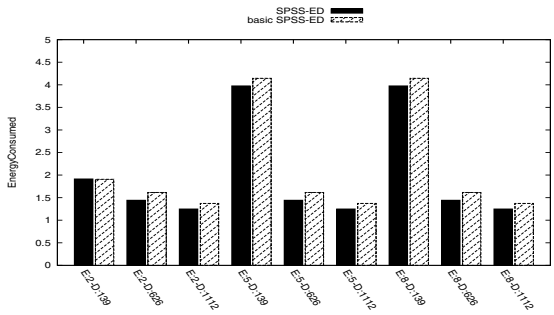


(d) AppCost

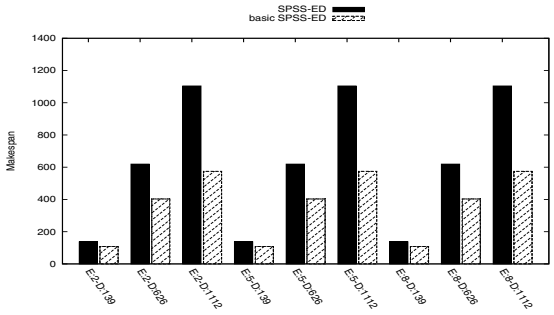
Fig. 5: Results for SIPHT under energy and deadline constraints



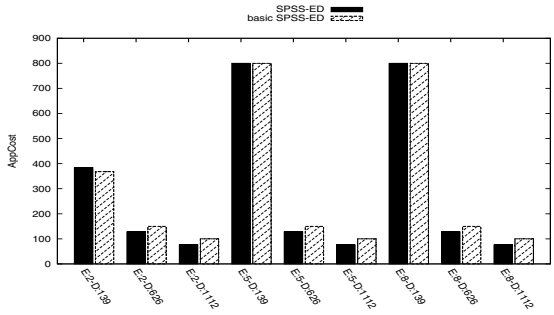
(a) WorkflowsCompleted



(b) Energy



(c) Makespan



(d) AppCost

Fig. 6: Results for MONTAGE under energy and deadline constraints

SPSS-ED exploits the spare time in order to reduce the average number of hosts used. Higher utilization of the hosts explains better the results. In the case of Montage (Fig.6), SPSS-ED reduces the average number of hosts needed for the execution of the workflows. However, the makespan of the ensemble is significantly increased, as shown in Fig.6c, leading to low energy savings. The changes in the makespan and the number of hosts used explain why the compared algorithms lead to similar total application cost (Fig.6d).

C. Summary of Observations

With energy and budget constrained provisioning, a cost-based approach uses a small number of VMs so that all the nodes are fully utilized for the hours the user is charged. On the other hand, in an energy efficient approach, the planning is made so that the servers utilize all their available VMs for the time they are required. In both approaches, booting a large number of VMs to be used is avoided. Another interesting observation is that pricing schemes can be improved to trade-off between energy and cost, as high energy savings can be achieved at the expense of the higher cost. In the case of energy and deadline provisioning, the parallelism of tasks can be exploited to reduce the achieved makespan and meet the deadline. However, when it is not required to execute a large number of tasks in parallel in order to meet the deadline, the execution of some tasks can be serialized to avoid using a large number of VMs and wasting energy when the hosts are not fully utilized. Assigning sub-deadlines to tasks allows to make the resource planning so that energy consumption is reduced without exceeding the deadline. Finally, planning the provisioning of resources for the ensemble and not each workflow independently allows better resource utilization by filling idle slots with tasks of other workflows.

VI. Conclusions

In this paper we considered the problem of resource planning under energy constraints for workflow ensembles with budget or deadline requirements. Two algorithms, SPSS-EB and SPSS-ED, were proposed to deal with energy constraints along with budget or deadline constraints and take into account energy consumption in the decision making. The performance of the algorithms was evaluated based on simulation. The results show that the proposed algorithms can achieve energy savings when compared with the cost-based schemes, by reducing the average number of hosts required over time and increasing host utilization. Depending on the application, the total cost incurred by the user may be increased when compared to the basic version, but remains within the limits in the case of applications with budget constraints. Also, the goal of maximizing the number of completed workflows in the ensemble under the given constraints is achieved. Future work can investigate the effect of: workflow structures, provisioning delays and data transfer costs on the performance of the algorithms. Finally, extending the current approach for heterogeneous hosts with different VM instance types is another future direction.

References

[1] E. Deelman, D. Gannon, M. Shields, and I. Taylor, "Workflows and e-science: An overview of workflow system features and capabilities," *Future Generation Computer Systems*, vol. 25, no. 5, pp. 528–540, 2009.

[2] S. Abrishami, M. Naghibzadeh, and D. Epema, "Cost-driven Scheduling of Grid Workflows Using Partial Critical Paths," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 8, pp. 1400–1414, 2012.

[3] H. M. Fard, R. Prodan, J. J. D. Barrionuevo, and T. Fahringer, "A Multi-objective Approach for Workflow Scheduling in Heterogeneous Environments," in *12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, 2012, pp. 300–309.

[4] R. Prodan and M. Wieczorek, "Bi-Criteria Scheduling of Scientific Grid Workflows," *IEEE Transactions on Automation Science and Engineering*, vol. 7, no. 2, pp. 364–376, 2010.

[5] M. Malawski, G. Juve, E. Deelman, and J. Nabrzyski, "Cost- and Deadline-Constrained Provisioning for Scientific Workflow Ensembles in IaaS Clouds," in *IEEE Supercomputing*, 2012, pp. 10–16.

[6] E.-K. Byun, Y.-S. Kee, J.-S. Kim, and S. Maeng, "Cost optimized provisioning of elastic resources for application workflows," *Future Generation Computer Systems*, vol. 27, no. 8, pp. 1011–1026, 2011.

[7] T. T. Huu and J. Montagnat, "Virtual Resources Allocation for Workflow-Based Applications Distribution on a Cloud Infrastructure," in *10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, 2010, pp. 612–617.

[8] D. Yuan, Y. Yang, X. Liu, and J. Chen, "A data placement strategy in scientific cloud workflows," *Future Generation Computer Systems*, vol. 26, no. 8, pp. 1200–1214, 2010.

[9] M. Mao, J. Li, and M. Humphrey, "Cloud auto-scaling with deadline and budget constraints," in *11th IEEE/ACM International Conference on Grid Computing*, 2010, pp. 41–48.

[10] L. Lefèvre and A.-C. Orgerie, "Designing and evaluating an energy efficient cloud," *The Journal of Supercomputing*, vol. 51, no. 3, pp. 352–373, 2010.

[11] C. Mastroianni, M. Meo, and G. Papuzzo, "Self-economy in Cloud Data Centers: Statistical Assignment and Migration of Virtual Machines," in *Euro-Par*, 2011, pp. 407–418.

[12] K. H. Kim, A. Beloglazov, and R. Buyya, "Power-aware Provisioning of Cloud Resources for Real-time Services," in *7th International Workshop on Middleware for Grids, Clouds and e-Science*, 2009, pp. 1–6.

[13] T. Wood, P. Shenoy, A. Venkataramani, and M. Yousif, "Sandpiper: Black-box and gray-box resource management for virtual machines," *Computer Networks*, vol. 53, no. 17, pp. 2923–2938, 2009.

[14] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing," *Future Generation Computer Systems*, vol. 28, no. 5, pp. 755–768, 2012.

[15] Y. C. Lee and A. Y. Zomaya, "Energy efficient utilization of resources in cloud computing systems," *The Journal of Supercomputing*, vol. 60, no. 2, pp. 268–280, 2012.

[16] A. Verma, P. Ahuja, and A. Neogi, "Power-aware dynamic placement of HPC applications," in *22nd International Conference on Supercomputing*, 2008, pp. 175–184.

[17] Cloud Workflow Simulator, https://github.com/malawski/cloudworkflowsimulator.

[18] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and Experience*, vol. 41, no. 1, pp. 23–50, 2011.

[19] D. Kusic, J. O. Kephart, J. E. Hanson, N. Kandasamy, and G. Jiang, "Power and Performance Management of Virtualized Computing Environments Via Lookahead Control," *Cluster Computing*, vol. 12, no. 1, pp. 1–15, 2009.

[20] S. Bharathi, A. Chervenak, E. Deelman, G. Mehta, M.-H. Su, and K. Vahi, "Characterization of Scientific Workflows," in *3rd Workshop on Workflows in Support of Large-Scale Science*, 2008, pp. 1–10.

[21] G. Juve, A. Chervenak, E. Deelman, S. Bharathi, G. Mehta, and K. Vahi, "Characterizing and profiling scientific workflows," *Future Generation Computer Systems*, vol. 29, no. 3, pp. 682–692, 2013.

[22] Workflow Generator, https://confluence.pegasus.isi.edu/display/pegasus/WorkflowGenerator.