Budget-aware performance optimization of workows in multiple data center clouds

Karima Oukfif, Fares Battou, Samia Bouzefrane

Budget-aware performance optimization of workflows in multiple data center clouds

Karima Oukfif ✉[1][0000−0003−4912−674X], Fares Battou[2], and Samia Bouzefrane[3][0000−0002−0979−1289]

[1] LARI Lab, Mouloud Mammeri University of Tizi-Ouzou, Tizi-Ouzou, Algeria
karima.oukfif@gmail.com
[2] University of Lille, France
[3] CEDRIC Lab, Conservatoire National des Arts et Metiers, Paris, France

Abstract. Users pay to use resources in cloud systems which makes them more demanding on performance and costs. Optimizing the response time of the applications and meeting user's budget needs are therefore critical requirements when scheduling applications.
The approach presented in this work is a scheduling based-HEFT algorithm, which aims to optimize the makespan of tasks workflow that is constrained by the budget. For this, we propose a new budget distribution strategy named Estimated task budget that we integrate in our budget-aware HEFT algorithm. We use a multiple datacenters cloud as a real platform model, where data transfer costs are considered. The results obtained by our algorithm relative to recent work, show an improvement of makespan in the case of a restricted budget, without exceeding the given budget.

Keywords: Workflow · Multiple data centers cloud · Makespan Optimization · Budget distribution · HEFT.

1 Introduction

Cloud platforms have become the trend for running applications. The cloud-computing paradigm has revolutionized the way of assessing computing resources by proposing a highly versatile availability of resources through a pay-as-you-go model. These features have enabled users to migrate their applications to cloud platforms. Workflows are examples of scientific applications which involve a higher performance-computing environment to be executed and cloud platforms offer huge opportunities to solve them. A workflow is a popular model for representing scientific computing, including complicated simulation and precise analysis of massive data [10]. Scheduling workflows, considered as NP-hard [4] problem, still remains a fundamental issue in cloud computing although it has been widely studied over the years [11,12].

During workflows scheduling in the cloud, both cloud providers and users are most involved with the makespan and monetary costs criteria. Makespan refers to the completion time of the entire workflow and the price that users need to pay due to the usage of cloud resources is the monetary cost. In cloud computing, resources of different capabilities at different prices are provided. Normally, faster computing resources are more expensive than slower ones. Thus, different scheduling strategies of a workflow using different resources may result in different makespan and different monetary cost. Therefore, the problem of workflow scheduling in the cloud requires both time and cost constraints to be satisfied [16].

These two criteria have conflicting objectives, and it is important to suggest a trade-off between them. This trade-off influences the different scheduling objectives, which include reducing costs while meeting the deadline, optimizing makespan while meeting the budget, or achieving the deadline and budget as a more flexible objective [10].

In this work, we focus on optimizing makespan while meeting the budget constraint. The intuition behind this strategy is to finish a workflow at a given budget as soon as possible. The objective is to minimize makespan under budget restrictions. Several authors have worked on this issue in single data center clouds, as detailed in the related work section. In this work, we are dealing with the same problem in the context of multiple data centers clouds. Indeed, currently cloud providers have tens of thousands of servers for providing sufficient computing resources for applications. These resources are deployed in multiple data centers.

We propose a budget-aware HEFT algorithm for performance optimization of workflows in IaaS multiple data centers cloud. Our goal is to optimize the makespan and meeting the budget while scheduling workflows. For this, we propose a novel budget distribution approach based on estimating task features.

The rest of this paper is organized as follows. Section 2 gives an overview of the related work, followed by our budget-aware scheduling algorithm formalization in sections 3. The proposed scheduling algorithm based on HEFT is outlined and evaluated in Sections 4 and 5 respectively. Finally, Section 6 summarizes the results and concludes the paper.

2 Related Work

It is generally accepted that the problem of workflows scheduling upon distributed systems is NP-hard [4]. In this mind, heuristic and meta-heuristic strategies are used to generate high-quality and approximate solutions with polynomial time complexity. Makespan and cost still remain the most relevant criteria to optimize while scheduling workflows in clouds. Several approaches have been proposed to optimize makespan, cost or both.

Work such as [15] and [8] proposed budget and deadline constrained heuristics based upon Heterogeneous Earliest Finish Time (HEFT) to schedule workflow over cloud resources. The proposed heuristics present a beneficial trade-off between execution time and execution cost under given constraints.

A list multiobjective optimization technique is designed by [9] to minimize monetary costs for deadline constrained workflows in cloud environments. The authors select the non-dominated solutions by combining the quick non-dominated sorting approach with the crowding distance.

In [2, 18], the authors focus on budget and deadline aware workflow scheduling. Their idea presented in [2] is to satisfy both budget and deadline constraints while introducing a tunable cost-time trade off over heterogeneous instances. In [18] a workflow scheduling algorithm to find a feasible solution for a workflow that meets budget and deadline constraints is proposed. Based on the execution cost of the task on the slowest resources and the optimistic spare budget, the algorithm generates the task optimistic available budget. Then, it builds the set of suitable resources according to the task's optimistic available budget to control the range of resources selection, and thus controls the task execution cost.

Many studies have addressed the issue of scheduling workflows effectively given budget constraints. In order to achieve this goal, many works apply budget distribution strategies for workflow scheduling in the cloud. A budget distribution strategy consists of assigning a sub-budget for each task of the workflow.

The algorithm presented by [17] is an extension of the HEFT heuristic except for the selection phase (task-ressource assignment). The algorithm distributes the budget to all tasks in proportion to their average execution time on the available resources. Then, the resource chosen is the one that allows the task to be accomplished early at a cost not greater than its allocated sub-budget.

The authors in [1] propose a workflow partitioning that focuses on the dependency structure inherent to workflow tasks. The partitioning leads to several levels each containing independent tasks. Several methods are introduced to distribute the global budget over these levels. According to the authors, the most effective strategy is the so-called 'All-in' which places the entire budget on the entry-level. Thus any remainders are trickled down to later levels. In our work, we implemented this strategy for comparison purposes.

In [13] authors propose a scheduling algorithm whose objective is to optimize a workflow's execution time under a budget constraint. Their approach focuses on finer-grained pricing schemes that provide users with more flexibility and the ability to reduce the inherent wastage that results from coarser-grained ones. Their approach partitions the DAG into a bag of tasks prior to its execution and then distributes the budget over the tasks. The drawback of their clustering phase is the parallelism limitation because of grouping several dependent tasks of workflow at the same level, which leads to the performance degradation of the workflow.

The authors in [5] extended the two well-known algorithms, MIN-MIN [3] and HEFT [14] to budget-aware scheduling algorithms for scientific workflows with stochastic task weights on heterogeneous. In addition, they improved these versions with refined ones that aim to re-schedule some tasks on faster VMs.

In [6], the authors proposed a normalization-based budget for constraint workflow scheduling algorithm. Their algorithm controls the resource selection phase of each task according to the available budget, thereby increasing the probability of the 'best' resource selection.

All of these studies deal with the workfow scheduling problem in a single data center cloud and eliminate or underestimate data transfer costs, unlike today's cloud providers that rely on multiple data centers. The cost model used in previous studies did not consider the charge of data transfers in the cloud although most cloud providers charge for the actual data storage depending on the amount of data being stored. Moreover, we propose a new way of distributing the budget over the workflow tasks defined based on estimating their characteristics.

In this paper, we assume a multiple data centers cloud platforms that provide for workflow applications heterogeneous resources under budget and we consider a charged peer-to-peer transfer mode instead of resorting external global storage.

3 Problem Formalization

This section begins by detailing the application and the platform models, then we formalize the workflow scheduling problem. The aim of our scheduling heuristic is to find a schedule S for the workflow (w) with a minimum makespan while respecting the budget.

3.1 Application model

A scientific workflow application (w) is modeled as a DAG (Directed Acyclic Graph): $G = (V, E)$, where V is a set of n vertices representing tasks t_i ($1 \leqslant i \leqslant n$), and E is a set of directed edges. An edge $e(i, j) \in E$ corresponds to a dependence constraint between task t_i and t_j, in which t_i is an immediate parent task of t_j, and t_j the immediate child task of t_i. A child task cannot be executed

until all of its parent tasks are completed. A task with no parent tasks is called an entry task and a task with no children tasks is called an exit task. The data matrix, with $n \times n$ dimensions, represents the data volume exchanged between tasks.

3.2 Cloud resource model

We consider platforms defined by an IaaS cloud where computer resources (instances) can be deployed on different data centers. During this section, we use the time model that specifies the data transfer times between data centers that we detailed in our previous work [12]. Then we present a cost model that takes into account the costs of the instances as well as those of the transferred data.

Time model In IaaS clouds, computer resources are instantiated as Virtual Machines (VMs) which are deployed on different data centers. Typically, cloud providers offer multiple types of VM; $VM = vm_1, vm_2, ..., vm_q$ is the set of VMs with heterogenous processing capacity.

The estimated execution time of the task t_i in a VM of type vm_p is defined by the computing time ($CT(t_i, vm_p)$).

The transfer time $TT_{(i,j)}$ (see equation (1)) taken to transfer data from task t_i (executed on vm_p lodged in data center DC_a) to task t_j (executed on vm_k lodged in data center DC_b) corresponds to an edge $(i, j) \in E$ in the application graph (DAG).

$$TT_{(i,j)} = \frac{data_{ij}}{Transfer_{rate_{(p,k)}}} \tag{1}$$

The transfer time $TT_{(i,j)}$ is proportional to the size of the output data $data_{ij}$ produced by task t_i and transferred to t_j, and is inversely proportional to the heterogeneous transfer rates or the bandwidths $Transfer_{rate_{(p,k)}}$ when the tasks are executed on different data centers.

We note that when two communicated tasks are executed on the same VM, the transfer time is equal to zero and when they are executed on different VMs that are lodged in the same data center also the transfer time is neglected. Thereby, the total computing time $TCT(t_i, vm_p)$ of a task in a VM is computed as shown in Equation (2).

In this equation, m refers to the number of parent tasks of t_i (predecessors). The boolean s_m is equal to 0 if t_i and t_j run on the same virtual machine ($p = k$) or to 1 otherwise. Besides, the value r_m is equal to 1 to indicate that the VMs can be hosted in different datacenters.

$$TCT(t_i, vm_p) = CT(t_i, vm_p) + \sum_{j=1}^{m} s_m.r_m.TT_{(j,i)} \tag{2}$$

In the formula (2) the product $s_m \times r_m$ indicates the inclusion of data transfer times between the VMs according to the following two cases:

1. Transfer time between VMs in the same datacenter is neglected:

$$r_m = \begin{cases} 1 \text{ if } DC_a \neq DC_b \\ 0, \text{ else} \end{cases}$$

2. Transfer time between VMs in the same datacenter is not neglected:

$r_m = 1$, then $s_m \times r_m = s_m$

In order to calculate the makespan, which refers to the total execution time $TCT(w)$ of all the tasks of the workflow, it is necessary to define for each task these two attributes $EST(t_i, vm_p)$ (*Earliest Start Time*) and $EFT(t_i, vm_p)$ (*Earliest Finish Time*) as following:

The $EST(t_i, vm_p)$ of the task t_i on the machine vm_p, is calculated according to the following recursive procedure: If the task has no parent tasks, then $EST(t_i, vm_p) = 0$, otherwise the task can start executing as soon as its parent tasks are finished and their output data is transferred. However, if the resource is occupied by another task at this point, the execution of the task t_i should be delayed until this virtual machine vm_p is free again.

The procedure (3) calculates the value of $EST(t_i, vm_p)$:

$$EST(t_i, vm_p) = \begin{cases} 0, \text{if } t_i \text{ has no parent tasks} \\ \max\{avail[vm_p], \max_{t_k \in pred(t_i)}\{EFT(t_k, vm_q)\}\}, \text{ else} \end{cases} \quad (3)$$

Where: $pred(t_i)$ is the set of immediate predecessors of the task t_i (parents of t_i), $avail[vm_p]$ is the next instant time when the resource vm_p will be ready or available for task execution.

To determine $EST(t_i, vm_p)$, the maximum value between $avail[vm_p$ and $\max_{t_k \in pred(t_i)}\{EFT(t_k, vm_q)\}$ is selected, where $avail[vm_p]$ is the maximum completion time of previous tasks of t_i on the same virtual machine vm_p. The value $avail[vm_p]$ guarantees that a virtual machine processes one task at a time, while $\max_{t_k \in pred(t_i)}\{EFT(t_k, vm_q)\}$ guarantees that a child task starts after all of its parents have finished their execution.

The $EFT(t_i, vm_p)$ of the task t_i on the machine vm_p is given by the following equation:

$$EFT(t_i, vm_p) = EST(t_i, vm_p) + TCT(t_i, vm_p) \quad (4)$$

The makespan or the total computing time $TCT(w)$ of the workflow is defined by the equation (5), and which corresponds to the completion time of the last task in the workflow.

$$TCT(w) = \max\{EFT(t_i, vm_p)\} \quad (5)$$

Cost model To deal with the pay-as-you-go cost model of the cloud, we need to incorporate the budget constraint in our approach.

The total cost of the whole workflow execution is the sum of the cost due to the execution of all its tasks in the platform.

The cost due to the execution of a task on a given VM vm_p is defined by the equation (6) and the cost of data transfers for this task is given by the equation (7).

$$Cost(t_i, vm_p) = CT(t_i, vm_p) \times UC_{vm_p} \quad (6)$$

$$Trcost(t_i) = \sum_{i=j}^{m} data_{ij} \times r_m \times UC_{data} \quad (7)$$

Where UC_{vm_p} is the per time unit cost for using the vm_p, UC_{data} is the unit cost for transferring data and m the number of successor tasks of t_i. In the equation (7), $data_{ij}$ refers to the data

transferred from t_i to t_j. The boolean r_m indicates if the data transfers are achieved in the same datacenter ($r_m = 0$) thereby the data transfers fees are eliminated, or within different ones ($r_m = 1$). The cost for a task is then given by the equation (8).

$$Cost(t_i) = Cost(t_i, vm_p) + Trcost(t_i) \tag{8}$$

Altogether, we give the total cost for the workflow execution as (9):

$$Cost(w) = \sum_{i=1}^{n} Cost(t_i) \tag{9}$$

Objective Given a budget B, our objective is to find the schedule that minimizes the makespan while the budget is respected, namely: $\min\{TCT(w)\}$ while $Cost(w) \leq B$.

4 Budget-aware HEFT Based Scheduling algorithm

In this section, we will first describe the HEFT algorithm, then we propose a new budget distribution approach named Estimated task budget. Afterward, we present the Budget-aware HEFT heuristic for scheduling workflows with budget constraints in multiple datacenter cloud. HEFT heuristic is one of the most popular list-based scheduling algorithms. It determines the scheduling of a DAG in a heterogeneous environment in order to minimize the makespan of the application. HEFT is integrated into important projects like ASKALON project [7] to provide scheduling for a quantum chemistry application.

4.1 HEFT Algorithm

HEFT is a well-established list scheduling algorithm that prioritizes a workflow task with a higher rank. It determines the rank value for each task, based on the average execution time and the average communication time between the resources of two successive tasks.

The HEFT algorithm orders the tasks on the different resources in a given order based on a value of a rank attribute. Ranks in HEFT are calculated based on the estimated average computation and communication times of the tasks. Indeed, the tasks are sorted according to their scheduling priorities, which are based on the increasing rank (upward rank, noted $rank_u$). The value $rank_u$ is the maximum distance of a task from the DAG output task. The rank of a task is defined recursively as follows:

$$rank_u(t_i) = \begin{cases} \overline{CT}(t_{exit}), & \text{if } t_i = t_{exit} \\ \overline{CT}(t_i) + \max_{t_j \in succ(t_i)}\{\overline{TT}_{(i,j)}, rank_u(t_j)\}, & \text{else} \end{cases} \tag{10}$$

Where:

- $succ(t_i)$ is the set of immediate successors of task t_i.
- $\overline{CT}(t_i)$ the average cost of running the task t_i.
- $\overline{TT}_{(i,j)}$ is the average cost of the communications of the edge (i, j).

This rank is calculated for all the tasks of the DAG starting with the exit task. The exit task has as rank only the average of the computation times on the different resources since it has no successor tasks.

The HEFT algorithm works in two phases; (*i*) The prioritization phase during which the priorities of all tasks are calculated using $rank_u$. Next, a list of tasks is generated by sorting the tasks according to their decreasing rank (priorities). (*ii*) The resource selection phase during which the tasks are assigned to computing resources which minimize their (*Finish Time*). The time $FT(t_i, vm_p)$ of a task i on a computational resource vm_p corresponds to the time at which the task finishes its execution, that is to say, the time of its start time $ST(t_i, vm_p)$ (*Start Time*) added to the time of its execution.

The HEFT algorithm proceeds according to the algorithm (1).

Algorithm 1: HEFT algorithm

```
   Data: W: workflow, PF: platform ;
   Result: S: a schedule;
 1 begin
 2 |   Calculate the rank for all the tasks by traversing the DAG upwards from the exit tasks
   |     according to the equation (10);
 3 |   Sort the tasks in a list following the rank in descending order ;
 4 |   while there are unscheduled tasks in the list do
 5 |   |   Select the first task t_i in the task list;
 6 |   |   foreach vm_p with p = 1..VM do
 7 |   |   |   Calculate FT(t_i, vm_p) ;
 8 |   |   |   Assign the task t_i to the machine vm_p such that FT(t_i, vm_p) is minimum ;
 9 |   |   end
10 |   end
11 end
```

Before detailing our algorithm, we explain the new budget distribution strategy that we designed for our budget-aware HEFT scheduling algorithm.

4.2 Budget Distribution

The main of budget distribution is to scatter the provided budget (B) to the workflow tasks. Each task receives a fragment of this budget. Several methods of budget distribution have been proposed. Example of recent strategies we quote [1,5] cited in the related work section. In this work, we propose a novel budget distribution strategy for our budget-aware scheduling algorithm, the Estimated Task Budget. With the Estimated Task Budget (ETB) strategy, we estimate the budget required for each task according to the costs of its processing time and the costs of the data transfers that it needs. This estimation is much closer to the specific needs of each task than any other fair approach. The ETB approach better reflects the needs of the tasks when the workflow is running in multiple data centers clouds. This is possible by also estimating the costs of transfers made between datacenters, which are not always free. Therefore, the initial budget used to cover the total execution of a workflow w can be distributed to the workflow tasks according to the features of each of them. We propose to estimate the proportional budget $B(t_i)$ reserved for a task t_i using the equation (11).

$$B(t_i) = \frac{\overline{CT}(t_i)}{\sum_{i=1}^{n} CT(t_i)} \times UC_{vm_p} + \frac{\sum_{j=1}^{m} \overline{data}_{ji}}{\sum_{(x,y)\in E} data_{(xy)}} \times UC_{data} \quad (11)$$

For each ready task, we calculate the corresponding sub-budget $B(t_i)$ of the initial budget (B) in proportion to the entire workflow. The first summand in the equation (11) concerns the unit cost of an instance by the ratio of the average computing time of a given task to the overall computing time of the workflow. The second summand represents the unit transfer cost by the ratio of the required data of a given task t_i to the total data to be transferred in the workflow.

Any unused fraction of the budget consumed when assigning previous tasks is recovered by the algorithm. For this purpose, the algorithm uses the variable B_{remain} which is initialized to zero for entry tasks but reclaims any remaining budget in previous assignments for the other tasks.

4.3 Budget-aware HEFT Algorithm

In this subsection, we present a budget-aware HEFT algorithm to allow its execution on workflows in multi-datacenter cloud platforms with budget constraints. We call this algorithm $B - HEFT$.

Initially, the HEFT heuristic [14] is presented to schedule a task graph on a set of heterogeneous processors. The adaptation of the HEFT heuristic, for the execution of workflow on platforms such as multi-datacenters clouds with budget constraint, requires rethinking the stage of instance selection to a given task (or the assignment of tasks to VMs phase).

Using the proposed time model described in section 3.2, we adapted the HEFT heuristic to the multi-datacenter cloud model. Moreover, we use the Estimated task budget for distributing the global budget over the workflow tasks.

The $B - HEFT$ is described by the following algorithm (2).

Our algorithm started by sorting the tasks in a list following the $rank_u$ in descending order (lines 2 and 3). Initially, for each task t_i in the ordered list, our algorithm affects the cheapest VM in the platform PF (lines 7 and 8) as the best VM, which does not necessarily provide the minimum earliest finish time for this task. The $EFT(t_i, vm_p)$ is obtained using the equation (4)(line 9). The algorithm calculates the share of the budget for each task using the proposed Estimation task budget (line 10). In addition, the algorithm recovers any part of the remaining budget from previous tasks (line 11). From line 12 to line 19, the algorithm calculates the $EFT(t_i, vm_p)$ according to the equation (4) with each VM vm_p. Then, the task cost is generated using equation (8). The B-HEFT algorithm verifies the finish time, checks if the budget $B(t_i)$ is not exceeded, and updates the minimal finish time and the best vm if it's necessary. Afterward, the algorithm evaluates the remaining budget by subtracting its real cost from its own reserved budget. Finally, the algorithm returns the best vm for the task (line 19).

5 Evaluation and Results

In this section, we present the experiments carried out to evaluate the performance of the proposed approach.

To assess the performance of our approach, we tested the algorithms with workflows generated based on the characteristics of two well-known real workflow applications, Montage and CyberShake, from two different scientific fields.

Algorithm 2: B-HEFT algorithm

Data: W:workflow, PF: platform, B: Budget;
Result: S: a schedule;

```
1  begin
2      Calculate the rank for all the tasks according to the equation (10);
3      Sort the tasks in a list following the rank in descending order ;
4      while there are unscheduled tasks in the list do
5          Select the first task t_i in the task list;
6          // initialization: Best_VM as the cheapest VM ;
7          UC_vm = min{UC_vm_p} with p = 1..VM ;
8          Best_VM = vm;
9          Calculate EFT(t_i, Best_VM) according to the equation (4) ;
10         Calculate B(t_i) according to the equation (11) ;
11         B(t_i) = B(t_i) + B_remain;
12         foreach vm_p with p = 1..VM do
13             Calculate EFT(t_i, vm_p) according to the equation (4) ;
14             Calculate Cost(t_i) according to the equation (8) ;
15             if ((EFT(t_i, vm_p) < EFT(t_i, Best_VM)) and (cost(t_i) ≤ B(t_i))) then
16                 EFT(t_i, Best_VM) = EFT(t_i, vm_p) ;
17                 Best_VM = vm_p ;
18                 B_remain = B(t_i) − cost(t_i)
19             end
20             Assign t_i to Best_VM ;
21         end
22     end
23 end
```

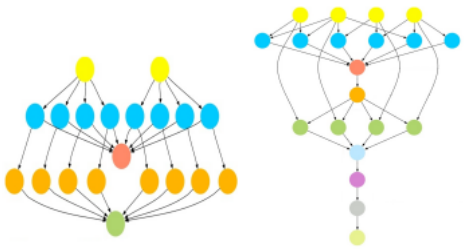
The CyberShake workflow is a data-intensive application of seismology used to describe earthquakes by generating synthetic seismograms. The Montage workflow is used in astronomy for generating mosaics personalized from the sky using a set of input images. Most of its tasks are I/O intensive. These workflows are generated by WorkflowGenerator[4] tool. The structure of the small workflows are shown by Fig. 1.

We modeled an IaaS provider platform with a platform of three data centers with 125 MBps as Bandwidth and \$0.055 per GB as data transfer cost. In each data center, three types of VMs can be allocated. Their characteristics (CPU power and cost) are similar to the c4 compute-optimized instance types offered by Amazon EC2. The configurations of the VM type used are shown in Table (1).

In the experiments, different budget intervals were employed. We assume that the minimum budget (B_{min}) for running the workflow equals the cost of running all tasks on the single cheapest VM. We establish four different budget intervals based on this minimum budget as indicated in equation (12), then we use the rounding to an integer value for B.

$$B = \alpha \star B_{min} \quad where \quad 0 < \alpha < 5 \tag{12}$$

[4] https://confluence.pegasus.isi.edu/display/pegasus/WorkflowGenerator



(a) CyberShake (b) Montage

Fig. 1: Structure of the small scientific workflows

Table 1: Type and prices of VMs

VM type	Name	vCPU	Price per Seconde ($)
1	small	2	0.0045
2	medium	4	0.009
3	large	8	0.018

We assessed different workflows to compare performance with respect to workflow size. We used 25, 50 and 100 tasks for Montage workflow and used 30, 50 and 100 tasks for CyberShake workflow.

To afford a baseline comparison, we implemented a 'uniform' distribution as a basic strategy and an adapted version of the 'All-in' distribution proposed in [1].

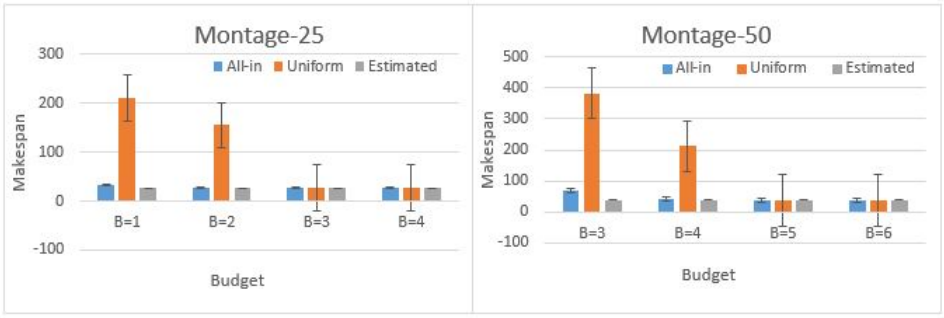
The uniform distribution strategy operates blindly, sharing the budget provided over the set of tasks by equal sharing. Except for the number of tasks, no other information concerning the workflow structure or task characteristics is considered. With the 'All-in' distribution strategy, the total budget is assigned to the first level. After serving all tasks at this level, any remaining budget is trickled to the next level. Since our B-HEFT algorithm does not perform level building preprocessing, we implement an 'All-in' version based on the ordered list of tasks instead of levels. In this case, the total budget is given for the first ready task in the workflow. Then after serving this first task, the remaining budget will be devoted to the next ready task in the list, and so on.

5.1 Montage workflow

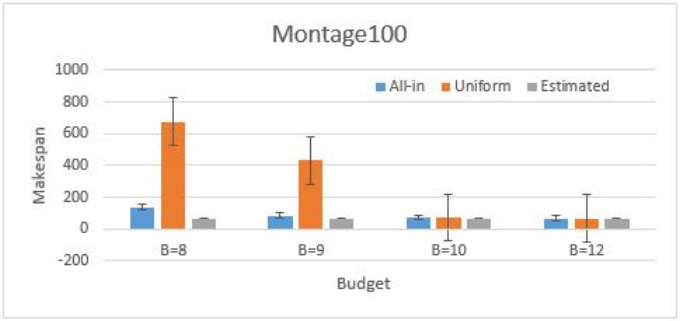
Fig. 2 shows the makespan achieved for Montage workflow using different budgets. With the different sizes of Montage workflow, our algorithm, plotted as 'Estimated' for Estimated task budget shows identical makespan as 'All-in' approach. Both 'Estimated' and 'All-in' perform better than the 'Uniform' strategy. This is due to the fact that when 'uniform' equitably distributes a restricted budget (B=1, B=2 for Montage-25 for example) between all the tasks, the share of each becomes insufficient. Then the algorithm assigns the tasks to the cheap instances, thus increasing the makespan (since the cheap instances are also slow). We point out the power of our algorithm to improve makespan compared to 'All-in' using a minimal budget.

In fact, our algorithm, using the 'Estimated' approach for sharing the budget, distributes for each task a sufficient sub-budget which is close to its real cost. On the other hand, 'All-in' is more

generous with the first tasks of the workflow (starting with the input tasks) by assigning them the maximum budget, therefore being able to run on faster instances. Unfortunately for the last tasks, if the budget is not sufficient, they run on slow instances which lengthens the makespan.



(a) Makespan for Montage, 25 tasks (b) Makespan for Montage, 50 tasks



(c) Makespan for Montage, 100 tasks

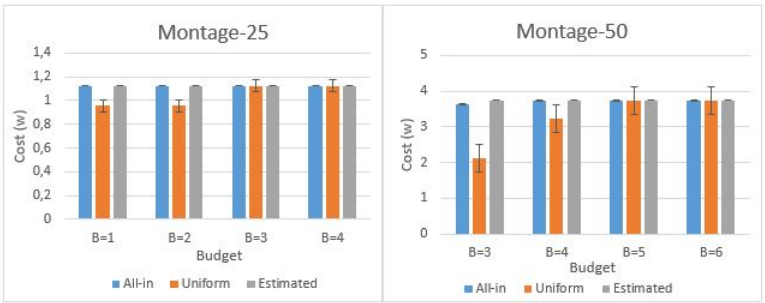
Fig. 2: Makespan for Montage workflows grouped by Budget

The costs recorded for Montage workflow based on different budgets are shown in Fig. 3. The costs recorded by our strategy are very close to those obtained by the 'All-in' case. The 'Uniform' strategy got reduced costs for the restricted budgets because it assigns the tasks to the cheap instances in this case to the detriment of the makespan. These benefits are visible in particular for significant sizes workflows like Montage with tasks sizes 50 and 100 (figure 3 (b) and (c)). In most cases, the three strategies meet the budget.

5.2 CyberShake workflow

Fig. 4 shows the makespan achieved for CyberShake workflow using different budgets. With CyberShake, our approach also achieves makespan identical to those obtained by 'All-in' with relaxed budgets, but rather better in the case of a restricted budget for all the CyberShake task sizes (Fig. 4 (a),(b) and (c)).

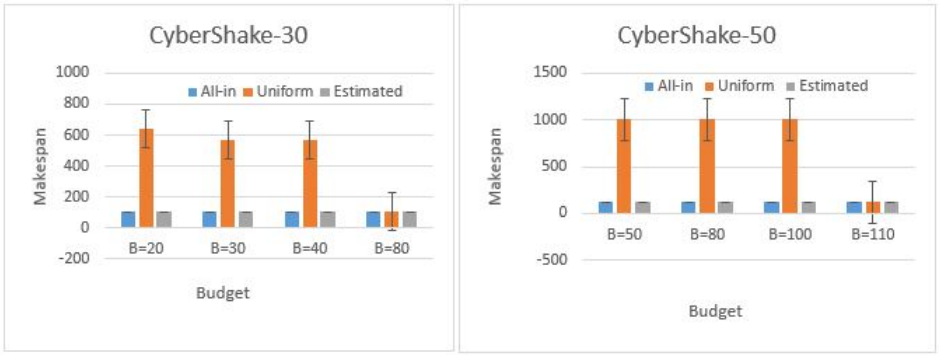
The makespan obtained by 'Uniform' is high compared to 'Estimated' and 'All-in' especially in with restricted budgets. Note that in the case of CyberShake with 100 tasks, 'Uniform' fails to



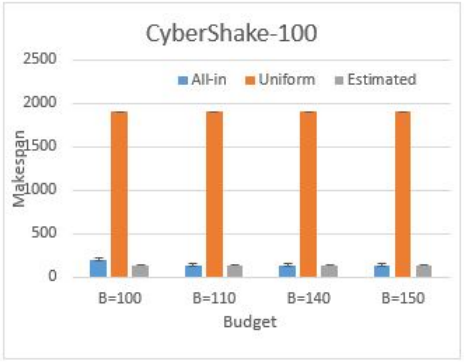
(a) Cost for Montage, 25 tasks (b) Cost for Montage, 50 tasks

(c) Cost for Montage, 100 tasks

Fig. 3: Cost for Montage workflows grouped by Budget



(a) Makespan for CyberShake, 30 tasks (b) Makespan for CyberShake, 50 tasks



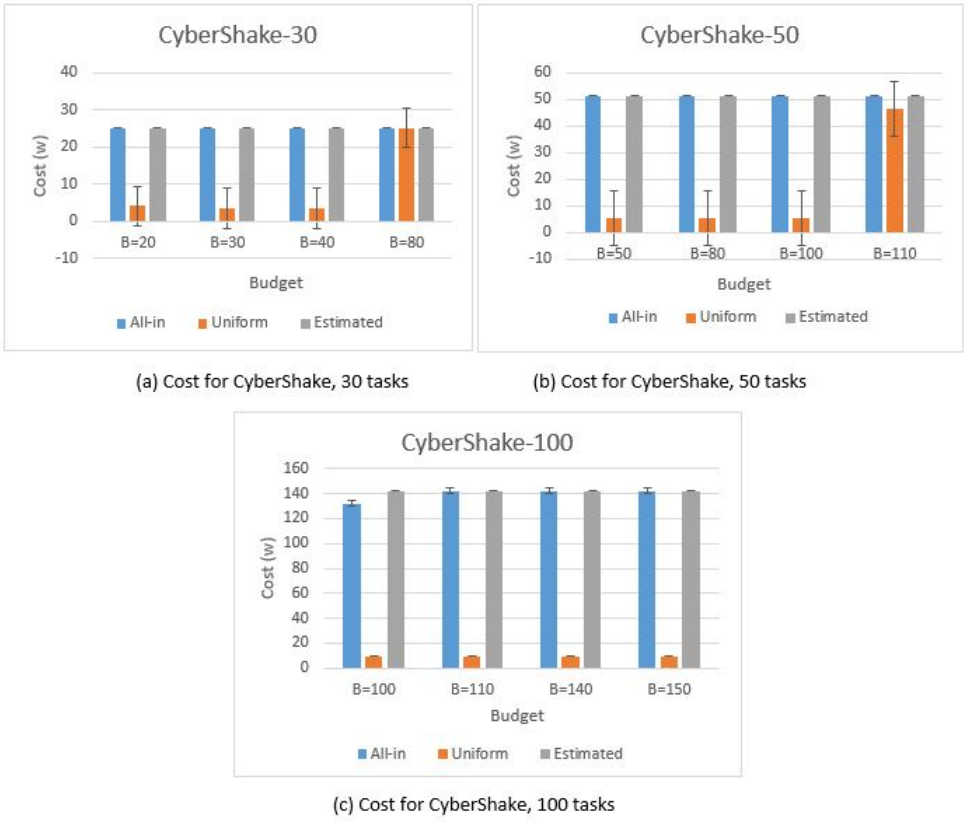
(c) Makespan for CyberShake, 100 tasks

Fig. 4: Makespan for CyberShake workflows grouped by Budget

optimize the makespan even with a relaxed budget (Fig. 4) (c)). Our approach achieves performances identical to those obtained by the 'All-in' strategy with relaxed budgets, but rather better in the case of a restricted budget.

As shown in Fig. 5, the results of CyberShake workflow regarding the costs are plotted. The costs realized by our approach are very close to those obtained by 'All-in' except for the workflow of size 100. In this case, 'All-in' behaves better compared to our approach. This is due eventually to the fact that the estimated values of the task's characteristics are not sufficiently accurate when dealing with a large number of tasks.

With the 'Uniform' strategy, the tasks costs are reduced. But in reality, the approach is unable to offer sufficient budgets for tasks with restricted budgets. It just runs them on cheap instances without actually meeting the task's needs. When offering a relaxed budget, 'Uniform' recorded costs as high as these of the other two approaches.



(a) Cost for CyberShake, 30 tasks

(b) Cost for CyberShake, 50 tasks

(c) Cost for CyberShake, 100 tasks

Fig. 5: Cost for CyberShake workflows grouped by Budget

6 Conclusion and Future work

In this paper, we presented a budget-aware HEFT algorithm for optimizing makespan when meeting the budget. The budget-aware algorithm is based on an estimated task budget which calculates the budget share required for each task in the workflow. We target as a platform the multi-datacenter clouds so that the costs of data transfers between data centers are considered.

We evaluated the makespan and costs using two real-world workflows with budget scenarios. Our approach shows identical makespan as 'All-in' distribution strategy but surpasses it in the case of restricted budgets. The 'Uniform' approach is far from being competitive with our algorithm by recording slower makespan. This makes the power of our algorithm to improve makespan with a minimal budget. In most cases, our algorithm behaves as an 'All-in' approach, without exceeding budget constraints.

Future work will focus on extending the experimental tests with large scale workflows. Other more precise performance metrics will be used to better assess the proposed algorithm.

References

1. Vahid Arabnejad, Kris Bubendorfer, and Bryan Ng. Budget distribution strategies for scientific workflow scheduling in commercial clouds. In *2016 IEEE 12th International Conference on e-Science (e-Science)*, pages 137–146. IEEE, 2016.
2. Vahid Arabnejad, Kris Bubendorfer, and Bryan Ng. Budget and deadline aware e-science workflow scheduling in clouds. *IEEE Transactions on Parallel and Distributed Systems*, 30(1):29–44, 2018.
3. Tracy D Braun, Howard Jay Siegel, Noah Beck, Ladislau L Bölöni, Muthucumaru Maheswaran, Albert I Reuther, James P Robertson, Mitchell D Theys, Bin Yao, Debra Hensgen, et al. A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems. *Journal of Parallel and Distributed computing*, 61(6):810–837, 2001.
4. Peter Brucker and P Brucker. *Scheduling algorithms*, volume 3. Springer, 2007.
5. Yves Caniou, Eddy Caron, Aurélie Kong Win Chang, and Yves Robert. Budget-aware scheduling algorithms for scientific workflows with stochastic task weights on heterogeneous iaas cloud platforms. In *2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 15–26. IEEE, 2018.
6. K Kalyan Chakravarthi, L Shyamala, and V Vaidehi. Budget aware scheduling algorithm for workflow applications in iaas clouds. *Cluster Computing*, pages 1–15, 2020.
7. Thomas Fahringer, Alexandru Jugravu, Sabri Pllana, Radu Prodan, Clovis Seragiotto Jr, and Hong-Linh Truong. Askalon: a tool set for cluster and grid computing. *Concurrency and Computation: Practice and Experience*, 17(2-4):143–169, 2005.
8. Robabeh Ghafouri, Ali Movaghar, and Mehran Mohsenzadeh. Time-cost efficient scheduling algorithms for executing workflow in infrastructure as a service clouds. *Wireless Personal Communications*, 103(3):2035–2070, 2018.
9. Pengcheng Han, Chenglie Du, Jinchao Chen, Fuyuan Ling, and Xiaoyan Du. Cost and makespan scheduling of workflows in clouds using list multiobjective optimization technique. *Journal of Systems Architecture*, page 101837, 2020.
10. Pingping Lu, Gongxuan Zhang, Zhaomeng Zhu, Xiumin Zhou, Jin Sun, and Junlong Zhou. A review of cost and makespan-aware workflow scheduling in clouds. *Journal of Circuits, Systems and Computers*, 28(06):1930006, 2019.
11. Karima Oukfif, Lyes Bouali, Samia Bouzefrane, and Fatima Oulebsir-Boumghar. Energy-aware dpso algorithm for workflow scheduling on computational grids. In *Future Internet of Things and Cloud (FiCloud), 2015 3rd International Conference on*, pages 651–656. IEEE, 2015.

12. Karima Oukfif, Fatima Oulebsir-Boumghar, Samia Bouzefrane, and Soumya Banerjee. Workflow scheduling with data transfer optimisation and enhancement of reliability in cloud data centres. *International Journal of Communication Networks and Distributed Systems*, 24(3):262–283, 2020.
13. Maria A Rodriguez and Rajkumar Buyya. Budget-driven scheduling of scientific workflows in iaas clouds with fine-grained billing periods. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 12(2):1–22, 2017.
14. Haluk Topcuoglu, Salim Hariri, and Min-you Wu. Performance-effective and low-complexity task scheduling for heterogeneous computing. *IEEE transactions on parallel and distributed systems*, 13(3):260–274, 2002.
15. Amandeep Verma and Sakshi Kaushal. Cost-time efficient scheduling plan for executing workflows in the cloud. *Journal of Grid Computing*, 13(4):495–506, 2015.
16. Fuhui Wu, Qingbo Wu, and Yusong Tan. Workflow scheduling in cloud: a survey. *The Journal of Supercomputing*, 71(9):3373–3418, 2015.
17. Wei Zheng and Rizos Sakellariou. Budget-deadline constrained workflow planning for admission control. *Journal of grid computing*, 11(4):633–651, 2013.
18. Naqin Zhou, Weiwei Lin, Wei Feng, Fang Shi, and Xiongwen Pang. Budget-deadline constrained approach for scientific workflows scheduling in a cloud environment. *Cluster Computing*, pages 1–15, 2020.