Data-Aware Scheduling of Scientific Workflows in Hybrid Clouds

Amirmohammad Pasdar[1], Khaled Almi'ani[2], and Young Choon Lee[1]

[1] Macquarie University, Department of Computing, Sydney, NSW 2109, Australia
[2] Al-Hussein Bin Talal University and Princess Sumaya University for Technology, Jordan
amirmohammad.pasdar@hdr.mq.edu.au, k.almiani@ahu.edu.jo, young.lee@mq.edu.au

Abstract. In this paper, we address the scheduling of scientific workflows in hybrid clouds considering data placement and present the **H**ybrid **S**cheduling for **H**ybrid **C**louds (***HSHC)*** algorithm. HSHC is a two-phase scheduling algorithm with a genetic algorithm based static phase and dynamic programming based dynamic phase. We evaluate HSHC with both a real-world scientific workflow application and random workflows in terms of makespan and costs.

1 Introduction

The cloud environment can be classified as a public, private or hybrid cloud. In a public cloud, users can acquire resources in a per-as-you-go manner wherein different pricing models exist as in Amazon EC2. The private cloud is for a particular user group and their internal usage. As the resource capacity of the latter is "fixed" and limited, the combined use of private and public clouds has been increasingly adopted, i.e., hybrid clouds [1].

In recent years, scientific and engineering communities are increasingly seeking a more cost-effective solution (i.e., hybrid cloud solutions) for running their large-scale applications [2],[3],[4]. Scientific workflows are of a particular type of such applications. However, as these workflow applications are often resource-intensive in both computation and data, the hybrid cloud alternative struggles to satisfy execution requirements, particularly of data placement efficiently.

This paper studies the data placement of scientific workflow execution in hybrid clouds. To this end, we design Hybrid Scheduling for Hybrid Clouds (HSHC) as a novel two-phase scheduling algorithm with the explicit consideration of data placement. In the *static* phase, an extended genetic algorithm is applied to simultaneously find the best place for datasets and assign their corresponding tasks accordingly. In the *dynamic* phase, intermediate data during workflow execution are dealt with to maintain the quality of schedule and execution. We evaluate HSHC in comparison with *FCFS* and *AsQ* [5]. Results show *HSHC* is a cost-effective approach that can utilize the private cloud effectively and reduce data movements in a hybrid cloud.

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93713-7_68

```
Algorithm 1: Static Algorithm
  Data: private VM list, task-list, dataset-list, #generation, mutation
        probability
  Result: The optimal placement of datasets and assignment of tasks
  Prepare the initGeneration;
  Evaluate initGeneration via Equation 2;
  while either the optimal is not found, or generation size is not exceeded do
      apply tournament-selection;
      apply k-way-crossover;
      apply concurrent-rotation mutation;
      evaluate the new solution and add it the to population;
  end
  rank the population and return the best one;
```

2 Hybrid Scheduling for Hybrid Clouds (HSHC)

HSHC consists of two phases: static and dynamic. The static phase deals with finding the optimal placement for tasks and their datasets in the private cloud, using a genetic algorithm (GA, Algorithm 1). In the dynamic phase, some tasks are expected to be re-allocated due to changing execution conditions.

2.1 Static Phase

To generate the initial population, we start by placing the fixed number of tasks and their corresponding data sets to computational resources (virtual machines or VMs). Then, we randomly assign the flexible datasets and tasks to the rest of VMs in the system. The representation of a solution (chromosome) in our algorithm is shown in Fig. 1. In this structure, a cell (i) consists of task-set (TS_i), data-set (DS_i), and a VM (VM_i) that hosts the assigned data-set and task-set.

The main objective here is determining the locality of the tasks and datasets such that the overall execution time is minimized. This results in considering several factors during the representation of the fitness function. During the construction of the solution, for any given task, in order to reduce the delay in execution, this task must be assigned to the VM that results in increasing the number of available data sets for this task execution. We denote the percentage of available data sets for task i at virtual machine j by $ava(t_i, VM_j)$. Moreover, for any given VM (VM_j), the dependency between data sets assigned to the VM and data sets located at different VMs must be minimized. In other words, the

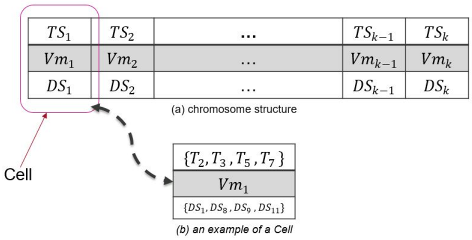


Fig. 1: Chromosome structure.

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93713-7_68

dependency between data sets assigned to the same VM must be maximized. For a VM (VM_j), we refer to the data dependency between data sets assigned to VM_j as DI_j and the data dependency between its data sets and data sets of other VMs' as DO_j.

To ensure the feasibility of a solution, we use the variable ck_f to check if the assignment for the fixed data sets and tasks does not violate the locality constraints. We also use the variable ck_r to check if the assigned task can retrieve the required data sets from its current VM. Moreover, to pick up the best VMs, we use a variable termed P_{ratio}. This ratio represents how robust the selected VMs for task execution are, and it is defined as:

$$P_{ratio} = \sum_{i=0}^{M} s_i \tag{1}$$

Tasks assigned to a virtual machine might have the ability to be executed concurrently. Thus, a delay is defined to help the fitness function with the selection of solutions that have less concurrent values. To find the concurrent tasks within a VM, the workflow structure has to be considered. In other words, tasks are monitored to be realized when they will be available for execution in accordance with their required intermediate datasets. Then, they are categorized, and their execution time is examined against their assigned deadline. If within a VM, there are tasks that they do not need any intermediate datasets, they will also influence concurrent tasks. Once the priorities are ready, the amount of time they would be behind their defined deadline is evaluated based on the total amount of delay for a solution, $Tdelay$. In some situations, a solution might have virtual machines that do not have either a task-set or data-set. Thus, to increase the number of used VMs, we introduced the variable Vu_j, which denotes the percentage of the used virtual machine in the solution. Given these variables, the fitness function is defined as follows.

$$\begin{aligned} fitness = (p_r \times ck_r \times ck_f \times \sum_{j=0}^{M} vu_j) \times (\sum_{i=0;j=0}^{|T|,M} ava(t_i, VM_j) \\ \times (\sum_{i=0;j=0}^{|T|,M} DI_j - \sum_{i=0;j=0}^{|T|,M} DO_j) - \sum_{j=0}^{M} Tdelay_j) \end{aligned} \tag{2}$$

As the solutions are evaluated, the new solution should be produced based on the available ones. Due to the combined structure of the chromosome, the normal crossover operation cannot be applied to the solutions. Thus, the newly proposed crossover knowns as *k-way-crossover* is introduced. In this operation, after getting the candidates based on a tournament selection, k-worst and k-best cells of each parent are swapped with each other. By utilizing this crossover, the length of a solution may change. If either the best or worst cell has any fixed datasets, they will remain within the cell along with their corresponding tasks. *Concurrent-rotation* is introduced to mutate a solution. It creates four different

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93713-7_68

ways to mutate a chromosome. For each cell within a solution, the task and dataset with least dependency are selected and by the direction-clockwise or counterclockwise- they are exchanged with the next or previous cell. If the least dependency is related to fixed datasets and its corresponding task, the next least task and dataset are selected.

The output solution is used to initially place datasets and assign their corresponding tasks to the most suitable VMs inside a private cloud.

2.2 Dynamic Phase

In the dynamic phase, based on the status of the available resources (VMs), task reallocation is done with the private cloud or public cloud. Given the ready-to-execute tasks, we map them to the VMs such that the total delay in execution is minimized. This mapping begins by trying to schedule tasks which become ready based on their required intermediate datasets in the private cloud. Then, we offload the tasks that cannot be scheduled in the private cloud to be performed in the public cloud.

In the beginning, we divide the ready-to-execute tasks into flexible and non-flexible sets. The non-flexible set contains tasks with fixed datasets, and this results in restricting the locality of the VMs. The flexible set contains the tasks that can be executed at any VMs, as long as necessary criteria like deadline and storage are met. We are mainly concerned with scheduling of the flexible tasks. We start by calculating the available capacity and workload for the current "active" VMs. This is used to determine the time in which these VMs can execute new tasks. For each task (t_i) and VM (vm_i), we maintain a value that represents the time when vm_i can finish executing t_i. We refer to this value as $T(t_i, vm_i)$. These values are stored in $FTMatrix$. Task reallocation is established by identifying the task ($t_i \in T$) and the VM ($vm_i \in VM$) such that the finish time ($FT(t_i, vm_i)$) is the lowest possible value in $FTMatrix$. If assigning t_i to vm_i does not violate this task deadline and its required storage, this assignment will be confirmed. In this case, we will refer to t_i and vm_i as the last confirmed assignment. Otherwise, this task will be added to an offloading list (l_{off}), where all tasks belonging to this list will be scheduled on the public cloud at a later stage. There is an exception to the list, and it is when the storage criterion would be the only violation for a task that could not be satisfied. Thus, the task would be removed from the list and would be added again in another time fraction to the ready-to-execute list. This process of task reallocation takes place for all tasks in the flexible set.

All of the offloaded tasks will be scheduled to be executed on the public cloud. Scheduling these tasks uses a strategy similar to the private cloud scheduling.

3 Evaluation

In this section, we present our evaluation results in comparison with FCFS and AsQ. The former is the traditional scheduling approach for assigning tasks to

DOI: 10.1007/978-3-319-93713-7_68

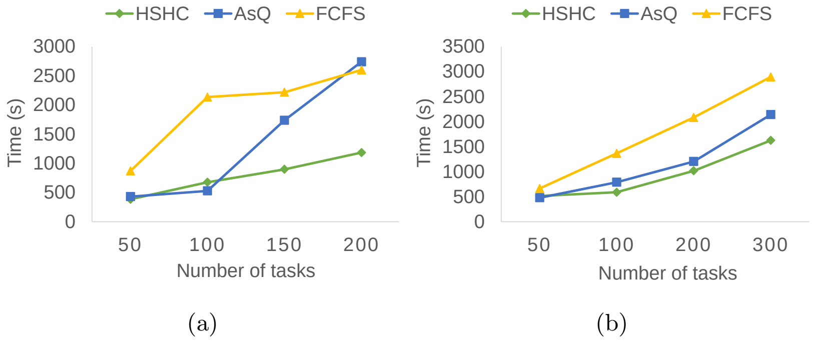


Fig. 2: Average execution time: (a) Random workflows. (b) Montage workflows.

VMs through a queue-based system. The latter schedules deadline-based applications in a hybrid cloud environment in a cost-effective manner.

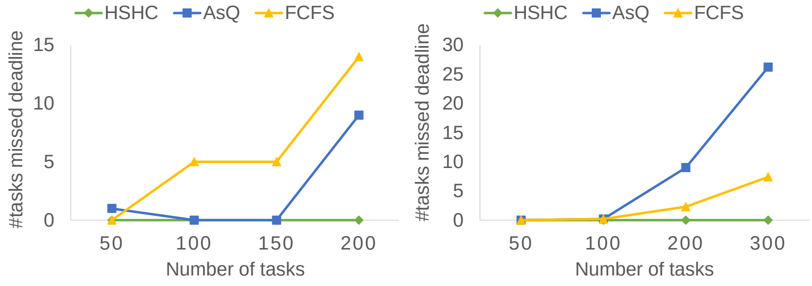
3.1 Environment Setup

The simulation environment is composed of a private cloud and a public cloud with 20 VMs each. The public cloud has three types of VM: small, medium and large with 4, 8 and 8 VMs, respectively. The cost of public cloud is similar to the pricing of Amazon EC2, i.e., proportional pricing based on the rate of the small VM, $0.023 per hour in this study. VMs in the private cloud are with the computing power of 250MIPS whereas that for VM types in the public cloud are 2500MIPS, 3500MIPS, and 4500MIPS, respectively.

For the *static* phase, the initial population size is 50, and max generation size is 500. Mutation probability is 0.2. We have used Montage scientific workflows (`http://montage.ipac.caltech.edu/docs/grid.html`) and random workflows with user-defined deadlines as the input for our simulation. The random workflow is created based on a hierarchical structure that has 85 datasets which have sizes in MB from [64 - 512]. The input and output degrees are chosen from [3 - 8]. Extracted results are based on the 10% fixed datasets for the random workflows and are reported in average based on 10 simulation runs.

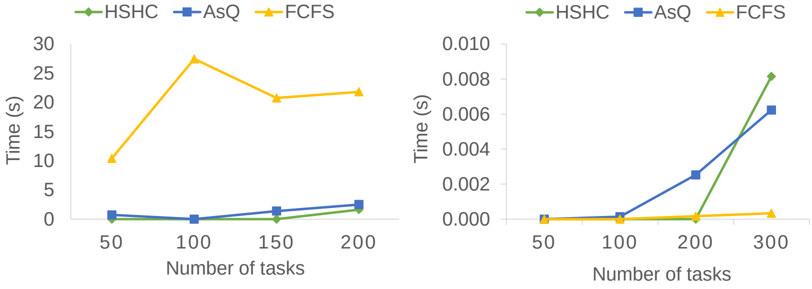
3.2 Results

As VMs in the homogeneous private cloud come with the low computation capabilities, consequently, the higher offloading rate to the public cloud would be noticeable to stay with the task's deadline; Fig. 5(a) and Fig. 5(b). Moreover, the workflow structure also influences the offloading process. Therefore, considering the deadline as well as the ability of the private cause to execute tasks in the public cloud. Also, the utilization of public cloud resources would lead having minimum execution time as it would reduce the average execution time but would increase the cost. Despite the fact that the cost for *FCFS* is almost zero for Montage shown in Fig. 6(b) it could not achieve meeting deadlines Fig. 3(b).



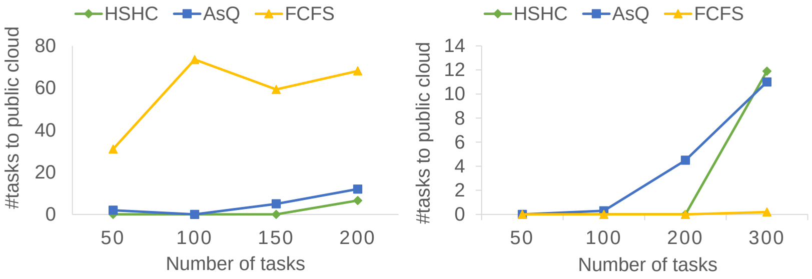
(a) (b)

Fig. 3: Average number of tasks that missed deadline: (a) Random (b) Montage.



(a) (b)

Fig. 4: Average transferring time: (a) Random (b) Montage.

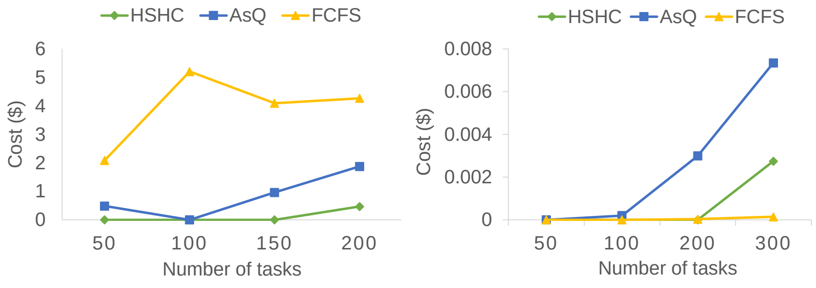


(a) (b)

Fig. 5: Average number of tasks dispatched to public cloud: (a) Random (b) Montage.

Our proposed method executed all the tasks within the expected deadline. As it is shown in Fig. 3, missed deadlines for the other methods is considerable.

Contrary, *HSHC* met all task deadlines by offloading a portion of tasks to the public cloud. For the random workflows shown in Fig. 5 *AsQ* sent fewer tasks to the public cloud in comparison to our approach, but it could not execute



(a) (b)

Fig. 6: Average cost: (a) Random (b) Montage.

tasks within their deadlines. This is also true for *FCFS* as it obtained a better execution time due to a higher rate of offloading but it could not also utilise the VMs properly. Thus, *HSHC* not only outperforms other approaches but also efficiently utilizes cloud VMs for task execution.

4 Conclusion

In this paper, we have studied data placement aware scheduling of scientific workflows in a hybrid cloud environment and presented HSHC. The scheduling process is divided into static and dynamic phases. The static phase leverages a customised genetic algorithm to concurrently tackle data placement and task scheduling. In the dynamic phase, the output schedule from the static phase adapts to deal with changes in execution conditions. The evaluation results compared with FCFS and AsQ demonstrated the efficacy of HSHC in terms particularly of makespan and cost by judiciously using public cloud resources.

References

1. Hoseinyfarahabady, M.R., Samani, H.R.D., Leslie, L.M., Lee, Y.C., Zomaya, A.Y.: Handling uncertainty: Pareto-efficient bot scheduling on hybrid clouds. In: International Conference on Parallel Processing (ICPP). (2013) 419–428
2. Bittencourt, L.F., Madeira, E.R., Da Fonseca, N.L.: Scheduling in hybrid clouds. IEEE Communications Magazine **50**(9) (2012)
3. Rahman, M., Li, X., Palit, H.: Hybrid heuristic for scheduling data analytics workflow applications in hybrid cloud environment. In: International Symposium on Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), IEEE 966–974
4. Farahabady, M.R.H., Lee, Y.C., Zomaya, A.Y.: Pareto-optimal cloud bursting. IEEE Transactions on Parallel and Distributed Systems **25**(10) (2014) 2670–2682
5. Wang, W.J., Chang, Y.S., Lo, W.T., Lee, Y.K.: Adaptive scheduling for parallel tasks with qos satisfaction for hybrid cloud environments. The Journal of Supercomputing **66**(2) (2013) 783–811