

A DATA PLACEMENT APPROACH FOR SCIENTIFIC WORKFLOW EXECUTION IN HYBRID CLOUDS

By

Amirmohammad Pashar

A THESIS SUBMITTED TO MACQUARIE UNIVERSITY

IN PARTIAL FULFILMENT OF THE DEGREE OF

MASTER OF RESEARCH

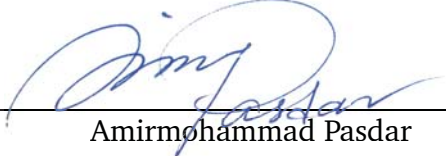
DEPARTMENT OF COMPUTING

APRIL 2018



Declaration

I certify that the work in this thesis entitled "A DATA PLACEMENT APPROACH FOR SCIENTIFIC WORKFLOW EXECUTION IN HYBRID CLOUDS" has not previously been submitted for a degree nor has it been submitted as part of the requirements for a degree to any other university or institution other than Macquarie University. I also certify that the thesis is an original piece of research and it has been written by myself. Any help and assistance that I have received in my research work and the preparation of the thesis itself have been appropriately acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.



Amirmohammad Pashar

Dedication

In Loving Memory of My Dear Parents, Hasan and Zinat

Deep In My Heart
You Will Always Stay
Loved and Remembered
Everyday

*To my beloved brothers and sister
With love and eternal appreciation*

Acknowledgements

I would like to express my sincere appreciation to Dr. Young Choon Lee for his professional supervisory role throughout my career. He has been supportive of my career goals and actively worked to provide me with the time to be a successful student. I would like to acknowledge that without his support this piece of work could not have been accomplished.

I also appreciate the support I was offered during my studies by the Computing Department for being on the right track of MRes. I would also like to acknowledge Macquarie University for offering the Australian Commonwealth Government Scholarship as a funding scheme for my MRes which paved the way for a successful Ph.D. career.

Lastly, I would sincerely like to thank my loving and supportive brothers, especially Amirhossein, and my wonderful sister, Yukabed, whose unconditional love, encouragement, and support are with me in whatever I pursue. I also wish to thank my elder brother, Alireza and his lovely family, in particular, my beloved nieces, Shima and Zahra, who have been very inspirational and motivational. May God surround all of them with loving care, always and forever.

List of Publications

- A. Pasdar, K. Almi'ani, and Y. C. Lee, *Data-Aware Scheduling of Scientific Workflows in Hybrid Clouds*, International Conference on Computational Science (ICCS2018).

Abstract

Cloud computing has been widely adopted by industry practitioners and researchers. Recently, applications in science and engineering such as scientific workflows have also been increasingly deployed in clouds. As these applications are becoming resource intensive in both data and computing, private clouds struggle to cope with their resource requirements. Public clouds claim to overcome many shortcomings of private clouds. However, the complete offloading of workflow execution to public clouds may introduce excessive data transfer and privacy/governance concerns. In this thesis, we propose a hybrid cloud solution for workflow scheduling explicitly considering data placement. To this end, we present **Hybrid Scheduling for Hybrid Clouds (HSHC)**, which schedules scientific workflows across private and public clouds incorporating a novel dynamic data placement policy. HSHC consists of two phases: static and dynamic. The former uses an extended genetic algorithm to solve the problem of workflow scheduling with static information of workflows and cloud resources. The latter adjusts scheduling and data placement decisions reflecting changing conditions of workflow execution in the hybrid cloud. We evaluate HSHC with both real-world scientific applications and random workflows in performance and cost. Experimental results demonstrate HSHC’s two-phase approach effectively deals with the dynamic nature of the hybrid cloud.

Contents

Declaration	iii
Dedication	v
Acknowledgements	vii
List of Publications	ix
Abstract	xi
List of Figures	xvii
List of Tables	xix
1 Introduction	1
1.1 Lead-in	1
1.2 Objective of The Thesis	3
1.3 Organization of The Thesis	4
2 Literature Review	5
2.1 Lead-in	5
2.2 Workflow Scheduling in Clouds	6
2.2.1 Computation-Centric Workflow Scheduling	6
2.2.2 Data-Aware Workflow Scheduling	6

2.3	Workflow Scheduling in Hybrid Clouds	7
2.4	Summary	9
3	Problem Statement	11
3.1	Lead-in	11
3.2	Definitions	13
3.2.1	Cost	13
3.2.2	Data Dependency	13
3.2.3	Task Execution	14
3.3	Problem Definition	15
3.4	Summary	15
4	Hybrid Scheduling for Hybrid Clouds	17
4.1	Lead-in	17
4.2	Static Phase: A Genetic Algorithm Approach	18
4.2.1	Chromosome Structure	18
4.2.2	Fitness Function	19
4.2.3	Static Algorithm	22
4.3	Dynamic Phase	23
4.3.1	In The Private Cloud	23
4.3.2	In The Public Cloud	24
4.4	Summary	28
5	Evaluation	29
5.1	Lead-in	29
5.2	Workflow Types	30
5.3	Environment Setup	31
5.4	Results and Discussion	32
5.4.1	A Heterogeneous Private Environment	32
5.4.2	A Homogeneous Private Environment	36
5.5	Summary	37

6 Conclusion	39
6.1 Lead-in	39
6.2 Conclusion	40
6.3 Future Work	41
 List of Symbols	 43
 References	 47

List of Figures

1.1	Montage workflow.	2
3.1	Example of a DAG representation of a workflow	12
4.1	The overall HSHC structure	17
4.2	The chromosome structure.	18
4.3	(a) A cell, (b) tasks with sorted required intermediate datasets, and (c) the priority of tasks within the virtual machine.	20
4.4	k-way-crossover.	21
4.5	The concurrent-rotation process.	21
5.1	Workflow types	30
5.2	Average execution time: (a) LIGO in hetero. (b) Montage in hetero. (c) Random in hetero. (d) LIGO in homo. (e) Montage in homo. (f) Random in homo.	33
5.3	Average cost of public cloud utilization: (a) LIGO in hetero. (b) Montage in hetero. (c) Random in hetero. (d) LIGO in homo. (e) Montage in homo. (f) Random in homo.	34
5.4	Average missed deadlines: (a) LIGO in hetero. (b) Montage in hetero. (c) Random in hetero. (d) LIGO in homo. (e) Montage in homo. (f) Random in homo.	35
5.5	Average dispatched tasks to the public cloud: (a) LIGO in hetero. (b) Montage in hetero. (c) Random in hetero. (d) LIGO in homo. (e) Montage in homo. (f) Random in homo.	36

5.6 Average transferring time to the public cloud: (a) LIGO in hetero. (b) Montage in hetero. (c) Random in hetero. (d) LIGO in homo. (e) Montage in homo. (f) Random in homo. 37

List of Tables

2.1 Summary of the literature and its comparison 10

5.1 Hybrid cloud simulation parameters 31

1

Introduction

1.1 Lead-in

Cloud computing has been brought forth with advances in x86 virtual machine (VM) techniques, such as VMware and Xen. The main difference between traditional computing systems and clouds is the elastic resource provisioning, with pay-as-you-go pricing, of clouds from the support of such VM techniques. Benefits from the elasticity and pricing model of clouds include cost efficiency, scalability, and improved performance. The major real-life debut was the launch of Amazon Web Services (AWS) S3 in late 2006¹ [1]. Since then, clouds have been widely adopted by not only industry practitioners, but also researchers.

¹<https://aws.amazon.com/s3/>

Over the past few decades, scientific applications, for example in bio-informatics and astronomy, have become increasingly large scale which necessitates the need for extensive computation and storage resources. Many of these applications deal with a large number of interdependent tasks, in the form of the workflow (i.e., scientific workflows) and they are resource intensive in terms of both computation and data. For example, a Montage astronomical image mosaic workflow (Figure 1.1) [2] with 6.0-degree data spawns nearly nine thousand precedence-constrained tasks and deals with several gigabytes of data. This trend of heavy resource usage has been handled to a certain extent by the fine-grained use of resources—VMs and containers in clouds whether they are private clouds or public clouds.

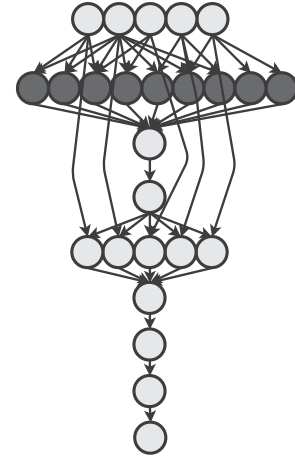


FIGURE 1.1: Montage workflow.

A cloud is classified as a private cloud or a public cloud depending on access mode. A private cloud is an infrastructure for performing workloads within a single administrative domain. It gives full system control, privacy and security and data governance while the resource capacity is limited or less flexible/elastic.

In the meantime, a public cloud, such as AWS, Microsoft Azure or Google Cloud Platform provides a wide range of cloud services with claims of virtually unlimited resources. Users, in theory, can acquire as many resources as needed and pay only for what has been used. For example, Amazon EC2 offers instances¹ of various types with different capacities. The pricing of these instances is based on the number of machine hours.

While scientists can offload their applications (scientific workflows in this study) to public clouds to take advantage of cost efficiency and improved performance, the data-intensiveness of recent large-scale scientific workflows significantly hinders such complete offloading (i.e., *cloud sourcing*). Besides, many organizations including research labs often operate their own private clouds due to privacy, security and data governance. A more practical alternative is the hybrid use of private cloud, and public cloud referred to as *cloud bursting*. In particular, tasks of a workflow are distributed across these different clouds explicitly taking into account data dependencies and

¹An instance in the context of Amazon EC2 is referred to a VM.

locality; here, scheduling plays a crucial role.

There have been extensive studies on workflow scheduling [3–6]. The work in [3] runs scientific workflows in grids considering resource heterogeneity. The works in [4, 5] focus on improving a specific objective, such as performance (makespan) and cost efficiency, respectively. The applicability of these works [3–5] is limited to a single system of tightly coupled resources within the same geographical location. Although the work in [6] considers the hybrid cloud, its focus is on compute-intensive workflows.

In this thesis, we address the problem of scheduling scientific workflows in hybrid clouds explicitly taking into account data placement. More specifically, for a given workflow, our workflow scheduling problem deals with the following: (1) resources of which cloud environment are used for which tasks, and (2) how data is distributed prior and during workflow execution considering costs and performance.

1.2 Objective of The Thesis

This thesis presents a data placement approach for scientific workflow execution in a hybrid cloud, called **Hybrid Scheduling for Hybrid Clouds (HSHC)**. The main objective is to optimize the trade-off between performance and costs by maximizing private cloud usage and minimizing data movement; however, they are conflicting objectives. HSHC reconciles this conflict with its two-phase approach based on a genetic algorithm and dynamic programming. They are defined as *static* and *dynamic*.

In the *static* phase, offline decisions are made through an extended genetic algorithm to deal with data locality before actual execution. The resultant schedule from this phase can be considered as the ideal execution plan with the assumption that tasks and resources perform as predicted.

In the *dynamic* phase, the “ideal” schedule from the static phase is adjusted based on changing conditions of both tasks and resources. These changing conditions include communication delays due to changes in intermediate data generation and volume, and performance variations of resources due to resource contention.

Our main contributions in this thesis can be summarised as follows:

- We present HSHC for workflow scheduling in a hybrid cloud with a data placement strategy.

- HSHC balances the performance and cost of workflow execution.
- We have conducted an extensive evaluation study with real-world scientific workflows.

We evaluate HSHC based on random workflows and real-world scientific workflows such as Montage and LIGO [7]. The results of HSHC are compared with First Come First Serve (*FCFS*) and *AsQ* [8]. Our results confirm the efficacy of HSHC in terms of performance and cost.

1.3 Organization of The Thesis

The remainder of the thesis is organized as follows: In [chapter 2](#), we overview studies on workflow scheduling in clouds. We define the problem and explain required preliminaries for a hybrid cloud in [chapter 3](#). In [chapter 4](#), HSHC is presented. We evaluate HSHC and show the results in [chapter 5](#). Lastly, the conclusion and future works are discussed in [chapter 6](#).

2

Literature Review

2.1 Lead-in

Scheduling plays an important role in the execution of workflows in terms of cost and performance. Workflow scheduling also needs to pay attention to user-defined Quality-of-Service (QoS) metrics such as security and energy consumption. However, it faces several challenges that emerge from limited resources in the private cloud and the multi-tenancy, on-demand, and pay-per-use model of resources in the public cloud.

Typical performance metrics of workflow scheduling include:

- **Makespan.** It mainly concerns the amount of time taken to execute a workflow.
- **Utilization.** The goal is to leverage idle time slots of instances to increase the overall utilization. It is beneficiary in terms of cost, energy consumption, and budget savings.

- **Cost efficiency.** It is an important matter as users are charged based on their usage. The less it costs, the more budget is saved, and as a consequence, more workflows could be dispatched for execution.

2.2 Workflow Scheduling in Clouds

Scheduling workflows have been extensively studied [4, 9, 10]. Due to their precedence constraint dictated by data dependency, the quality of schedule (e.g., makespan) is mainly dependent on reducing communication costs between tasks. This communication cost stems from transferring data among different instances, and it could happen in any cloud environments.

Workflow scheduling could be divided into two separate groups: computation-centric workflow scheduling and data-aware workflow scheduling. While the former focuses on improving workflow execution performance, the latter takes explicitly into account data management prior and/or during execution.

2.2.1 Computation-Centric Workflow Scheduling

It has been a challenge to acquire the proper amount of resources for workflow execution. The availability of resources could explicitly affect the execution time and its usage cost. Cost reduction has necessitated resource management which can be through an extended Partial Critical Paths (PCP) [4], dynamic, and priority-based scheduling [11], and Cluster Combining Algorithm (CCA) [12] to allocate resources to tasks for maximizing resources usage appropriately. It also goes further as successful execution of a workflow depends on the type and amount of resources that a workflow should request. Therefore, an optimal resource provisioning to execute the workflow within an acceptable deadline could be achieved through resource estimation [5].

2.2.2 Data-Aware Workflow Scheduling

Optimizing the execution performance in a cloud environment is very important when it deals with data-intensive workflows. It increases the total cost as users are also charged for the amount of bandwidth usage. Data placement is well-known for being an NP-Hard [9, 10, 13] problem. An

efficient data placement could explicitly reduce the execution time. Data clustering could be done through machine learning strategies like k-means [9, 13, 14], Bayesian Network [15], or graphs [10]. What it follows is to keep the relevant datasets close to each other and could be stored as chunks in different locations.

In terms of dependency, data locality could also become extended into data-task and task-task [16, 17] which might be done through task duplication. Data replication [17] could also increase data locality that would ease required data retrieving; however, it would increase storage space usage. It can also be in the form of choosing the best place for intermediate datasets [18] produced during the workflow execution. Thus, the usage rate of the intermediate data per a specific time could be monitored to decide to store the data or remove it from the system to reduce the overall storage cost.

2.3 Workflow Scheduling in Hybrid Clouds

Workflow scheduling has recently leveraged hybrid clouds as well to take advantage of both internal and external resources. As cloud resources are available with different capacities, an appropriate scheduling approach should take control of instances selection. The scheduling approach may be defined as a cost model to address minimum execution time as well as public resources rental cost by presenting a list of resources for execution purposes to the user [19]. The division for the price is based on considering the budget as a part of user application requirement. Workflow scheduling within a hybrid cloud is also implicitly related to the workflow type. For example, the majority of studies have addressed Bag-of-Tasks applications due to its simple structure as well as parallelism [20, 21].

Scheduling for a Bag-of-Tasks application can be classified as cost-based approaches which may come with prediction [22–24] to reduce the total cost. In other words, estimation could have a significant impact on the decision for task offloading. It may also apply a binary nonlinear programming [25] to maximize the local utilization and minimize the cost of public environment utilization. The cost model could also be in the form of profit maximization [26], a resource management and allocation scheme [8, 20, 27], increasing user privacy [28], and improving the throughput [29] via a combination of a grid environment and a public cloud.

The cost model can also be extended in a way to support deadline and/or a specific budget. It may be a cost comparison between private and public cloud [30] or a binary integer programming approach [9] as a part of a scheduler component. Extending the cost model [31] is also done by introducing the network cost. The size of datasets as an essential factor should be considered as it influences deadline as well as the total cost of the public environment to reduce the makespan and to achieve a minimum cost [32].

In contrast, scientific workflows are also considered to some extent but employing a data management scheme has been the neglected matter. In other words, datasets regardless of their locations are accessible at any time [6, 33]. Malawski et al., [33] presented a resource calculator for scientific applications on multiple cloud environments through a mathematical model and integer programming. The aim was providing a better resource management decisions for both users and providers. They divided the workflow into several layers with the objective to treat each layer as a set of tasks that could be executed in one instance. Then, they utilized the mixed integer programming to optimize the total cost via multiple constraints including, but not limited to, the deadline. However, due to lack of a data management scheme, the transferring cost had a negative influence on the cost optimization for their model.

In [6] the cost was also modeled for workflow scheduling by considering data accessibility time and queue time services on instances where resources might be limited or unlimited. The cost model was solved through a mixed integer nonlinear programming for better deployment of resources for workflow execution in a hybrid cloud environment. Lin et al., [34] proposed an online strategy for scheduling continuous workflow submission on hybrid clouds. Their approach aims to execute the workflow within the given deadline while keeping the public cloud utilization at a lower cost. Their method consists of leveraging a hierarchical iterative application partition (HIA) to cluster the workflow into a set of dependent tasks. It then collaborates with a hybrid scheduler to achieve a minimum cost for execution. Due to uncertainties of task arrival time, new arrival tasks were periodically clustered to be scheduled together. Then, via a hybrid scheduler task dispatching was performed by an iterative hierarchical algorithm to turn the tasks into sub-groups for better scheduling decisions. An internal queue scanner was also presented to manage public cloud dispatching when a task deadline could not be met in the private cloud. For the queue, different strategies were considered with the focus on selecting tasks which could be executed

quicker.

Lin et al., [35] also presented a scheduling approach for considering the QoS regarding meeting the deadline and minimizing the overall cost. They also considered the heterogeneity of cloud resources for a workflow structure by compacting a Critical Path (CP). The aim was to find two points as the right path for offloading purposes of reducing execution time through planing all unscheduled parents by greedily assigning to a best-fit instance. Before dispatching tasks for execution, they applied a pre-assessment on the workflow structure to extract and merge tasks that they have a directed edge; tasks whose the rear and front nodes have access to one node. This assessment aims to reduce the transferring time and reduce the complexity of the workflow.

Rahman et al., [36] developed an Adaptive Hybrid Heuristic (AHH) scheduling strategy to map tasks to the available instances through a genetic algorithm to stay with the deadline and budget while user constraints are also satisfied. A pre-schedule approach is applied to assign and distribute the given budget and deadline within the workflow structure to the task levels. A heuristic is then utilized to dispatch the tasks level-by-level based on the initial schedule dynamically. Combination of transferring and execution fee when datasets were already placed, were considered to model the cost of a Workflow Management System (WMS) to stay with user satisfaction criteria.

Luiz and Edmundo [37] with the help of Heuristic Path Clustering (HPC) tried to manage resource assignment to stay with the expected deadline. As a resource selection strategy, it aims to choose proper resources from the public cloud to be attached to the private environment for providing the sufficient processing power, i.e., cores, to stay with the expected execution time. Prioritized task groups as an initial scheduling step were used for instance selections. They are initially checked whether the private cloud can execute tasks within the deadline. If an expected deadline could not be met, it would be rescheduled to be assigned to the public cloud.

2.4 Summary

This chapter has reviewed studies about workflow scheduling in a cloud environment. Workflow scheduling has been intensively studied, and different approaches have been proposed for better makespan and/or in a cost-effective way. This cost could be increased if a cloud environment

TABLE 2.1: Summary of the literature and its comparison

Related Work	Cost Model	Deadline	Budget	Communication Cost	Data Management
Cunha et al., [22]	✓	✗	✗	✗	✗
Marcu et al., [23]	✓	✓	✓	✗	✗
Rahman et al., [36]	✓	✗	✗	✓	✗
Wang et al., [25]	✓	✓	✗	✓	✗
Chunlin and LaYuan [38]	✓	✓	✓	✗	✗
Charrada and Tata [39]	✓	✗	✗	✓	✗
Rezaeian et al., [28]	✓	✓	✓	✗	✗
Wei and Meng [40]	✗	✗	✗	✗	✗
Bossche et al., [31]	✓	✓	✓	✗	✗
Lin et al., [35]	✓	✓	✗	✓	✗
Yuan et al., [26]	✓	✓	✗	✗	✗
Abdi et al., [41]	✓	✓	✗	✓	✗
Bittencourt et al., [32]	✓	✓	✓	✓	✗
Bossche et al., [42]	✓	✓	✓	✓	✗
Zennen and Engel [24]	✓	✓	✗	✗	✗
Calatrava et al., [29]	✗	✗	✗	✗	✗
Hoseinyfarahabady et al., [27]	✓	✓	✓	✗	✗
Chu and Simmhan [20]	✓	✗	✗	✓	✗
Wang et al., [8]	✓	✓	✓	✗	✗
Bittencourt and Maderia [37]	✓	✓	✗	✓	✗
Malawski et al., [6]	✓	✓	✗	✓	✗
Malawski et al., [33]	✓	✓	✗	✓	✗
Lin et al., [30]	✓	✓	✗	✗	✗

would deal with the execution of data-intensive workflows. Thus, data management approaches have become more and more essential as a complementary role for workflow scheduling.

Table 2.1 represents the overview of workflow scheduling in the hybrid cloud context. It illustrates that although existing approaches address the cost reduction to some extent, they barely consider better data management that could complement the cost model to reduce the public utilization fees significantly.

3

Problem Statement

3.1 Lead-in

A workflow structure is presented as a Directed Acyclic Graph (*DAG*), $G = (V, E)$, to show the precedence-constrained in the form of a set of nodes $V = \{v_1, \dots, v_n\}$ as tasks and a set of edges $E = \{e_{ij}, \dots, e_{mn}\}$ as data dependencies between tasks. A task v_i is the parent task of v_j if v_j relies on the output of v_i . In this case, v_j is considered as the child task of v_i which leads to a data dependency in a way that v_j will not be performed until the parent v_i is executed. A workflow may have some tasks that do not have any parents. In this case, these tasks are called entry tasks. If a task does not have any children, it is considered an exit task.

The time required to transfer necessary data from a node v_i to another node v_j to run a task is considered as communication time c_{ij} . If tasks v_i and v_j are on the same node, the communication

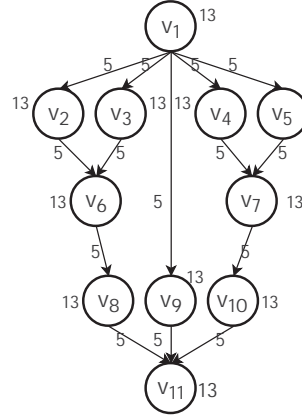


FIGURE 3.1: Example of a DAG representation of a workflow

time will be zero; otherwise, it will be a non-zero value if they run on different nodes.

An example of a workflow which is represented by a DAG is shown in figure 3.1. In this structure, there are 11 nodes that v_1 and v_{11} are the entry and exit nodes, respectively. When node v_1 is executed, the rest of nodes in the workflow will be performed based on the availability of their required inputs. Moreover, nodes are labeled with their computation cost which can be, for example, number of instructions. In the given DAG, edges are also tagged with their communication cost, e.g., the amount of transferred data in megabytes.

A private cloud is considered as a set of M heterogeneous or homogeneous resources $R = \{r_1, \dots, r_m\}$, with associated computation $C = \{c_1, \dots, c_m\}$, bandwidth $B = \{b_1, \dots, b_m\}$, and storage $S = \{s_1, \dots, s_m\}$. Values of computation (C) and storage (S) are chosen from \mathbb{R}^+ . The bandwidth (B) is a set of links $\in \mathbb{R}^+$ between resources $r_i \in R$ that can be represented as $l_{r_i} = \{l_{i,r_1}, \dots, l_{i,r_m}\}$. If the cloud is a homogeneous environment, instances will have the same computation, bandwidth, and storage configuration.

A public cloud is a set of N virtualized resources $U = \{u_1, \dots, u_n\}$ that users are charged in a pay-as-you-go manner. These resources have characteristics in terms of computation $C_u = \{c_{u1}, \dots, c_{un}\}$, bandwidth $B_u = \{b_{u1}, \dots, b_{un}\}$, and storage $S_u = \{s_{u1}, \dots, s_{un}\}$ that their values belong to \mathbb{R}^+ . Each resource $u_j \in U$ has a set of links $l_{u_j} = \{l_{j,u_1}, \dots, l_{j,u_m}\}$. Furthermore, each $u_i \in U$ has a cost per time unit $Cost_{u_j}$. A set $L_u^\tau \subseteq U$ is contemplated as leased resources that are obtained from the public cloud at the time of τ .

A hybrid cloud is a blend of private and public cloud environments with their available resources

which can be stated as $H = (\bigcup_{i=1}^m R) \cup (\bigcup_{j=1}^n U_j^n)$. The public cloud can provide virtually unlimited resources. Therefore, the size of resources in the public cloud could be unbounded unless the number of required resources would be demanded.

3.2 Definitions

There are variables in terms of cost, data dependency, and execution which are defined as follows:

3.2.1 Cost

Utilizing the public cloud is costly and as public instances come with different characteristics, their corresponding cost also differs. In other words, users are charged based on the amount of allocated storage $Cost(S)$, consumed bandwidth $Cost(B)$, and assigned computation $Cost(C)$ for an instance in the public cloud. $Cost(B)$ is determined based on the amount of input and output bandwidth consumption, thus, it can be rewritten as follows:

$$Cost(B) = |B(in)| + |B(out)| \quad (3.1)$$

The cost of execution on the private cloud is negligible but this value for the public cloud based on the type of instance it is assigned to would be calculated as follows.

$$Cost(Vm_i) = Cost(B_i) + Cost(S_i) + Cost(C_i) \quad (3.2)$$

3.2.2 Data Dependency

To define the dependency, suppose task i and j require datasets DT_i and DT_j , respectively. $Dependency_{ij}$ is defined as the following equation.

$$Dependency_{ij} = |DT_i \cap DT_j| \quad (3.3)$$

This dependency can also be extended and defined as internal DI_v and external DO_v dependency. The DI_v represents the correlation degree among datasets which are placed within an instance v , and DO_v shows the correlation degree between datasets within instance v and the other available

instances within an environment. They are defined as follows.

$$DI_v = \sum_{i=0}^{|DT_i|} \sum_{j=0}^{|DT_j|} Dependency_{ij} \quad (3.4)$$

$$DO_v = \sum_{i=0}^{|DT_i|} \sum_{j=0, j \neq v}^{|VM|} Dependency_{i,DT_j} \quad (3.5)$$

In equation 3.5, $Dependency_{i,DT_j}$ presents the correlation degree between task i of virtual machine(VM) v and DT_j of virtual machines other than v .

A workflow consists of t interdependent tasks that uses d input datasets and produces m intermediate data. A portion of intermediate datasets may not be used during the workflow execution, hence, the effective intermediate data is called m_e .

Each task t_i has a user-specified deadline D_i , required input data DS_i and intermediate datasets IDS_i . Each DS_i consists of some ds that might be flexible or fixed data. In a case of fixed dataset, it has to be placed at the designated virtual machine.

$$DS_i = \{ds_1, \dots, ds_n\} \quad (3.6)$$

3.2.3 Task Execution

Execution time of a task t_i is denoted as $Exec(t_i, Vm_j)$ in its instance Vm_j which is calculated based on the equation 3.7. In this equation, $MT(t_i, Vm_j)$ represents the amount of time that the task has to wait to get its required datasets in the target instance. $PT(t_i, Vm_j)$ shows how long Vm_j takes to process t_i .

$$Exec(t_i, Vm_j) = MT(t_i, Vm_j) + PT(t_i, Vm_j) \quad (3.7)$$

A task may have to wait to be executed if there are other tasks to be performed on a specific instance. The waiting time termed as $delay(t_i, Vm_j)$ based on the concurrent tasks could be determined by equation 3.8. In this equation, k represents the amount of tasks that are ahead of task t_c in the concurrent list. The deadline D_i of task i in that list is subtracted from its corresponding execution time of t_i to be understood that those tasks within the list of that instance are not behind their deadlines. If they are, the value would be negative and would show some tasks would miss their

deadlines on the same virtual machine.

$$delay = \sum_{i=0}^k D_i - Exec(t_i, Vm_j) \quad (3.8)$$

3.3 Problem Definition

The hybrid cloud model in our problem consists of a private cloud and a public cloud which contains M and N virtual machines, respectively. Instances within these environments are heterogeneous that each of which has their resources in terms of storage, computation, and bandwidth. They are called $R_i^{private}$ ($i \leq M$) and R_j^{public} ($j \leq N$), correspondingly.

The input of our algorithm is a workflow with t interdependent tasks which require d datasets. The execution of a task(t_i) on an instance(Vm_j) is influenced by the amount of required data-set DS_{t_i} . The presented problem intends to allocate tasks and their required datasets of a given workflow to the resources within a hybrid cloud environment; $t \rightarrow R_{private} + R_{public}$ such that maximize the private cloud utilization and reduce the number of offloading tasks to the public cloud which would lead to having a cost-effective strategy that meets the tasks deadline.

3.4 Summary

This chapter has provided preliminaries for task execution on virtual machines in a hybrid cloud environment. As a workflow is an interdependent structure, data dependency is defined to illustrate how tasks are related to each other. This dependency will be used in HSHC for the static phase. Determination of a task makespan and its cost on a public cloud are also discussed. In the end, the problem is defined as a mapping function to assign tasks to the hybrid cloud resources in a cost-efficient way while considering tasks deadlines.

4

Hybrid Scheduling for Hybrid Clouds

4.1 Lead-in

Hybrid Scheduling for Hybrid Clouds (HSHC) goes through two different phases to prepare the right location of datasets and tasks in a hybrid cloud environment. They are *static* and *dynamic*, respectively. The overall structure of the approach is shown in figure 4.1. The main reason for this division is to adequately prepare the combined environment for task execution in a productive way and avoid any unwanted data movement or lousy task scheduling for the available resources. The *static* phase aims to use

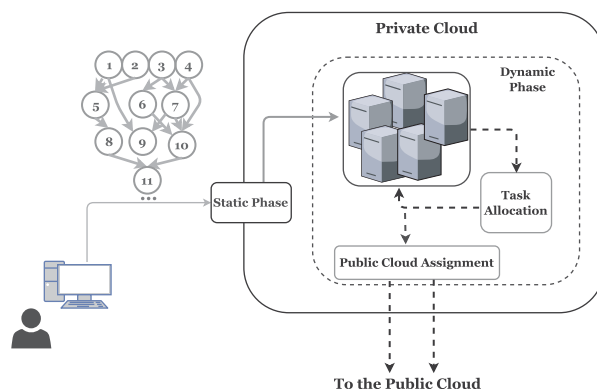


FIGURE 4.1: The overall HSHC structure

The *static* phase aims to use

an extended genetic algorithm approach to place optimally datasets and tasks at the same time based on the available virtual machines in a private cloud. In the next phase- *dynamic*- a dynamic programming approach is used which utilizes the previous phase output to find the best place of tasks in the combined environment based on the status of the private cloud workload.

4.2 Static Phase: A Genetic Algorithm Approach

The genetic algorithm (GA) is an optimization technique [43], which can be used to find the optimal (or near-optimal) solution. GA is a search method that is inspired by the theory of evolution. In this method, the solution space (population) is represented as a set of chromosomes, where each chromosome represents a feasible solution to the problem. Each chromosome consists of several variables (genes), which defines the structure of the problem. Each solution (chromosome) is assigned a fitness value, which represents how *fit* that chromosome is compared to other chromosomes.

4.2.1 Chromosome Structure

The genetic algorithm as an optimization approach is used to find the optimal placement of data and assign their corresponding tasks to the available virtual machines. In the core of algorithm, chromosome plays the important role as a solution representative. Therefore, for the current problem the structure shown in figure 4.2 is used that presents a combination of datasets and tasks at the same time to find the optimal allocation of tasks and data concurrently. The cell i consists of a task-set TS_i , data-set DS_i , and an instance Vm_i .

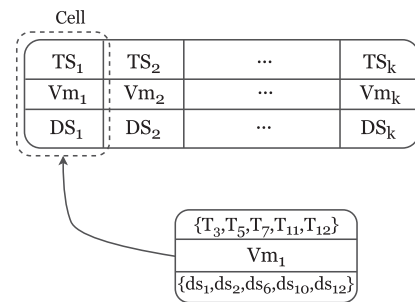


FIGURE 4.2: The chromosome structure.

4.2.2 Fitness Function

Evaluation of a chromosome is done through a fitness function which considers data dependency and availability, controlling parameters, and the *delay* to choose the most eligible solution as a result.

4.2.2.1 Data Dependency and Availability

The main objective of our approach 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 datasets for this task execution. We denote the percentage of available datasets for task i at virtual machine j by $ava(t_i, VM_j)$. Moreover, for any given virtual machine (j), the dependency between datasets assigned to this VM and datasets located at different VMs must be minimized. In other words, the dependency between datasets assigned to the same VM must be maximized. For a virtual machine (j), we refer to the data dependency between the datasets assigned to this VM as DI_j and the data dependency between these VM datasets and other VMs datasets as DO_j which were discussed in chapter 3, equations 3.4 and 3.5.

4.2.2.2 Controlling Parameters

To ensure the feasibility of a solution, we use the variable ck_f to check if the assignment for the fixed datasets 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 datasets from its current VM. Moreover, to pick up the best VMs, we use a variable termed P_{ratio} . The value for the variable reflects the overall computational power for the selected VMs. For example, among M private virtual machines, there are four unique computation capabilities as CP_1, CP_2, CP_3, CP_4 , which are sorted in descending order with the corresponding values 1, 0.75, 0.5, 0.25. This ratio represents how robust the selected instances for task execution are. The higher the ratio, the better the solution; this ratio is calculated as follows:

$$P_{ratio} = \sum_{i=0}^M s_i \quad (4.1)$$

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 virtual machine, the workflow structure has to be considered. In other words, tasks are monitored to be realized when they will be available for execution under their required intermediate datasets.

4.2.2.3 Task Delay Within an Instance

Then, they are categorized, and their execution time is examined against their assigned deadline. If within an instance, there are tasks that do not need any intermediate datasets, they will also influence concurrent tasks. The lesser the delay, the better the solution is provided. The process is shown in figure 4.3. Once the priorities are ready, the amount of time they would be behind their defined deadline is evaluated.

The total amount of delay for a solution based on equation 3.8 in chapter 3 is obtained by the following:

$$Tdelay = \sum_{i=0}^M delay(vm_i) \quad (4.2)$$

where $delay(vm_i)$ represents the total delay at virtual machine i . 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.

$$\begin{aligned} fitness = & (p_r \times ck_r \times ck_f \times \sum_{j=0}^M vu_j) \times (\sum_{i=0; j=0}^{|T|, M} ava(t_i, VM_j) \\ & \times (\sum_{i=0; j=0}^{|T|, M} DI_j - \sum_{i=0; j=0}^{|T|, M} DO_j) - \sum_{j=0}^M Tdelay_j) \end{aligned} \quad (4.3)$$

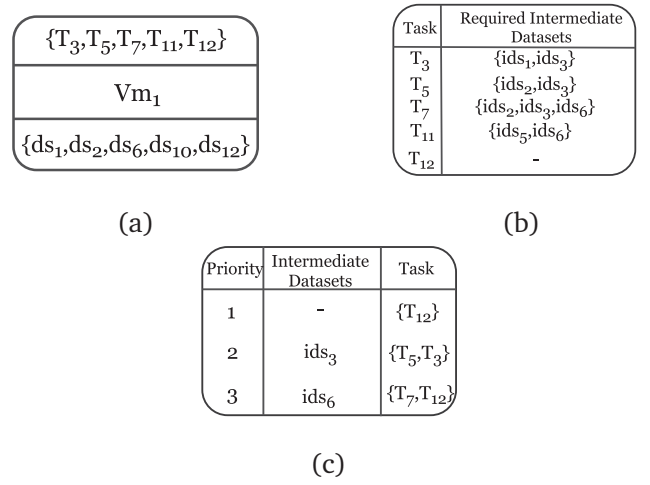


FIGURE 4.3: (a) A cell, (b) tasks with sorted required intermediate datasets, and (c) the priority of tasks within the virtual machine.

The first part of the fitness function deals with the controlling parameters. The parameters ck_r and ck_f ensure the solution meets criteria in terms of fixed dataset constraint and the feasibility of transferring task i datasets to the designated instances, respectively. Variables p_r and vu_j aim to select better solutions in terms of computation capability while the selected virtual machines are properly in charge of hosting data-sets or task-sets.

The second part mainly facilitates discovering the most suitable solutions which consist of a mixture of data availability, dependency, and the tasks delays inside an instance. If the availability of highly correlated data inside a virtual machine would exist, less delay for tasks would be attained.

4.2.2.4 Genetic Algorithm Operators

The new solutions are produced based on the available ones. Due to the combined structure of the chromosome, the normal *crossover* operation cannot be applied for generating new solutions. Thus, the newly proposed crossover called *k-way-crossover* is introduced in figure 4.4. 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 be increased, decreased, or fixed. If either the best or worst cell has any fixed datasets, they and their corresponding tasks will remain within the cell.

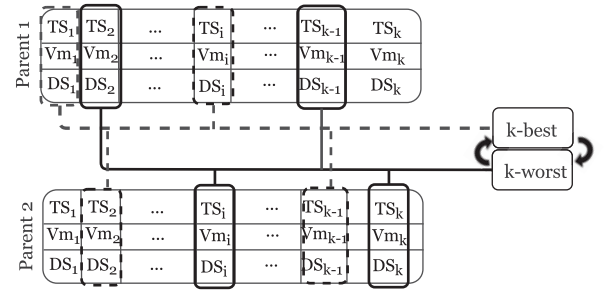


FIGURE 4.4: k-way-crossover.

Concurrent-rotation is introduced to mutate a solution. In figure 4.5 the structure is shown. It would create four different ways to mutate a chromosome. For each cell within a solution, the task and dataset with least dependency would be selected and by the direction- *clockwise* or *counterclockwise*- they would be exchanged with the next or previous cell, correspondingly. If the least

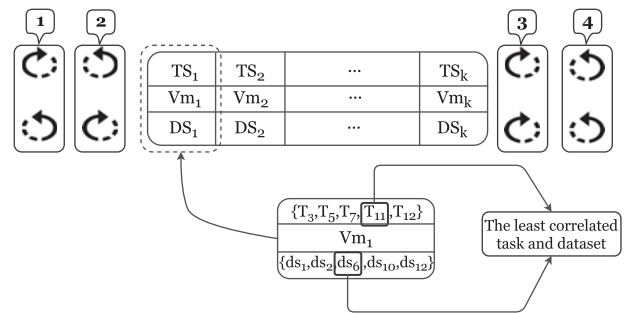


FIGURE 4.5: The concurrent-rotation process.

Algorithm 1: Static Algorithm

Data: private instances list, task-list, dataset-list, generation size, mutation probability

Result: The optimal placement of datasets and assignment of tasks

```

1 Prepare the initGeneration;
2 Evaluate initGeneration by equation 4.3;
3 while either the optimal is not found, or generation size is not exceeded do
4     apply tournament-selection;
5     apply k-way-crossover;
6     apply concurrent-rotation mutation;
7     evaluate the new solution;
8     add the new solutions to the population;
9 end
10 Rank the population and return the best one;
  
```

dependency is related to a fixed dataset and its corresponding task, the next least task, and dataset would be selected.

4.2.3 Static Algorithm

The *Static* algorithm represents the *static* phase which is shown in algorithm 1. The outcome of this phase will be used to initially place datasets and assign their corresponding tasks to the most suitable virtual machines inside a private environment. This phase would provide better a situation during the *dynamic* phase in terms of execution time as well as data transfer.

Typically, the initial set of chromosomes is created randomly. Furthermore, the number of evolutionary iteration on the initial population is pre-defined. In each round, some of the chromosomes will be chosen via a selection strategy, e.g., tournament selection. Chromosomes with higher fitness values via equation 4.3 are more likely to be nominated. These selected chromosomes will be passed to a *k-way* crossover operator, which generates new chromosomes (offspring). In addition, the *concurrent* mutation operator is also used to generate new chromosomes. At the end of each iteration, the fitness value for each chromosome is evaluated, and the ones with highest

values survive.

4.3 Dynamic Phase

The *static* outcome is used to place available datasets and assign corresponding tasks to the designated virtual machines. As the previous step is an offline phase, the situation during the workflow execution would be changed, and the cloud environment is required to be reevaluated. In other words, some of the tasks would be needed to either assign to a different virtual machine or to be offloaded to the public cloud environment for execution. The *dynamic* phase is split up into two sub-phases to increase the performance of the private environment as well as reducing the offloading costs to the public cloud.

4.3.1 In The Private 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, where these tasks must be executed. The *flexible* set contains the tasks that can be executed at any VMs, as long as necessary criteria like deadline and storage are met. Our algorithm is mainly concerned with scheduling the flexible tasks. We start by calculating the available capacity and workload for the current *ActiveVMs*; instances that are processing tasks. This is used to determine the time in which these VMs can execute new tasks. During the execution of our algorithm, 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 the *FTMatrix* (line 1).

The objective of algorithm 2 is to determine the identity of the task that can be assigned in each round (for loop line 8). The allocation is established by identifying the task ($t_i \in T$) and the VM ($vm_i \in VM$) such that $FT(t_i, vm_i)$ value 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.

In the first round, we determine the expected finishing time for every task (FT) at each VM while it considers required data availability. For any given task and VM, this is calculated by adding the execution time for this task to the time where this VM becomes available to execute a new task (line 16). After the execution of the first round, we examine where the task with the lowest $FT(t_i, vm_i)$ value can be assigned without violating its deadline and required storage. If this assignment is confirmed, we update the values in the $FTMatrix$ to state that vm_i will be responsible for executing task t_i . If this assignment violates t_i deadline and/or storage, this task will be added to the offloading list. In both cases, t_i will be removed from consideration on consecutive rounds.

Thereafter in each round, the expected finishing time for every task on a specific VM is determined based on the identity of the task and the VM. Assuming that we are currently processing task (t_j) and virtual machine (vm_j), where the last confirmed assignment in the previous round is represented by t_c to vm_c , if vm_j is actually vm_c , we increase the expected finishing time of task t_j on vm_j , since this VM is responsible for executing t_c . In other situations, the value of the task FT will stay the same. By adopting this strategy, we aim to maximize the number of tasks to be executed in the private cloud.

The procedure *findMinValue* checks the $FTMatrix$ and evaluates it against the current storage of the corresponding virtual machines. What it returns is the *index* of the task and the corresponding instance for offloading purposes. If it fails to find the indexes, the task would be added to the *offloadingList*.

4.3.2 In The Public Cloud

All of the offloaded tasks will be scheduled to be executed in the public cloud, and algorithm 4 shows the steps in this scheduling process.

Scheduling these tasks uses a strategy similar to the private cloud scheduling. We start by

Algorithm 2: Tasks allocation algorithm**Data:** F (flexible tasks), NF (non-flexible tasks), and private cloud $VMs = idleVMs \cup activeVMs$ **Result:** S (final schedule)

```

1 initialize  $FTMatrix$  with size  $|F| \times |VMs|$ 
2  $l_{off} \leftarrow null, V \leftarrow null$ 
3 Assign  $NF$  tasks and initialize  $Storage[|VMs|]$ 
4 for  $vm_i \in VMs$  do
5   | update  $Storage[vm_i]$  and calculate  $vm_i$  workload
6 end
7 for  $i \leftarrow 0$  to  $|F|$  do
8   | for  $t_k \in F \setminus (l_{off} \cup V)$  do
9     | for  $vm_j \in VMs$  do
10      | if  $i = 0$  then
11        | |  $FTMatrix[t_k][vm_j] \leftarrow Exec(t_k, vm_j)$ 
12        | end
13      | else
14        | if  $vm_j = vm_c$  and  $t_k \neq t_r$  then
15          | |  $FTMatrix[t_k][vm_j] \leftarrow FTMatrix[t_k][vm_j] + Exec(t_k, vm_j)$ 
16          | end
17        | end
18      | end
19    | end
20  end
21   $[T_r, Vm_c] = FindMinValue(FTMatrix, Storage)$ 
22   $S = DeadlineConstraintCheck(T_r, Vm_c)$ 
23   $S \leftarrow S \cup publicCloudAssignment(l_{off})$ 
24 end

```

Algorithm 3: Deadline Constraint Check

Data: T_r (corresponding task index in the task-set), Vm_c (corresponding Vm index in the private cloud VMs)

Result: S (final schedule)

```

1 if  $t_r$  deadline is satisfied then
2   | assign  $t_r$  to  $vm_c$ 
3   | Update  $FTMatrix$  and add  $t_r$  to  $V$ 
4   | add  $[T_r, Vm_c]$  to  $S$ 
5 end
6 else
7   | add  $t_r$  to  $l_{off}$ ,  $t_r \leftarrow null$ 
8 end
9 Return  $S$ 

```

calculating the workload for all VMs. In this algorithm, for each task (t_i) and virtual machine(vm_i), we maintain a value (DT) that represents the gap between this task deadline (D_i) and the expected finishing time for this task on vm_i (FT). In situations where this value is less than zero, this task cannot be executed on this VM. These values are stored in the $DTMatrix$ (line 1).

In each round, we identify the task ($t_i \in T$) and the virtual machine ($vm_i \in VM$) such that $DT(t_i, vm_i)$ value is the lowest possible in the $DTMatrix$. This is established to minimize the cost of execution, since low-speed VMs that satisfy the task deadline will result in a lower gap, compared to high-speed virtual machines. Once they are identified, we perform this assignment and remove this task from consideration in consecutive rounds. Then in each round, the execution gap (DT) for each task on a specific VM is determined based on the identity of the task and the VM in the last confirmed assignment. Assuming that we are currently processing task (t_j) and VM (vm_j), in this situation, if vm_j is the same VM used in the last confirmed assignment, we modify the execution gap of t_j on vm_j to measure the processing time of the last assigned task.

Algorithm 4: PublicCloudAssignment**Data:** l_{off} and public cloud $VMs = idleVMs \cup activeVMs$ **Result:** $C(\text{schedule on cloud})$

```

1 initialize  $DTMatrix$  with size  $|l_{off}| \times |VMs|$ 
2  $V \leftarrow null$ 
3 for  $vm_i \in VMs$  do
4   update  $Storage[vm_i]$ 
5   calculate  $vm_i$  workload
6 end
7 for  $i \leftarrow 0$  to  $|l_{off}|$  do
8   for  $t_k \in (l_{off} \setminus V)$  do
9     for  $vm_j \in VMs$  do
10      if  $i = 0$  then
11         $DTMatrix[t_k][vm_j] \leftarrow D_{t_i} - Exec(t_k, vm_j)$ 
12      end
13      else
14        if  $vm_j = vm_c \text{ and } t_k \neq t_r$  then
15           $DTMatrix[t_k][vm_j] \leftarrow DTMatrix[t_k][vm_j] - Exec(t_k, vm_j)$ 
16        end
17      end
18    end
19  end
20 end
21  $[T_r, vm_c] = MinCost(DTMatrix, V)$ 
22 assign  $t_r$  to  $vm_c$ 
23 Update  $DTMatrix$  and add  $t_r$  to  $V$ 
24  $add[t_r, vm_c] to C$ 
25 end

```

4.4 Summary

This chapter has illustrated the core components of HSHC as *static* and *dynamic*. The first phase with the help of an extended GA mainly deals with data placement and assign tasks to the most suitable virtual machines. The fitness function is elaborated to select the most suitable solutions. It consists of the following parts: controlling parameters, variables and a combination of data dependency and task delay in instances. Controlling parameters such as ck_f and ck_r are defined to check immovability of datasets and storage requirement for the task and data at the designated host, respectively. Variables Vu_j and p_r are explained for determination of instances regarding hosting a data-set or task-set and also selecting powerful instances for execution as much as possible, correspondingly. The task delay and data dependency are criteria to evaluate how well tasks are assigned to instances and how their corresponding data are placed.

The *dynamic* phase is responsible for task dispatching in the hybrid cloud during actual execution. It utilizes two algorithms to correctly assign tasks to the best environment based on the storage and the deadline. In fact, each environment has its scheduler to dispatch the tasks to the most capable instances while the constraints are considered. If the private cloud could not execute some tasks, they would be offloaded to the best performance to cost ratio instances in the public cloud.

5

Evaluation

5.1 Lead-in

To evaluate HSHC, several experiments are conducted, and they are mainly divided into two different sections as *heterogeneous* and *homogeneous* for the private environment. This is due to policies which may be followed by installing new devices or setting up the same devices for the internal cloud that may create different infrastructure. Also, baseline policies that are used and implemented are *FCFS* and *AsQ*. The former is the traditional scheduling approach for assigning tasks to instances through a queue-based system that the first-assigned task to the instance would be executed early. The latter intends scheduling deadline-based applications in a hybrid environment with a cost-effective approach, however, without a specific data management strategy. In fact, it is an adaptive scheduling approach with QoS satisfaction to maximize private cloud utilization and reduce the task response time. It also utilizes a cost strategy for dispatching tasks to the public

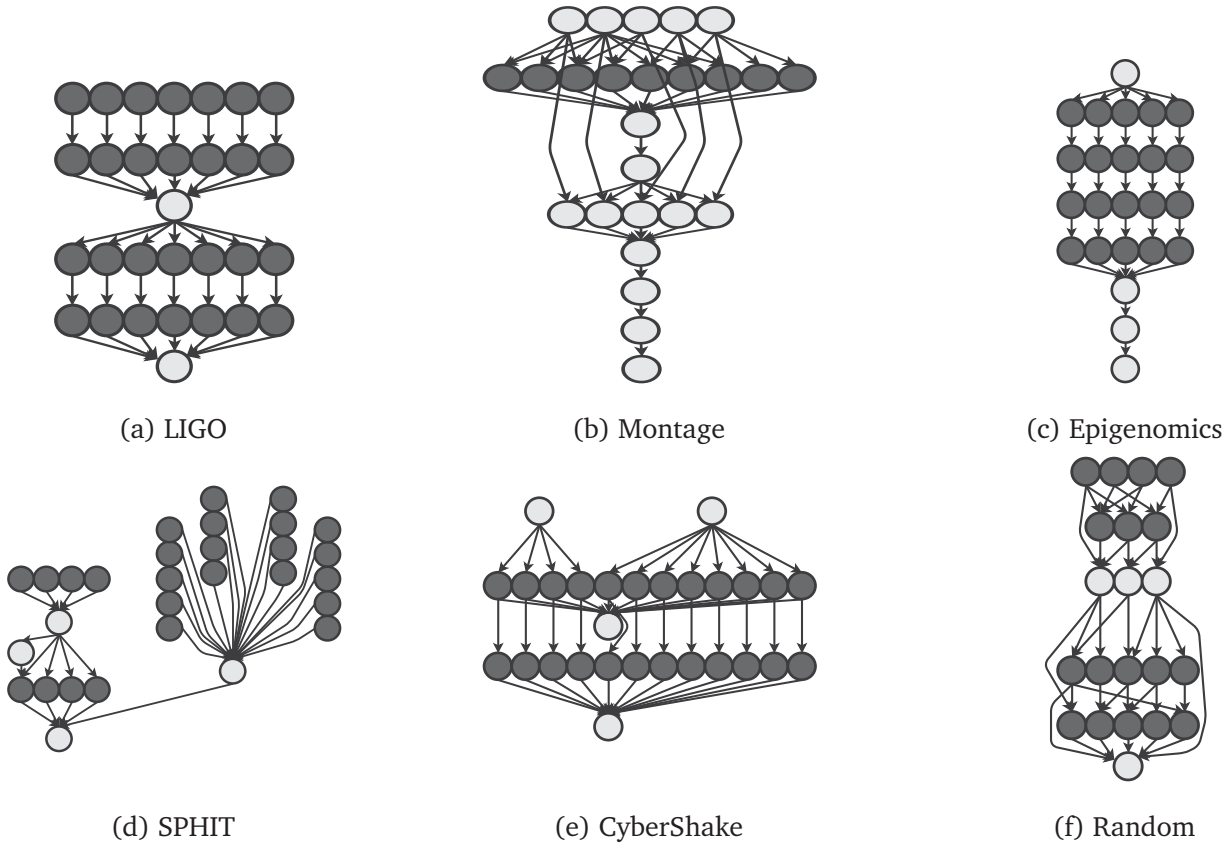


FIGURE 5.1: Workflow types

cloud.

5.2 Workflow Types

There are five types of scientific workflows¹ which are presented in figure 5.1. Figure 5.1(a) as **Laser Interferometer Gravitational Wave Observatory (LIGO)** is a gravitational-wave detectors network to detect and determine gravitational waves as anticipated by Einstein in his theory of gravity. Features of this workflow are less in the number of tasks as well as dealing with terabytes of data to produce the useful results. Figure 5.1(b) describes the **Montage** workflow as an astronomy application which forms large image mosaics of the sky and is recognized as a data-intensive application. **Epigenomics** that is presented in figure 5.1(c) shows a data-processing pipeline that

¹<https://confluence.pegasus.isi.edu/display/pegasus/WorkflowGenerator>

TABLE 5.1: Hybrid cloud simulation parameters

Cloud Environment		Hosts		Instances	Price(\$/h)		
Public cloud		3 hosts		20	Small	Medium	Large
	Small	Medium	Large		Default	Double	Triple
	4 inst.	8 inst.	8 inst.				
Private cloud		1 host		20	Free		

facilitates the execution of the various genome-sequencing operations automatically. SiphT which is depicted in figure 5.1(d) is a bioinformatics project that is administrating an extensive search for small raw RNAs that organize several processes such as secretion or virulence in bacteria. CyberShake is shown in figure 5.1(e) is a seismology application that computes probabilistic seismic hazard curves for geographic sites in the Southern California region.

Random workflows presented in figure 5.1(f) is created based on an arbitrary structure that considers multiple inputs and output connections for making the workflow more interdependent. This inter-dependency causes the relationship between tasks to become more dependent on multiple outputs of the other tasks within the structure.

We utilize scientific workflows such as Montage and LIGO [37] as data-intensive jobs and also random workflows with user-defined deadlines as the input for our simulation.

5.3 Environment Setup

The simulation environment is composed of a private cloud and a public environment which is shown in table 5.1. The cost of public cloud utilization is based on Amazon EC2 price list¹ that for the small instance as a default cost is \$0.023 per hour, and \$0.1 per GB for computation and network, respectively. For the heterogeneous private environment, computation capabilities are 7 instances with 250MIPS, 4 instances with 500MIPS, 4 instances with 750MIPS, and 5 instances with 1000MIPS. For the homogeneous private cloud, the computation capability is considered 250MIPS. For the public environment and based on the defined hosts, they are 2500MIPS, 3500MIPS, and

¹<https://aws.amazon.com/ec2/pricing/on-demand/>

4500MIPS, respectively.

For the *static* phase, the initial population size is 50, and max generation is 500. The probability of the mutation after several experiments is considered close to 0.2. The random workflow is created based on a hierarchical structure that has 85 datasets, and their size in MB is chosen from [64-512]. The input and output degree are chosen from [3-8]. Due to the different type of datasets within the cloud environment, the results are extracted based on the 10% fixed datasets for the random workflow. The results are reported in average based on 10 simulation runs in CloudSim [44].

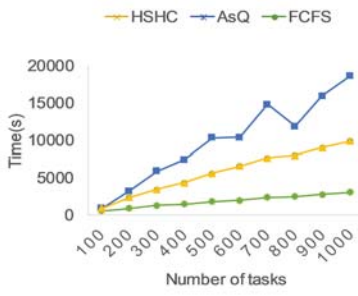
5.4 Results and Discussion

The results of our proposed approach are compared with *FCFS* and *AsQ* above based on the number of tasks that their deadlines are met, the time that is spent for data movements, execution time, and the cost of utilizing the public cloud.

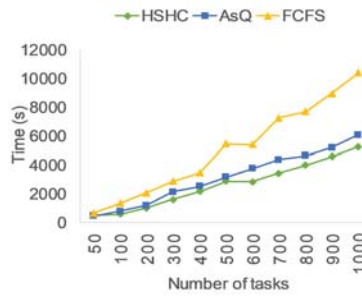
5.4.1 A Heterogeneous Private Environment

Our results in figure 5.2 agree with the fact that maximizing the private cloud utilization plays an essential part in a hybrid cloud structure. Figure 5.2(a-c) shows the overall execution time for scientific workflows as well as random workflows in the heterogeneous environment. It approves that *HSHC* outperforms the other scheduling strategies. One reason which implicitly affects the average execution time is that *AsQ* and *FCFS* look for the available instances despite their capabilities.

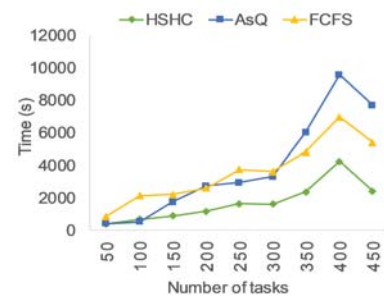
As different computation capabilities exist in this environment, *HSHC* tends to select powerful instances based on the *static* phase. In fact, it looks for available instances on the private cloud based on the deadline and storage constraints. This leads to having minimum execution time even when the number of tasks is increased as tasks could be performed quicker, and their results could be transferred to the private cloud. Furthermore, *AsQ* and *FCFS* have different execution time based on the type of workflows. Although *HSHC* in figure 5.2(a-b) has achieved the minimum time,



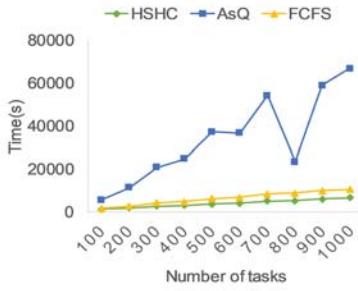
(a) LIGO in Hetero.



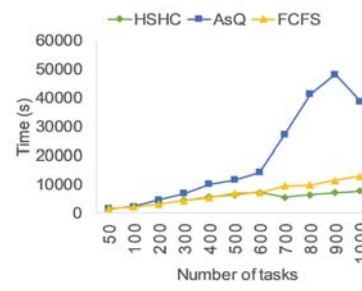
(b) Montage in Hetero.



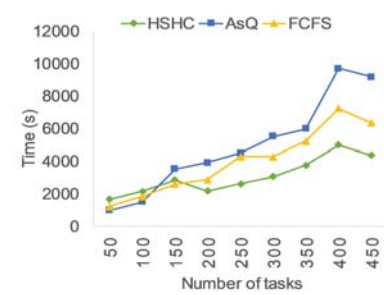
(c) Random in Hetero.



(d) LIGO in Homo.



(e) Montage in Homo.

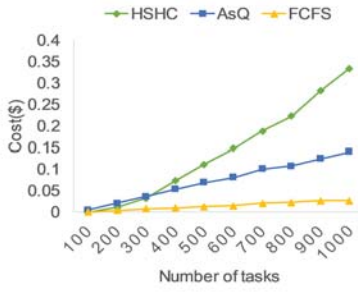


(f) Random in Homo.

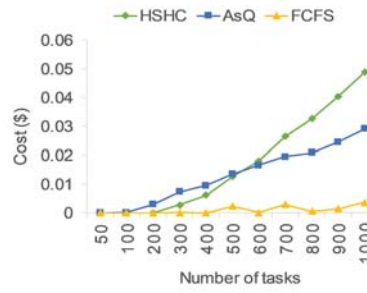
FIGURE 5.2: Average execution time: (a) LIGO in hetero. (b) Montage in hetero. (c) Random in hetero. (d) LIGO in homo. (e) Montage in homo. (f) Random in homo.

the second minimum value for *FCFS* and *AsQ* have happened for LIGO and Montage, respectively. This difference is due to the structure of workflows they have. They are both data-intensive, but they have the different number of concurrent tasks for execution based on their required intermediate datasets. It could also be explained as dispatching those tasks that are either behind their deadline or unable to retrieve their required datasets at the designated virtual machines. In figure 5.5(a-c) increasing the number of tasks would require extra facilities which the private cloud could not provide when the workload is high, therefore, it explicitly represents that dispatching tasks to the public cloud is the ideal approach to meet the deadline constraint.

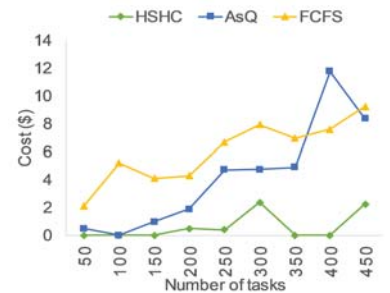
Moreover, it is also understood that the structure of workflow could have an indirect impact on offloading tasks to the public cloud. In other words, 5.5(c) represents that for different random workflow structures, different dispatching numbers are expected. This could be explained from different perspectives: (1) the private cloud could handle all concurrent tasks that become activated due to their required intermediate datasets and (2) the level of concurrency it presents differs from a workflow to another workflow. This concurrency also affects the private instances workload at



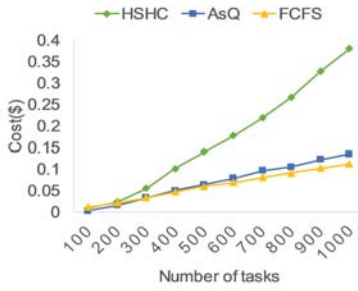
(a) LIGO in Hetero.



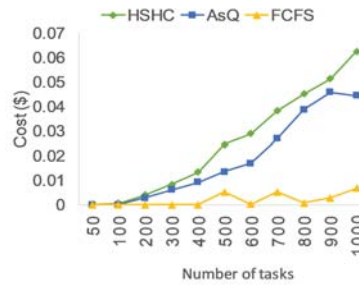
(b) Montage in Hetero.



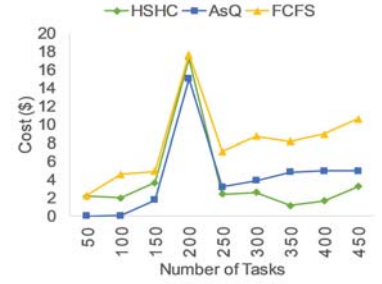
(c) Random in Hetero.



(d) LIGO in Homo.



(e) Montage in Homo.

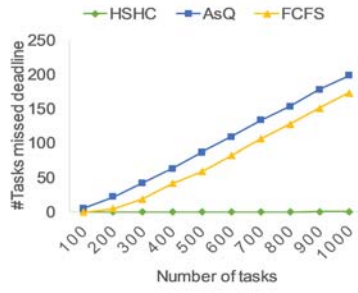


(f) Random in Homo.

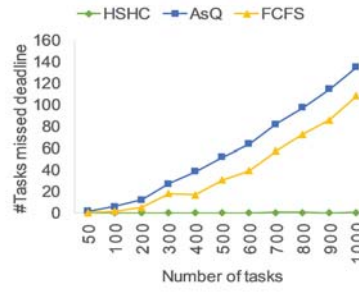
FIGURE 5.3: Average cost of public cloud utilization: (a) LIGO in hetero. (b) Montage in hetero. (c) Random in hetero. (d) LIGO in homo. (e) Montage in homo. (f) Random in homo.

different execution stages which might be high or low. Hence, having all the execution within a private environment can cause a few tasks to be sent off to the public cloud.

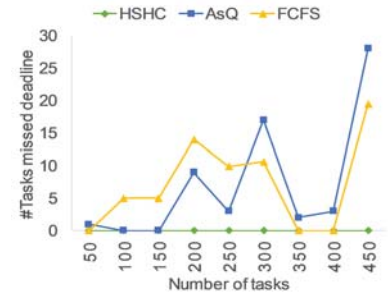
Dispatching to the public environment also explains the amount of transferring time that has to be spent on the offloaded tasks. In other words, figure 5.6(a-c) for the LIGO and Montage show how *HSHC* leverages public cloud resources to meet the deadline and/or storage constraints. Figure 5.4(a-c) can explicitly explain that to avoid missing tasks deadlines, they would have to be offloaded to the public cloud. As a result, *HSHC* meets all the deadline, however, *FCFS* and *AsQ* could not execute tasks within the expected time frame. On the one hand, the dynamic nature of *HSHC* for virtual machine selections, and the other hand, passive strategies followed by *FCFS* and *AsQ* have created differences regarding missed deadlines. In fact, assigning tasks to available instances either on the private or public cloud would not guarantee that tasks could be performed within the expected time. For example, if the queue-based structure of *FCFS* would not be able to have the right vision for the tasks within the queue, it would lead to dispatching many tasks to an instance for execution which would end up starving tasks at the back of the queue. Therefore, it



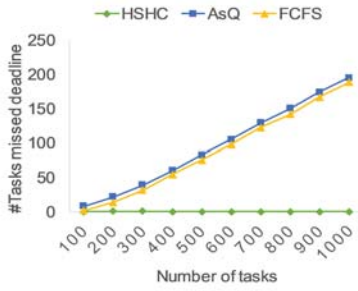
(a) LIGO in Hetero.



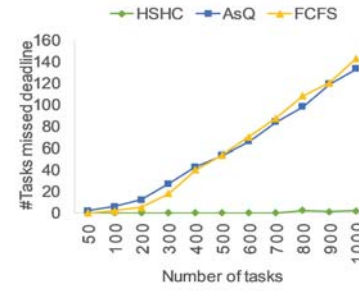
(b) Montage in Hetero.



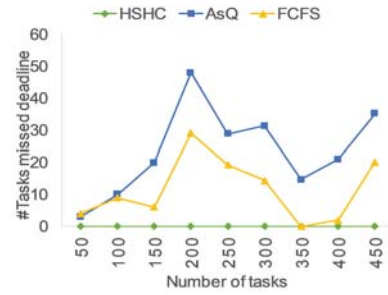
(c) Random in Hetero.



(d) LIGO in Homo.



(e) Montage in Homo.



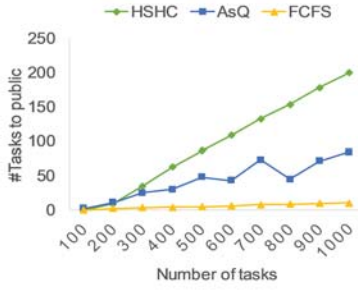
(f) Random in Homo.

FIGURE 5.4: Average missed deadlines: (a) LIGO in hetero. (b) Montage in hetero. (c) Random in hetero. (d) LIGO in homo. (e) Montage in homo. (f) Random in homo.

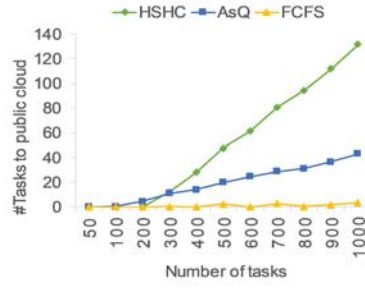
explains why these strategies have the minimum number of dispatched tasks to the public cloud for the LIGO and Montage; figure 5.5(a-b). However, it has the maximum number of dispatched tasks in figure 5.5(c) for the random workflows due to its different structure.

From a cost perspective, it also explains why *HSHC* achieves relatively higher costs in figure 5.3(a-b) for scientific workflows and a lower costs for random workflows in figure 5.3(c). The higher the offloading rate, the higher the cost of public cloud utilization. Therefore, considering the deadline as well as the ability of the private cloud caused to execute tasks in the public cloud which can directly impact on the utilization costs. However, the offloading process may be influenced by the workflow structure as for random workflows in figure 5.3(c) the costs have been the highest for the other methods.

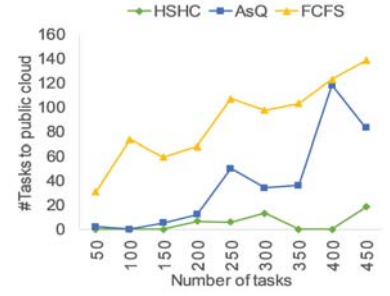
Therefore, *HSHC* by the right placement of datasets as well as the proper assignment of their corresponding tasks outperformed *AsQ* and *FCFS* in both random and scientific workflows.



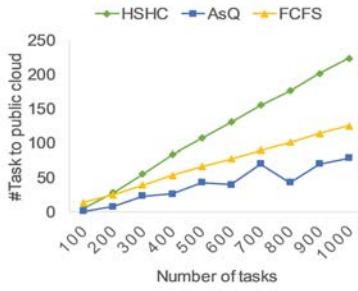
(a) LIGO in Hetero.



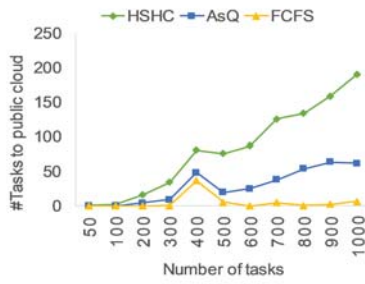
(b) Montage in Hetero.



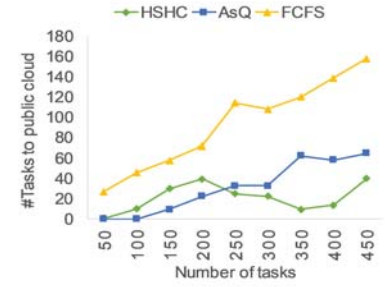
(c) Random in Hetero.



(d) LIGO in Homo.



(e) Montage in Homo.



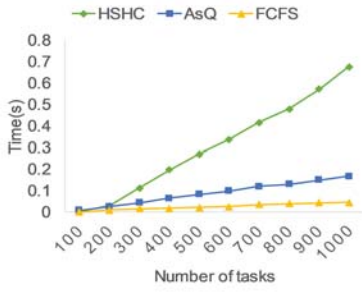
(f) Random in Homo.

FIGURE 5.5: Average dispatched tasks to the public cloud: (a) LIGO in hetero. (b) Montage in hetero. (c) Random in hetero. (d) LIGO in homo. (e) Montage in homo. (f) Random in homo.

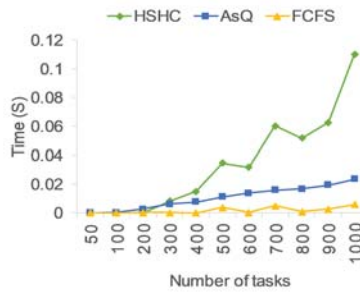
5.4.2 A Homogeneous Private Environment

Instances 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. The low computation capabilities will cause the higher execution time which is shown in figure 5.2(d-f). However, utilization of public instances would lead to having minimum execution time as it would reduce the average execution time but would increase the cost.

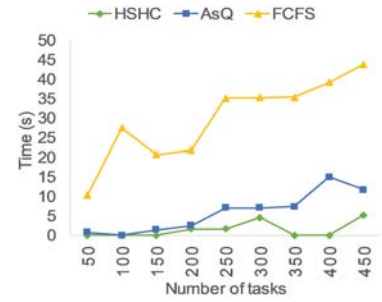
It is referred from the figure that the baseline minimum and maximum time for all scheduling strategies have been increased, but *HSHC* has achieved the minimum one. As task execution would take considerable time in the private cloud, dispatching more tasks to the public cloud is anticipated. In figure 5.5(d-e) dispatched tasks for LIGO and Montage are higher than for *HSHC* approach in comparison to *FCFS* and *AsQ*. However, this value for the random workflows in 5.5(f) is the lower one for *HSHC* in comparison to other strategies due to its different workflow structure.



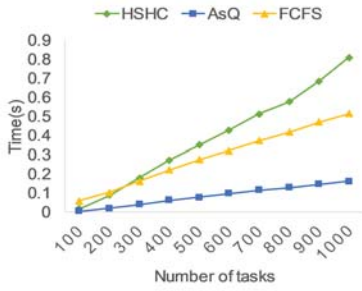
(a) LIGO in Hetero.



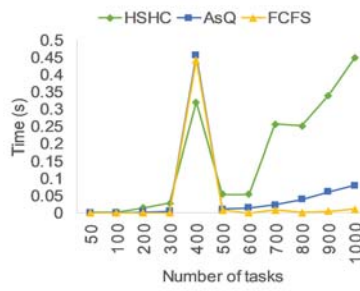
(b) Montage in Hetero.



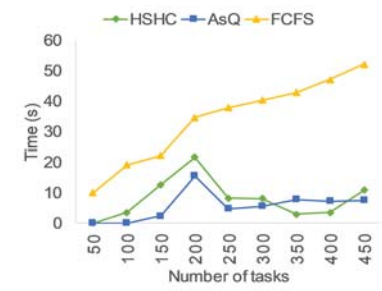
(c) Random in Hetero.



(d) LIGO in Homo.



(e) Montage in Homo.



(f) Random in Homo.

FIGURE 5.6: Average transferring time to the public cloud: (a) LIGO in hetero. (b) Montage in hetero. (c) Random in hetero. (d) LIGO in homo. (e) Montage in homo. (f) Random in homo.

The higher offloading rate has an explicit impact on the cost. Therefore, it is expected that *HSHC* would achieve relatively higher costs for the scientific workflows and the lower cost of the random workflows. In figure 5.3(d-e) our proposed approach has a relatively higher cost than *FCFS* and *AsQ*. This higher cost can be explained in this way that *HSHC* aims to execute tasks within the expected deadline; thus, it is obvious that the private cloud would be helped more by the public cloud. Thus, *HSHC* has executed tasks within the user-defined deadline in figure 5.4(d-f). Despite the fact that the cost for *FCFS* is almost zero for Montage which is shown in figure 5.3(e) it could not achieve meeting deadlines in figure 5.4(e) and has a slightly higher rate of the missed deadline.

5.5 Summary

In this chapter, *HSHC* is evaluated based on random workflows and data-intensive workflows such as Montage and LIGO in a heterogeneous private cloud environment. The results for a

homogeneous private cloud is also extracted due to policies which may be followed by installing new devices for the internal cloud that may create different infrastructure.

Our results are compared with scheduling methods *FCFS* as a traditional strategy and *AsQ* as cost-effective approach. The results are extracted based on execution time, cost of public cloud utilization, transferring time, the number of tasks that have missed their deadlines, and the number of tasks that are dispatched to the public cloud. We have achieved comparatively better results regarding maximizing the private cloud utilization for task execution and meeting task deadlines.

6

Conclusion

6.1 Lead-in

Cloud computing is a widely adopted technology that has created a robust platform to help developers to thrive newly cloud-based services and applications like Spotify¹ or Netflix². Regardless of the public cloud popularity, there are still small and medium organizations which leverage owned infrastructure for internal usage. Although it might be beneficiary to utilize such administrated infrastructure, drawbacks like incapability of dealing with the workload surges have been the open issue. A solution to this problem is to use a public cloud next to a private cloud to address insufficiency of resources when required.

This research has provided us with open issues in the hybrid cloud when data and computation

¹<https://www.spotify.com/>

²<https://www.netflix.com/>

intensive applications may be deployed in this environment. We have studied workflow scheduling from a data placement prospective to efficiently schedule interdependent tasks through reducing unnecessary data movements between clouds. We have proposed HSHC through a dynamic programming approach and with the help of an extended genetic algorithm. In this chapter, the future directions for extending the current study is presented. We also summarize the research work and highlight our significant contributions.

6.2 Conclusion

Scheduling is the key pivot for the execution of workflows in a cloud environment to achieve better resource utilization which has been extensively studied. The thesis has presented HSHC for addressing scientific workflow scheduling in a hybrid cloud based on a data placement approach to properly manage data locality in the hybrid cloud to avoid unnecessary movements and eventually reduce public cloud utilization cost.

This thesis has discussed HSHC in [chapter 4](#). HSHC is a two-phase approach that is referred to as *static* and *dynamic*. The first phase leverages an extended genetic algorithm to deal with data locality. It uses a chromosome structure as the combination of task-set and data-set within a designated instance which is discussed in [section 4.2.1](#). To select the proper candidates a fitness function- [section 4.2.2](#)- is presented based on the following criteria: (1) inner and outer virtual machine data dependency, (2) datasets availability for the corresponding tasks, (3) controlling factors for instance selections and constraint checks, and (4) the concurrent delay inside instances which assigned tasks may have. The extended algorithm also utilizes customized operators known as *k-way crossover* and *concurrent mutation*. The former swap the worst cells (genes) within a solution with the best ones from the other solution. The latter rotates the least correlated task or data with the adjacent cells based on the rotation direction. The output of this algorithm provides better data management and tasks allocation to the available instances.

The second phase consists of two algorithms which deal with the situation of the hybrid cloud during workflow execution. The first algorithm termed as *Task Allocation* deals with scheduling tasks within the private cloud based on their deadline and required storage. It leverages a dynamic programming approach such that it increases the private cloud utilization. The other algorithm

referred to as *Public Cloud Assignment* copes with selecting the most performance to cost ratio in the public environment.

In [chapter 5](#), we have evaluated our approach against other scheduling strategies known as *FCFS* and *AsQ*. We have comparatively obtained better results for both scientific and random workflows in performance and cost. Our approach illustrates that effectively leveraging private resources for workflow execution has an explicit influence on total execution time as well as reducing dispatching tasks to the public cloud. Therefore, the more the private instances are utilized, the fewer jobs are dispatched to the public cloud in the hybrid cloud.

6.3 Future Work

Hybrid cloud as a promising model for workload deployment still needs to be studied more regarding security, energy efficiency, and load balancing. This thesis covered a portion of security aspect which could be extended to more complex scenarios like applying encryption and studying the effect of encryption on offloading decisions. As a conventional strategy splitting up the data and tasks into sensitive and non-sensitive groups would resolve the issue to some extent. However, how would the hybrid cloud manage workflow scheduling while all tasks and datasets could be offloaded to the public cloud via different encryption methods? The overhead which could be created by the encryption would explicitly influence the time which would be spent on transferring, encrypting, and decryption of the data at the computation hosts [45]. Thus, light-weight strategies should be utilized to extend the ability for offloading decisions to the public cloud.

List of Symbols

HSHC Hybrid Scheduling for Hybrid Clouds

FCFS First Come First Served

BoT Bag-of-Tasks

DAG Directed Acyclic Graph

SaaS Software as a Service

PaaS Platform as a Service

IaaS Infrastructure as a Service

FaaS Function as a Service

QoS Quality of Service

S Storage

C Computation

B Bandwidth

$Cost(S)$ Cost(Storage)

$Cost(C)$ Cost(Computation)

$Cost(B)$ Cost(Bandwidth)

$B(in)$ Bandwidth(Input)

$B(out)$ Bandwidth(Output)

l_{r_i} Links of Resource i

L_r^τ Leased resources at the time of τ

TS Task Set

DS Data Set

IDS Intermediate Data Set

ds Dataset

DT Data Task

D Deadline

$Exec$ Execution

PT Performing Time

MT Movement Time

t Task

R Resource

VM Virtual Machine

DI Dependency Inner

DO Dependency Outer

P Performance

ava Availability

CP Computation Capability

Vu Virtual Machine Usage

ck_f Check Fixed dataset

ck_r Check data retrieval

FT Finishing Time

FTMatrix Finishing Time Matrix

l_{off} Offloading List

F Flexible Tasks

NF Non-Flexible Tasks

DTMatrix Deadline Task Matrix

MIPS Million Instructions Per Second

Homo. Homogeneous

Hetero. Heterogeneous

inst. Instance

References

- [1] A. Weiss. *Computing in the clouds*. netWorker **11**(4), 16 (2007). 1
- [2] J. C. Jacob, D. S. Katz, G. B. Berriman, J. C. Good, A. C. Laity, E. Deelman, C. Kesselman, G. Singh, M.-H. Su, T. A. Prince, and R. Williams. *Montage: a grid portal and software toolkit for science-grade astronomical image mosaicking*. Int. J. Comput. Sci. Eng. **4**(2), 73 (2009). URL <https://doi.org/10.1504/IJCSE.2009.026999>. 2
- [3] S. Abrishami, M. Naghibzadeh, and D. H. J. Epema. *Cost-driven scheduling of grid workflows using partial critical paths*. IEEE Transactions on Parallel and Distributed Systems **23**(8), 1400 (2012). 3
- [4] S. Abrishami, M. Naghibzadeh, and D. H. J. Epema. *Deadline-constrained workflow scheduling algorithms for infrastructure as a service clouds*. Future Generation Computer Systems **29**(1), 158 (2013). 3, 6
- [5] E.-K. Byun, Y.-S. Kee, J.-S. Kim, E. Deelman, and S. Maeng. *Bts: Resource capacity estimate for time-targeted science workflows*. Journal of Parallel and Distributed Computing **71**(6), 848 (2011). 3, 6
- [6] M. Malawski, K. Figiela, and J. Nabrzyski. *Cost minimization for computational applications on hybrid cloud infrastructures*. Future Generation Computer Systems **29**(7), 1786 (2013). 3, 8, 10
- [7] A. Abramovici, W. E. Althouse, R. W. P. Drever, Y. Gürsel, S. Kawamura, F. J. Raab, D. Shoemaker, L. Sievers, R. E. Spero, K. S. Thorne, R. E. Vogt, R. Weiss, S. E. Whitcomb, and M. E.

- Zucker. *Ligo: The laser interferometer gravitational-wave observatory*. Science **256**(5055), 325 (1992). <http://science.sciencemag.org/content/256/5055/325.full.pdf>, URL <http://science.sciencemag.org/content/256/5055/325>. 4
- [8] W.-J. Wang, Y.-S. Chang, W.-T. Lo, and Y.-K. Lee. *Adaptive scheduling for parallel tasks with qos satisfaction for hybrid cloud environments*. The Journal of Supercomputing **66**(2), 783 (2013). 4, 7, 10
- [9] D. Yuan, Y. Yang, X. Liu, and J. Chen. *A data placement strategy in scientific cloud workflows*. Future Generation Computer Systems **26**(8), 1200 (2010). 6, 7, 8
- [10] K. Deng, J. Song, K. Ren, D. Yuan, and J. Chen. *Graph-cut based coscheduling strategy towards efficient execution of scientific workflows in collaborative cloud environments*. In *2011 IEEE/ACM 12th International Conference on Grid Computing*, pp. 34–41. 6, 7
- [11] M. Malawski, G. Juve, E. Deelman, and J. Nabrzyski. *Algorithms for cost- and deadline-constrained provisioning for scientific workflow ensembles in iaas clouds*. Future Generation Computer Systems **48**, 1 (2015). 6
- [12] A. Deldari, M. Naghibzadeh, and S. Abrishami. *Cca: a deadline-constrained workflow scheduling algorithm for multicore resources on the cloud*. The journal of Supercomputing **73**(2), 756 (2017). 6
- [13] K. Deng, L. Kong, J. Song, K. Ren, and D. Yuan. *A weighted k-means clustering based co-scheduling strategy towards efficient execution of scientific workflows in collaborative cloud environments*. In *2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing*, pp. 547–554. 6, 7
- [14] X. Liu and A. Datta. *Towards intelligent data placement for scientific workflows in collaborative cloud environment*. In *2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum*, pp. 1052–1061. 7
- [15] F. Ma, Y. Yang, and T. Li. *A data placement method based on bayesian network for data-intensive scientific workflows*. In *2012 International Conference on Computer Science and Service System*, pp. 1811–1814. 7

- [16] L. Teylo, U. de Paula, Y. Frota, D. de Oliveira, and L. M. Drummond. *A hybrid evolutionary algorithm for task scheduling and data assignment of data-intensive scientific workflows on clouds*. *Future Generation Computer Systems* **76**, 1 (2017). [7](#)
- [17] E. I. Djebbar and G. Belalem. *Optimization of Tasks Scheduling by an Efficacy Data Placement and Replication in Cloud Computing*, pp. 22–29 (Springer International Publishing, Cham, 2013). [7](#)
- [18] D. Yuan, Y. Yang, X. Liu, and J. Chen. *On-demand minimum cost benchmarking for intermediate dataset storage in scientific cloud workflow systems*. *Journal of Parallel and Distributed Computing* **71**(2), 316 (2011). [7](#)
- [19] M. R. H. Farahabady, Y. C. Lee, and A. Y. Zomaya. *Pareto-optimal cloud bursting*. *IEEE Transactions on Parallel and Distributed Systems* **25**(10), 2670 (2014). [7](#)
- [20] H. Y. Chu and Y. Simmhan. *Cost-efficient and resilient job life-cycle management on hybrid clouds*. In *2014 IEEE 28th International Parallel and Distributed Processing Symposium*, pp. 327–336. [7](#), [10](#)
- [21] S. Wu, B. Li, X. Wang, and H. Jin. *Hybridscaler: Handling bursting workload for multi-tier web applications in cloud*. In *Parallel and Distributed Computing (ISPDC), 2016 15th International Symposium on*, pp. 141–148 (IEEE). [7](#)
- [22] R. L. Cunha, E. R. Rodrigues, L. P. Tizzei, and M. A. Netto. *Job placement advisor based on turnaround predictions for hpc hybrid clouds*. *Future Generation Computer Systems* **67**, 35 (2017). [7](#), [10](#)
- [23] O.-C. Marcu, C. Negru, and F. Pop. *Dynamic scheduling in real time with budget constraints in hybrid clouds*. In *International Conference on Grid Economics and Business Models*, pp. 18–31 (Springer). [10](#)
- [24] A. Zinnen and T. Engel. *Deadline constrained scheduling in hybrid clouds with gaussian processes*. In *High Performance Computing and Simulation (HPCS), 2011 International Conference on*, pp. 294–300 (IEEE). [7](#), [10](#)

- [25] B. Wang, Y. Song, Y. Sun, and J. Liu. *Managing deadline-constrained bag-of-tasks jobs on hybrid clouds*. In *Proceedings of the 24th High Performance Computing Symposium*, p. 22 (Society for Computer Simulation International). 7, 10
- [26] H. Yuan, J. Bi, W. Tan, and B. H. Li. *Temporal task scheduling with constrained service delay for profit maximization in hybrid clouds*. *IEEE Transactions on Automation Science and Engineering* **14**(1), 337 (2017). 7, 10
- [27] M. R. Hoseinyfarahabady, H. R. D. Samani, L. M. Leslie, Y. C. Lee, and A. Y. Zomaya. *Handling uncertainty: Pareto-efficient bot scheduling on hybrid clouds*. In *2013 42nd International Conference on Parallel Processing*, pp. 419–428. 7, 10
- [28] A. Rezaeian, H. Abrishami, S. Abrishami, and M. Naghibzadeh. *A budget constrained scheduling algorithm for hybrid cloud computing systems under data privacy*. In *Cloud Engineering (IC2E), 2016 IEEE International Conference on*, pp. 230–231 (IEEE). 7, 10
- [29] A. Calatrava, G. Molto, and V. Hern'andez. *Combining grid and cloud resources for hybrid scientific computing executions*. In *2011 IEEE Third International Conference on Cloud Computing Technology and Science*, pp. 494–501. 7, 10
- [30] Y. C. Lee and B. Lian. *Cloud bursting scheduler for cost efficiency*. In *2017 IEEE 10th International Conference on Cloud Computing (CLOUD)*, pp. 774–777. 8, 10
- [31] R. Van den Bossche, K. Vanmechelen, and J. Broeckhove. *Online cost-efficient scheduling of deadline-constrained workloads on hybrid clouds*. *Future Generation Computer Systems* **29**(4), 973 (2013). 8, 10
- [32] L. F. Bittencourt, C. R. Senna, and E. R. Madeira. *Scheduling service workflows for cost optimization in hybrid clouds*. In *Network and Service Management (CNSM), 2010 International Conference on*, pp. 394–397 (IEEE). 8, 10
- [33] M. Malawski, K. Figiela, M. Bubak, E. Deelman, and J. Nabrzyski. *Cost Optimization of Execution of Multi-level Deadline-Constrained Scientific Workflows on Clouds*, pp. 251–260 (Springer Berlin Heidelberg, Berlin, Heidelberg, 2014). 8, 10

- [34] B. Lin, W. Guo, and X. Lin. *Online optimization scheduling for scientific workflows with deadline constraint on hybrid clouds*. *Concurrency and Computation: Practice and Experience* **28**(11), 3079 (2016). [8](#)
- [35] B. Lin, W. Guo, N. Xiong, G. Chen, A. V. Vasilakos, and H. Zhang. *A pretreatment workflow scheduling approach for big data applications in multicloud environments*. *IEEE Transactions on Network and Service Management* **13**(3), 581 (2016). [9](#), [10](#)
- [36] M. Rahman, X. Li, and H. Palit. *Hybrid heuristic for scheduling data analytics workflow applications in hybrid cloud environment*. In *Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 2011 IEEE International Symposium on*, pp. 966–974 (IEEE). [9](#), [10](#)
- [37] L. F. Bittencourt and E. R. M. Madeira. *Hcoc: a cost optimization algorithm for workflow scheduling in hybrid clouds*. *Journal of Internet Services and Applications* **2**(3), 207 (2011). [9](#), [10](#), [31](#)
- [38] L. Chunlin and L. LaYuan. *Hybrid cloud scheduling method for cloud bursting*. *Fundamenta Informaticae* **138**(4), 435 (2015). [10](#)
- [39] F. B. Charrada and S. Tata. *An efficient algorithm for the bursting of service-based applications in hybrid clouds*. *IEEE Transactions on Services Computing* **9**(3), 357 (2016). [10](#)
- [40] H. Wei and F. Meng. *A novel scheduling mechanism for hybrid cloud systems*. In *2016 IEEE 9th International Conference on Cloud Computing (CLOUD)*, pp. 734–741. [10](#)
- [41] S. Abdi, L. PourKarimi, M. Ahmadi, and F. Zargari. *Cost minimization for deadline-constrained bag-of-tasks applications in federated hybrid clouds*. *Future Generation Computer Systems* **71**, 113 (2017). [10](#)
- [42] R. V. d. Bossche, K. Vanmechelen, and J. Broeckhove. *Cost-optimal scheduling in hybrid iaas clouds for deadline constrained workloads*. In *2010 IEEE 3rd International Conference on Cloud Computing*, pp. 228–235. [10](#)
- [43] X.-S. Yang. *Nature-inspired optimization algorithms* (Elsevier, 2014). [18](#)

-
- [44] 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*. *Softw. Pract. Exper.* **41**(1), 23 (2011). URL <http://dx.doi.org/10.1002/spe.995>. 32
- [45] Z. Zhou, H. Zhang, X. Du, P. Li, and X. Yu. *Prometheus: Privacy-aware data retrieval on hybrid cloud*. In *2013 Proceedings IEEE INFOCOM*, pp. 2643–2651. 41