DATA-AWARE PLACEMENT AND SCHEDULING FOR SCIENTIFIC WORKFLOW IN CLOUD COMPUTING

A

Thesis

Submitted to

For the award of

DOCTOR OF PHILOSOPHY (Ph.D.)

IN

COMPUTER SCIENCE AND ENGINEERING

By

Avinash Kaur

41400090

***S*upervised By :**

Dr. Pooja Gupta

***C*o-Supervised By :**

Dr. Manpreet Singh

LOVELY FACULTY OF TECHNOLOGY AND SCIENCES

LOVELY PROFESSIONAL UNIVERSITY

PUNJAB

2019

DECLARATION

This thesis is an account of research undertaken between August 2014 and April 2019 at The Department of Computer Science and Engineering, Lovely Professional University, Phagwara, India.

Except where acknowledged in the customary manner, the material presented in this thesis is, to the best of my knowledge, original and has not been submitted in whole or part for a degree in any university.

Avinash Kaur
Registration no. 41400090
Department of Computer Science and Engineering
Lovely Professional University, Phagwara, India

CERTIFICATE

This is to certify that the declaration statement made by the student is correct to the best of my knowledge and belief. She has submitted the Ph.D. thesis **Data-aware Placement and Scheduling for Scientific Workflow in Cloud Computing** under my guidance and supervision. The present work is the result of her original investigation, effort and study. No part of the work has ever been submitted for any other degree at any University. The Ph.D. thesis is fit for the submission and fulfillment of the conditions for the award of Ph.D. degree in Computer Science Engineering from Lovely Professional University, Phagwara.

Dr. Pooja Gupta
Associate Professor
Department of CSE
Lovely Professional University,
Phagwara, India

Dr. Manpreet Singh
Assistant Professor
Department of IT
Guru Nanak Dev Engineering
College, Ludhiana, India

ABSTRACT

Cloud computing is the latest distributed computing paradigm with a lot of opportunities to run workflow applications without having in-house infrastructure. In particular, cloud offers scalable, easily accessible and cost-efficient services for scientific workflow in cloud data centers. Cloud provides an unlimited virtual pool of resources that can be acquired, configured, used and released as per the real-time requirement of the applications.

Scientific Workflow is a composition of coarse-grained and fine-grained tasks with varying execution requirements. In the big data era, it is compelling to extract and process the knowledge from consistent data producing applications such as gravitational wave detectors, particle accelerators, and telescopes. Large-scale data transfer is involved in these applications, so efficient data placement techniques are required to reduce the data-movement between the datacenters. The scheduling algorithms is a key to automate their execution in cloud environment. It is a challenging task to develop a scheduling algorithm, which can complete the execution of scientific workflows within budget and deadline constraints.

In this thesis, the state-of-the-art is enhanced with task clustering technique, data placement scheme and scheduling algorithm for scientific workflows in cloud environment. A Hybrid Balanced (HYB) task clustering technique has been developed to group the task either horizontally or vertically based on the impact factor of the tasks in scientific workflow. Further, a meta-heuristic approach has been applied to develop a Crow Search Algorithm (CSA) based data placement scheme in order to find an appropriate

data center for intermediate datasets. Furthermore, Data Placement Oriented Heterogeneous Earliest Finish Time (DPO-HEFT) scheduling algorithm has been developed using the comprehension of task clustering and data placement with the balanced trade-off of budget and deadline specifications.

The proposed HYB task clustering algorithm is efficient to combine the task in both horizontal and vertical directions. This approach minimize the execution time through a reduction in scheduling overhead. Impact factor technique applied in the proposed clustering also resolves the distance and runtime imbalance issue in existing clustering techniques for scientific workflow. Crow Search Algorithm (CSA) based placement scheme precisely place intermediate datasets at appropriate data centers. The developed model reduce the data movement in the data centers, which directly affect the data transfer cost and execution time of the scientific process. The DPO-HEFT scheduling algorithm successfully addressed the fundamental issues of scientific workflow scheduling for budget and deadline constraints.

The experiment results show that the data placement and scheduling algorithms make a significant improvement in data movement and scheduling overhead. The designed approach is dynamic in nature which takes clustering and placement decisions as per budget and deadline parameters. The proposed model provides a low cost and fast execution as compared to existing techniques for scientific workflow in a cloud environment.

ACKNOWLEDGEMENTS

I take this opportunity to express my sincere thanks to everyone who has helped me in various capacities to carried out this research and prepare the report.

I am delighted to thank our respected supervisors Dr. Pooja Gupta and Dr.Manpreet Singh who have offered tremendous support in the completion of this research. Their unparalleled knowledge, judgment, and moral fiber were together with their expertise.

I acknowledge the Department of Computer Science and Engineering, Lovely Professional University to provide me the appropriate resources and financial support to pursue the doctoral degree. I am grateful to the administration staff at the Centre for Research Degree Programmes for the numerous applications.

I would also thank my parents, husband, daughter, friends, and contemporaries for their co-operation and compliance. I cannot cherish a greater fortune other than having them in my life. Their care and love are indispensable for my achievements.

Avinash Kaur

CONTENTS

Declaration ii

Certificate iii

Abstract iv

Acknowledegments vi

List of Figures xiii

List of Tables xv

List of Abbreviations xvi

1 Introduction 1

1.1 Introduction to Cloud Computing 1

1.2 Overview of Workflow 2

1.2.1 Classification of Workflow 2

1.2.2 Scientific Workflow Management System 3

1.2.3 Challenges of Workflow Management in Cloud 5

1.3 Research Issues and Objectives 6

1.4 Research Methodology 7

1.5 Thesis Contributions 9

1.6 Thesis Organization 10

2 Literature Review **13**

2.1 Introduction 13

2.2 Review on Workflow Management System 14

2.3 Review on Task Clustering Techniques 16

2.4 Review on Data Placement Techniques 19

2.4.1 Classification of Data Placement Techniques 19

2.5 Review on Scheduling Algorithms 35

2.6 Summary 37

3 Hybrid Balanced Task Clustering for Scientific Workflow in Cloud Computing **39**

3.1 Introduction 39

3.2 Related Work 41

3.2.1 Load Imbalance 41

3.2.2 Granularity 42

3.2.3 Structural Similarity 42

3.2.4 Data Dependency 43

3.3 System Architecture 44

3.3.1 Workflow 44

3.3.2 Workflow Model 45

3.3.3 Workflow Execution Environment 46

3.4 Proposed Hybrid Balanced Task Clustering Algorithm 47

3.4.1 Problem Formulation 47

3.4.2 Research Methodology 48

3.5 Experimental Evaluation 55

3.5.1 Scientific Workflow Applications 55

3.5.2 Balanced Task Clustering Algorithms 57

3.5.3 Description of Baseline Balanced Clustering Algorithms 57

3.5.4 Experimental Evaluation 61

3.5.5 Results and Discussion 62

3.6 Summary 67

4 A Data Placement Strategy Based on Crow Search Algorithm in Cloud Computing 68

4.1 Introduction . . . 68
4.2 Related Work . . . 71
4.3 Data Placement Process . . . 73
4.3.1 Data Placement Stages . . . 73
4.3.2 Need for Data Placement . . . 74
4.4 Research Methodology . . . 76
4.4.1 Proposed Crow Search Algorithm (CSA) Based Data Placement 76
4.4.2 Pseudo Code of Proposed CSA-based Runtime Data Placement 78
4.5 Comparison of CSA and PSO . . . 82
4.6 Experimental Evaluation . . . 82
4.6.1 Experimental Setup . . . 83
4.6.2 Results and Discussion . . . 84
4.7 Summary . . . 88

5 Data Placement Oriented Scheduling Algorithm for Scientific Workflows in Cloud: A Budget-Aware Approach 90

5.1 Introduction . . . 90
5.2 Related Work . . . 92
5.3 System Architecture . . . 93
5.3.1 Cloud Data Center Model . . . 93
5.3.2 Workflow Model . . . 96
5.3.3 System Model . . . 96
5.4 Research Methodology . . . 98
5.4.1 Phase 1: Budget Distribution . . . 99
5.4.2 Phase 2: Task Clustering in Workflow . . . 100
5.4.3 Phase 3: Deadline Distribution . . . 101
5.4.4 Phase 4: Task Selection . . . 102
5.4.5 Phase 5: Instance Selection . . . 102
5.5 Experimental Evaluation . . . 104
5.5.1 Experimental Methodology . . . 104
5.5.2 Experimental Setup . . . 104

5.5.3 Results and Discussion 107
5.6 Summary 115

6 Conclusion and Future Directions **117**
6.1 Conclusion and Discussion 117
6.2 Future Scope 119
6.2.1 Energy-Efficient Placement and Scheduling 119
6.2.2 Replication Management 119
6.2.3 Fault-tolerance and Provenance Management 119
6.2.4 Pricing Models 120

References **140**

Publications **141**

LIST OF FIGURES

1.1 Scientific Workflow Management System functional architecture[1] . . 4
1.2 Research methodology flowchart 8
1.3 Chapter wise thesis organization 11

2.1 Data placement schemes in Cloud computing 20

3.1 WfMC reference model [2] 45
3.2 Workflow model 45
3.3 Workflow management system 46
3.4 Flowchart of Proposed Hybrid Balanced Clustering Algorithm 49
3.5 A sample workflow 50
3.6 Merging of clusters in a job 52
3.7 Clustering to avoid runtime imbalance and dependency imbalance . . . 52
3.8 Merging clusters into a job 52
3.9 Clustering to avoid runtime imbalance and dependency imbalance . . . 53
3.10 Merging clusters into a job 53
3.11 Clustering to avoid runtime imbalance and dependency imbalance . . . 54
3.12 Merging clusters into a job 54
3.13 A simplified version of the Montage workflow [3]. 55
3.14 A synthetic version of the CyberShake workflow [4]. 56
3.15 A synthetic version of Epigenomics workflow with multiple branches [5]. 56
3.16 A simplified version of the SIPHT workflow [6]. 57

3.17 A simplified version of the LIGO Inspiral workflow [7]. 58
3.18 Working of Horizontal Clustering 59
3.19 Working of Vertical clustering 59
3.20 Working of the Horizontal Runtime Balancing (HRB) method 60
3.21 Working of Horizontal Impact Factor Balancing (HIFB) method 60
3.22 Working of Horizontal Distance Balancing (HDB) method 61
3.23 Comparison of performance Gain (μ) for various clustering methods . . 65
3.24 Execution time while varying virtual machines in Horizontal Clustering technique 66
3.25 Execution time while varying virtual machines in Hybrid Balanced Clustering technique 67

4.1 Cloud architecture for Workflow management system 69
4.2 Data Placement Process [8]. 74
4.3 Flowchart of Crow Search Algorithm(CSA) based data placement. . . . 77
4.4 Data Placement Map of initial datasets. 78
4.5 Data movement without storage limit without fixed location datasets (a) varying number of datasets (b) varying number of datacenters 85
4.6 Data movement without storage limit with 20% fixed location datasets (a) varying number of datasets (b) varying number of datacenters 86
4.7 Proportion of three types of data movements (a) Non-Fixed datasets (b) 20% Fixed datasets 87

5.1 Workflow model 95
5.2 Model for scheduling scientific workflows in cloud 96
5.3 A simplified display of the Montage workflow [3]. 105
5.4 A simplified display of Cybershake Workflow [3]. 106
5.5 A simplified display of Epigenomics workflow [3]. 106
5.6 A simplified display of the SIPHT workflow [6] 106
5.7 A simplified display of LIGO workflow [3] 107
5.8 Makespan Time of Montage workflow with 25, 50, 100 and 1000 tasks . 109
5.9 Makespan Time of Cybershake workflow with 30, 50, 100 and 1000 tasks 110
5.10 Makespan Time of Epigenomics workflow with 24, 46, 100 and 997 tasks 112

5.11 Makespan Time of LIGO workflow with 30, 50, 100 and 1000 tasks . . 113
5.12 Makespan Time of SIPHT workflow with 30, 60, 100 and 1000 tasks . . 115

LIST OF TABLES

2.1 Objectives of correlation based data placement schemes 21
2.2 Properties of coorelation based data placement schemes 21
2.3 Objectives of Genetic algorithm energy based data placement schemes . 23
2.4 Properties of Genetic algorithm energy based data placement schemes . 23
2.5 Objectives of energy efficient based data placement schemes 25
2.6 Properties of energy efficient based data placement schemes 25
2.7 Objectives of PSO based data placement schemes 28
2.8 Properties of PSO based data placement schemes 28
2.9 Objectives of ACO based data placement schemes 29
2.10 Properties of ACO based data placement schemes 29
2.11 Objectives of optimization based data placement schemes 31
2.12 Properties of optimization based data placement schemes 31
2.13 Objectives of fault tolerant based data placement schemes 32
2.14 Properties of fault tolerant based data placement schemes 32
2.15 Objectives of replication based data placement schemes 34
2.16 Properties of replication based data placement schemes 34

3.1 Symbols used in this work 47
3.2 Configuration setup for the experiment 62
3.3 Scientific workflow for experiment and number of tasks 63
3.4 Makespan time of workflow with or without clustering algorithms . . . 63
3.5 Performance gain for various workflows 63

4.1 Parameters for data placement 84

5.1 Amazon EC2: Optimized instance series of memory 94
5.2 Amazon EC2: Optimized instance series of computing 94
5.3 Amazon EC2: Optimized instance series of storage vs CPU 94
5.4 Configuration setup for the experiment 105
5.5 Execution time for Montage Workflow 108
5.6 Execution Time for Cybershake Workflow 110
5.7 Execution Time for Epigenomics Workflow 111
5.8 Execution Time for LIGO Workflow 113
5.9 Execution Time for SIPHT Workflow 114

LIST OF ABBREVIATIONS

ACO	Ant Colony Optimization
AP	Awareness Probability
AWS	Amazon Web Services
BaRRS	Balanced and Reuse-Replication Scheduling
BEA	Bond Energy Algorithm
CPOP	Critical Path of Processor
CSA	Crow Search Algorithm
DAG	Directed Acyclic Graph
DC	Dynamic Clustering
dg	Dependency Gain
DPO-HEFT	Data Placement Oriented Hetrogenous Earliest Finish Time
DVFS	Dynamic Voltage and Frequency Scaling
GA	Genetic Algorithm
HC	Horizontal Clustering
HDB	Horizontal Distance Balancing
HEFT	Hetrogenous Earliest Finish Time
HIFB	Horizontal Impact Factor Balanacing
HRB	Horizontal Runtime Balancing
HRV	Horizontal Runtime Variance
IFV	Impact Factor Variance

LIGO	Laser Inferometer Gravitational Wave Observatory
MTC	Multiple Task Computing
PPR	Price/Performance Ratio
PSO	Particle Swarm Optimization
QoS	Quality of Service
SLA	Servie Level Agreement
SOA	Service Oriented Architecture
SR	Selective Reclustering
SWFMS	Scientific Workflow Management System
VC	Vertical Clustering
VM	Virtual Machine
WfMC	Workflow Management Coalition
WPA	Workflow and Platform Aware Task Clustering

CHAPTER 1

INTRODUCTION

1.1 Introduction to Cloud Computing

In 1960 John McCarthy imagined that computation in a certain time would be given as a utility. The acknowledgment of this is cloud computing. Technologies such as Service Oriented Architecture (SOA), web-services and hardware virtualization lay the path of cloud landscape [9].

Cloud computing has emerged from various computing paradigms of distributed computing over the growth of both hardware and Internet technology. Seamlessly using the power of virtualization and distribution, cloud today provides the use of applications and services over the Internet as if they are being used on the local machine. A cloud computing stage is built with the broadly disseminated set of equipment stage which is arranged and running assorted software services [10].

Most of the users are unaware of the services of cloud in their applications such emails, social networking sites and other services offered or hosted on a cloud [11]. Users do not need to have any information about the background services. They can make communication with many servers at the same time and there is communication among servers also. Cloud computing aims to provide services to the users that host documents on the Internet and outsource the IT infrastructure [12].

At an abstract level there are three types of cloud services:

1. **Software as a Service**

   It delivers the software applications services as a utility to the clients. They can get these services from the basic interface, for example, a web program over the Internet [13].

2. **Platform as a Service**

   It presents a high level cohesive framework to assemble, test and deploy the client made applications.

3. **Infrastructure as a Service**

   It gives certain critical computing resources to clients such as processing, storage, network, etc. IaaS clients can convey self-assertive application, software, working frameworks on the infrastructure, which can be scaled on the basis of needs of application resource [13].

1.2 Overview of Workflow

Workflow Management Coalition (WfMC) defined workflow as : "The automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules" [14].

Here, WfMC described the work processes are the progression of organized exercises and calculations of a business procedure, which is a unique depiction of the undertakings to be executed for that business process. They are utilized to join a few diverse computational procedures into a solitary lucid procedure. The business applications can now be seen as complex work processes, which comprise of different changes performed on the information required in accomplishing the goal. Workflows offer points of interest in isolating capacities from applications. In this manner offering data framework to be segment based, which arrange and deployed workflow tasks in cloud data center.

1.2.1 Classification of Workflow

The workflows vary significantly in their computation and data requirement. A workflow may require a large set of data to execute a particular task. It may also require

large computation time in order to finish the execution of a particular task. This can adhere to the combination of data and compute intensive jobs, which produce intermediate data after the processing of the task. The size of produced dataset during the computation can vary [15]. The workflows are classified as scientific workflows and business workflows [16].

1. **Business Workflow:** The business process is automated, where information, tasks or documents are transferred between the users as per the protocols. The objective in business is achieved by a set of linked procedures or activities usually within the structure of an organization. The efficiency and reliability of business processes are increased by a business workflow.

2. **Scientific Workflow:** They are used for running and modeling scientific experiments. In order to reduce the makespan, it assembles data processing and execution of scientific activities. It represents the complex set of data processing activities with data dependencies. There are various ways of representing the scientific workflow and, Directed Acyclic Graph (DAG) is a generic way to represent the scientific workflow.

1.2.2 Scientific Workflow Management System

Scientific Workflow Management System (SWFMS) fulfill the following requirements: supporting parallel computing, software reuse, distributed resource management, user interaction, collaborative sharing, fault tolerance, and other required features. The Scientific Workflow Management System [1] is classified into four layer architecture shown in Figure 1.1.

1. **Application Layer:** In this layer, workflows are constructed, further textual and graphical user interfaces submit requirements to the system. The reusability and flexibility of cloud service components and local resource component is provided by modular programming. The translation of workflows accomplished with a mathematical model in the next layer. Furthermore, this layer presents and visualize the accomplished results.

2. **Service Layer:** The scientific workflows are executed in this layer. It deals with fault tolerance and workflow monitoring for a workflow management system to

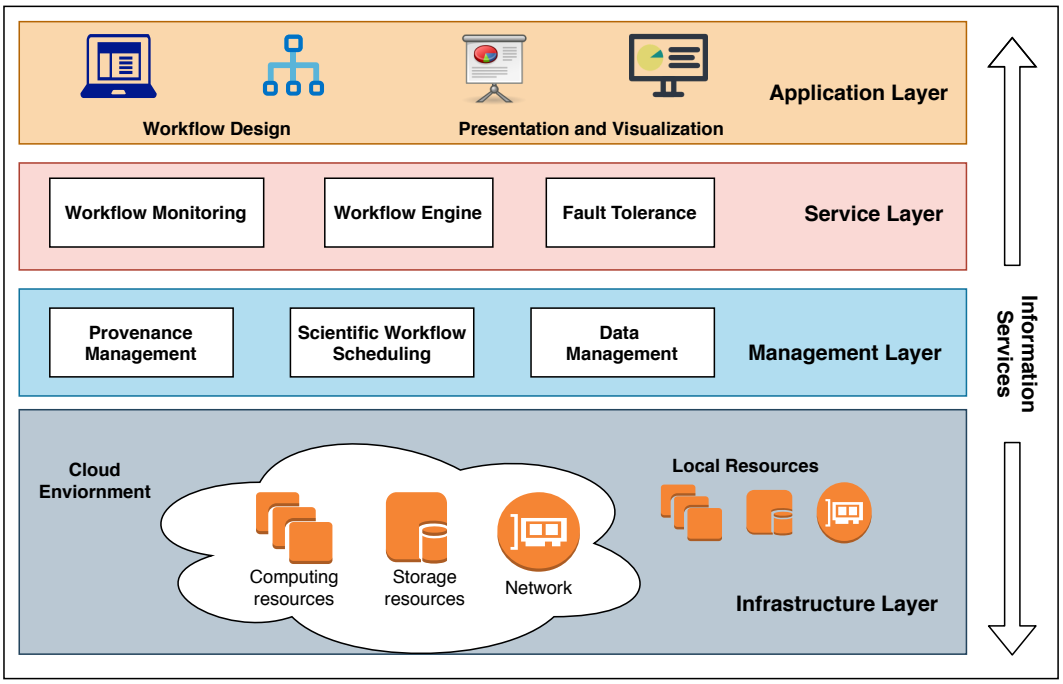


Figure 1.1: Scientific Workflow Management System functional architecture[1]

work well. This layer receives eligibility requirements from the application layer and scheduling requirements are obtained from the management layer. The stability of management system is improved by separating workflow execution from task scheduling and task acquiring.

3. **Management Layer:** It is the bridge between the execution of workflows and physical resources. The major part of this layer is *Scientific Workflow Scheduling* that determines which scheduling algorithm to use. This layer has a significant role in optimizing task procedures and parallelizing the processing of data. Before planning the scheduling, this layer obtains information about the resources and applications. The movement of data is also involved during the scheduling process which is then needed to be handled by efficient data management module. The derivation history of data productions is recorded for tracking the experiment processes and validating the results.

4. **Infrastructure Layer:** It is responsible for expanding the foregoing service platform designed using cloud-based local resources. Using local resources, more service selections and cost savings can be provided. Storage, computing, network and other resources are included in each computing environment.

1.2.3 Challenges of Workflow Management in Cloud

To organize the scientific workflows in the cloud data centers, the researcher considers three major aspects 1) Workflow Clustering, 2) Data Placement and 3) Scheduling. The clustering is a process to merge the tasks of a workflow based on their dependencies. Furthermore, the data placement process is responsible for the placement of data sets required by the tasks on an instance. The scheduling finally executes the tasks of workflow in a cloud environment by considering the budget and deadline constraint.

Challenges in Tasks Clustering

Dependency calculation at run-time is difficult because of complex scientific workflow task (e.g. Many tasks require one dataset and one task may require multiple datasets) which directly affects the efficiency of clustering algorithms. Load balancing of scientific workflow task on the different data centers is still a challenge. Improper clustering of jobs increases the waiting time of Virtual Machines (VMs). All the techniques are considering the homogeneous cloud environment where the heterogeneous environment is provided by cloud providers nowadays.

Challenges in Data Placement

Improper data placement policy increases data movement in the data center which leads to increased cost and delayed services. Placement of runtime datasets is still a challenge in a complex scientific workflow. Storage and computation capacity is a bottleneck in some situations while transferring the intermediate data sets. Most of the techniques are focused on decreasing the data movement but not considered the data size. If the data size is considered it may improve the data placement policy to a significant level.

Challenges in Tasks Scheduling

Pricing model allocates resources to scientific workflow jobs for a fixed number of hours as per the predetermined policy. A small variance in prediction leads to cost desperation for the user and partial use of cloud instances. Performance variance in cloud resources due to failures, sharing of resources, VM consolidation (e.g VM migration) affects the execution time. So it is a research challenge to calculate the execution time of the job.

Scheduling policy required modification with different Quality of Service (QoS) parameters requirement. Optimized use of on-demand, reserve and spot instances resources in scheduling algorithms can give major benefits to users and service providers.

1.3 Research Issues and Objectives

The main objectives of this research is to address the fundamental issues related to clustering algorithms, data placement schemes and scheduling algorithms for scientific workflows in cloud infrastructure. In the present study, researcher targets the following major components of SWfMS.

- **Workflow Mapper:** It is responsible for constructing a job from the small tasks called task clustering. This reduces the system overhead.
- **Data Management:** It is used to manage the large volume of data and store the data at an appropriate data center.
- **Scheduler:** It is responsible for mapping the tasks to the resources with certain policies.

To address the above-mentioned challenges, first, there is a need to develop a task clustering technique to merge the tasks of workflow. This facilitates the cloud in reducing systems overheads and enabling faster execution of tasks. It avoids wastage of resources. The following issues need to be addressed :

- How to reduce the under-utilization of resources?
- How to minimize system overheads?
- How to consider the data dependencies in a workflow?
- How to check the accuracy of clustering technique?

After the task clustering, the researcher developed a data placement technique to place the datasets at an appropriate location in the data center. This approach reduced the cost of execution through minimizing the data movements. The following issues need to be addressed:

- How to effectively distribute data onto a storage device?
- How runtime generated datasets are to be placed?
- How the budget is going to be considered for storing of data?
- What will be the impact of developed technique on the performance metrics?

After the data placement, a scheduling technique need to be designed for the execution of workflows jobs in the cloud data centers. The following issues need to be addressed:

- How execution is to be scheduled with budget constraint?
- How execution is to be scheduled before the completion of the deadline?
- How to effectively develop the execution schedule while considering the available resources?

Therefore, the researcher summarizes the following objectives:

1. To design a dynamic clustering algorithm for data-intensive scientific workflow in cloud environment.
2. To develop a data placement strategy for optimization of data movement in cloud.
3. To develop a budget-aware scheduling algorithm for data-intensive workflow applications in cloud computing.
4. To analyze the proposed algorithms with existing algorithms on parameters such as makespan, data movement, budget, deadline, etc.

1.4 Research Methodology

The expected outcome of the present study is to design and develop the budget and deadline aware scheduling algorithm for data-intensive scientific workflow applications. The proposed approach considers the budget and deadline constraints of the clients. A flow chart in Figure 1.2 is a list of activities to achieve the mentioned objectives

followed by the research methodology for each objective. To achieve the mentioned objectives following research methodology has been applied.

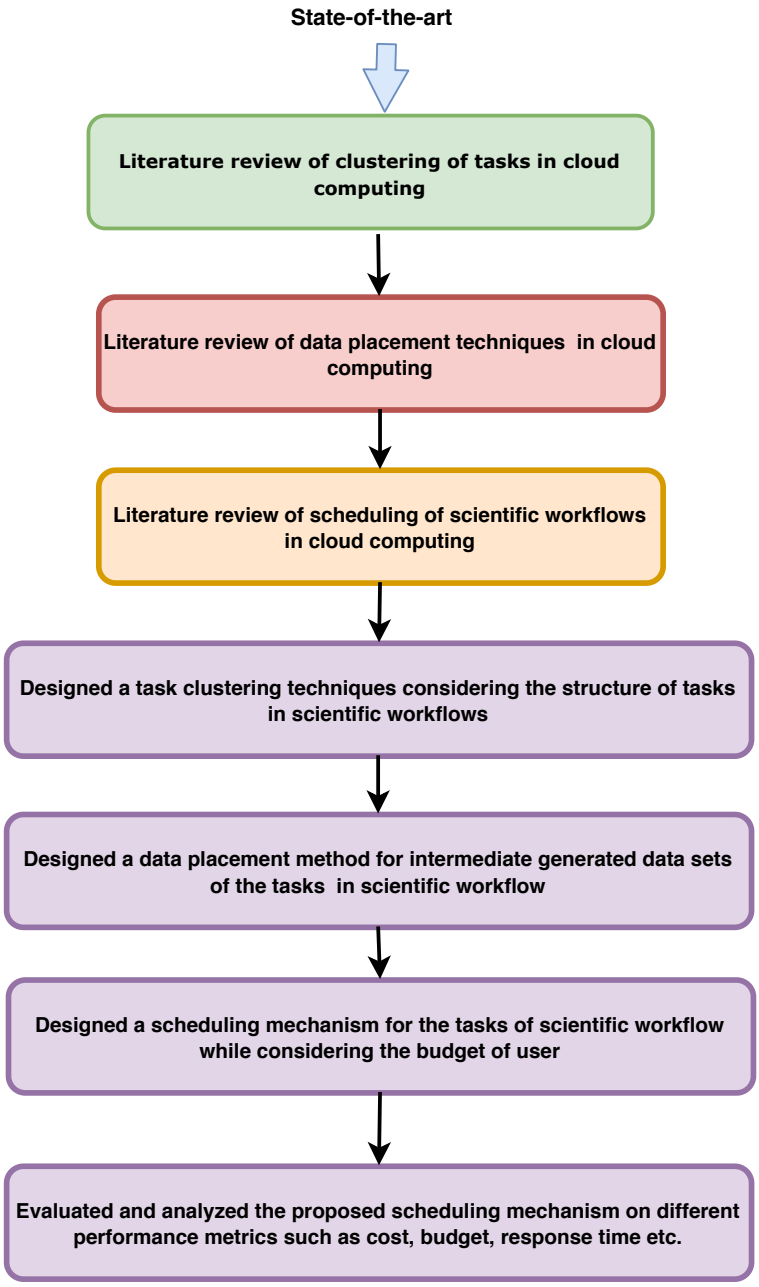


Figure 1.2: Research methodology flowchart

1. To achieve the first objective hybrid balanced task clustering algorithm has been designed on the basis of ordering the tasks in a workflow. The optimal hybridization of horizontal and vertical clustering is used to reduce the makespan of the scientific workflow in the workflow management system for data-intensive applications. The experimental evaluation is performed on workflowsim.

2. To achieve the second objective, a data placement strategy has been designed based on a meta-heuristic approach Crow Search Algorithm (CSA). This approach optimizes the overall data movement among the data centers and chooses data placement location at runtime stage. Experimental evaluation is performed on SwinDeW-C.

3. To achieve the third objective Data Placement Oriented Heterogeneous Earliest Finish Time (DPO-HEFT) scheduling algorithm has been designed to sustain the parallelism in workflow scheduling while considering deadline and budget constraints. The optimization technique has been used for data-intensive workflows. Experimental evaluation is performed on workflowsim with real workflow traces.

4. To achieve the last objective training and testing of the system is performed with Workflowsim and SwinDeW-C. Finally, the performance of the proposed algorithms is evaluated with different scientific workflow applications and compared with existing state-of-the-art algorithms.

1.5 Thesis Contributions

The present study addresses the defined research questions. The thesis contribution is mentioned as follows:

- A literature survey of task clustering, data placement and scheduling mechanisms for scientific workflow in cloud computing.

    1. A detailed investigation carried to study various existing task clustering techniques, data placement methods and scheduling algorithms in cloud computing.
    2. The classification of the mentioned techniques is done as per the common characteristics.
    3. Future research directions in the area of task clustering, data placement and scheduling are presented.

- Task clustering technique for clustering tasks of scientific workflows in cloud computing to reduce the makespan of the workflow.

1. Design and development of task clustering technique to group the tasks of scientific workflows in a cloud.
2. Design and development of task clustering technique to reduce the makespan and scheduling overhead of scientific workflows in cloud.

- Data Placement mechanism for placing the intermediate datasets at an appropriate location in the data center in cloud computing.
  1. Design and development of data placement scheme for placement of data sets at an appropriate location in the data center.
  2. Optimization of data placement technique to reduce the data movement of datasets among the data centers of cloud.
- Scheduling method to execute the scientific workflows with the appropriate inputs.
  1. Design and development of a scheduling algorithm for scientific workflows in a cloud.
  2. Design and development of a scheduling algorithm to execute the workflows with the budget and deadline specified by the user.

1.6 Thesis Organization

The structure of the thesis and its dependencies are shown in Figure 1.3. Chapter 2 is related to the literature survey of clustering, scheduling, and taxonomy of data placement techniques. Chapter 3 focuses on task clustering techniques in cloud computing. Chapter 4 describes data placement of the datasets required by the tasks in cloud computing. Chapter 5 presents the scheduling of scientific workflows in cloud computing. Chapter 6 concludes with future directions. In detail, the organization of thesis as follows:

- Chapter 2 presents the methodological survey of task clustering techniques, the taxonomy of data placement techniques and scheduling of scientific workflows in cloud computing. This chapter is partially derived from:

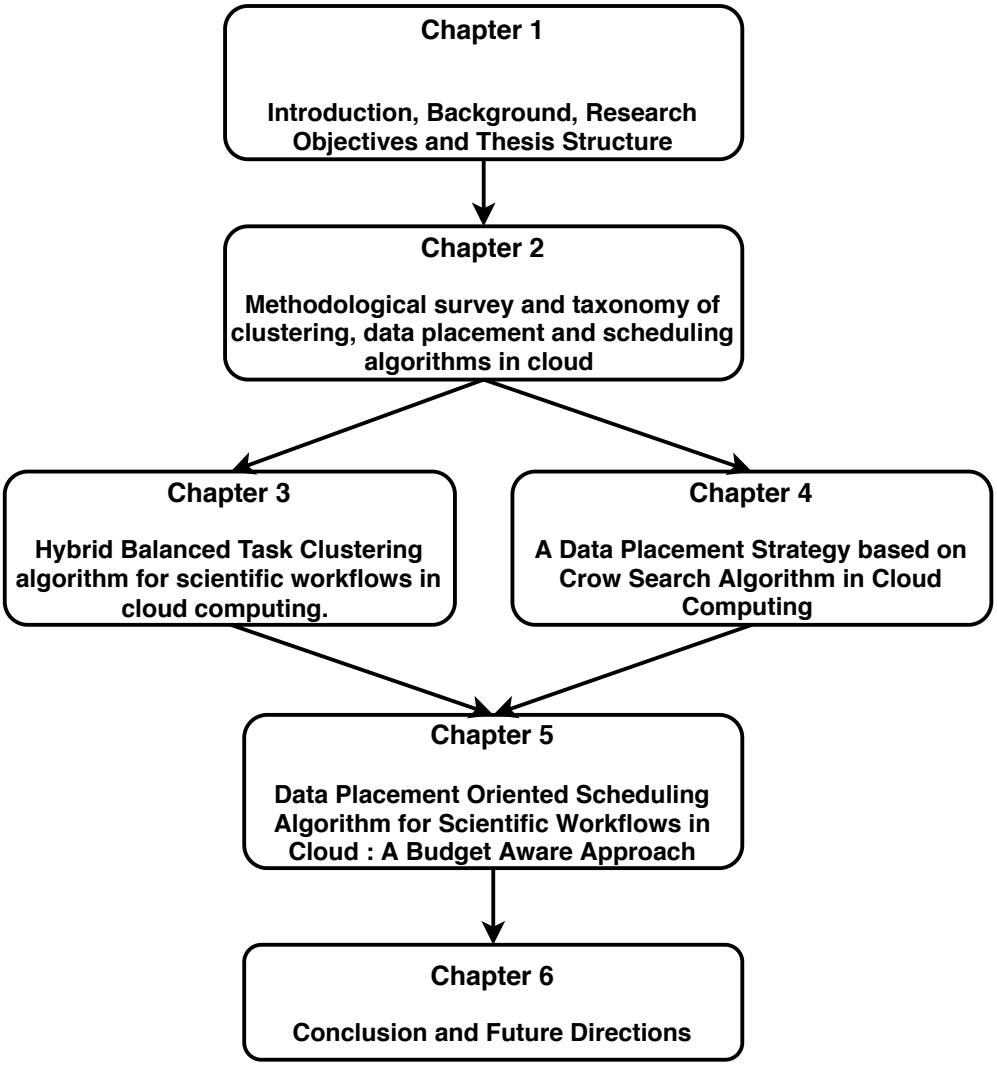


Figure 1.3: Chapter wise thesis organization

  - **Avinash Kaur,** Pooja Gupta and Manpreet Singh, Anand N. "Data Placement in Era Of Cloud Computing: A Survey, Taxonomy And Open Research Issues", *Scalable Computing: Practice and Experience, 20(2), pp. 377-398*, DOI : https:// doi.org/10.12694/scpe.v20i2.1530, 2019. (Scopus, ESCI)

- Chapter 3 describes the proposed task clustering technique for scientific workflows in cloud computing. This chapter is derived from:

  - **Avinash Kaur,** Pooja Gupta and Manpreet Singh, "Hybrid Balanced Task Clustering Algorithm for Scientific Workflows in cloud computing", *Scalable Computing: Practice and Experience, 20(2), pp. 237-258*, DOI: https: //doi.org/10.12694 /scpe.v20i2.1515, 2019. (Scopus, ESCI)

- Chapter 4 presents the proposed data placement technique for scientific workflows in cloud computing. This chapter is derived from :

  – **Avinash Kaur,** Pooja Gupta , "A Data Placement Strategy Based on Crow Search Algorithm in Cloud Computing", *Recent Patents on Computer Science 12(1)*, DOI : https://doi.org/10.2174/ 2213275912666181127123431, 2019. (Scopus)

- Chapter 5 describes the proposed scheduling method for scheduling scientific workflows in cloud computing. This chapter is derived from :

  – **Avinash Kaur,** Pooja Gupta and Manpreet Singh, "DPO-HEFT(Data Placement Oriented HEFT) for Scheduling Scientific Workflows in Cloud Computing", *Recent Patents on Computer Science* (Under Review), 2019. (Scopus)

- Chapter 6 presents the conclusion of the thesis finding and introduce possible future directions. This chapter is partially derived from:

  – **Avinash Kaur,** Pooja Gupta and Manpreet Singh, Anand N. "Data Placement in Era Of Cloud Computing: A Survey, Taxonomy And Open Research Issues", *Scalable Computing: Practice and Experience, 20(2), pp. 377-398*, DOI : https:// doi.org/10.12694/scpe.v20i2.1530, 2019. (Scopus, ESCI)

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

Cloud Computing offers a new standard towards the development of workflow applications. It efficiently executes business as well as scientific workflows. The requirement for the workflow applications is complex execution environment [9]. It is a challenge for the organizations to provide such complex environments in-house for the workflow applications. Therefore, Cloud is considered the best platforms for executing workflow applications due to various features of cloud environment such as elasticity, on-demand pricing policy, distributed data centers, heterogeneous resources, etc [10].

Executing the scientific workflows on cloud requires innovation in task clustering, data placement and scheduling. The scientific workflow in cloud environment seeks the consideration of the following points:

- Scientific workflows tasks need to be clustered for reduction of inter-task communication cost and scheduling overhead.
- Build time and intermediate datasets used in the tasks execution need to be placed appropriately.
- Scheduling of task to the appropriate resources in order to reduce the cost and makespan time.

All the above objectives are necessary to execute the workflow efficiently within the budget, minimum execution and scheduling cost. In this chapter, the literature survey on different aspects has been presented on the Workflow Management Systems (WMS), task clustering techniques, data placement methods and scheduling algorithms.

2.2 Review on Workflow Management System

Deelman et al. [17] designed a Peagus framework for scientific workflows with complex mapping. It plans, executes and monitors the scientific workflows. Pegasus WMS consists of DAGMan execution engine, Pegasus mapper, and condor scheduler for task execution. The user defines the job as a Direct Acyclic Graph (DAG), in which tasks are represented by vertices and dependency between tasks constitutes as edges. Pegasus generates the concrete workflow converted from the abstract workflows by mapping the tasks on the grid resources at runtime. The framework provides a relative timing for scheduling the task over a grid resource. Hull et al. [18] proposed Taverna workflow management system, it is an open source project that supports complex, service-based and data-intensive process automation. It aims to improve the scalability of workflow execution. It also provides clear performance tuning configurations for third-party developers and expandability points for adding new builds. Travena processors collect a large volume of data values through the web service interface from a variety of databases. Travena offers efficient data management and workflow design testing.

Warneke et al. [19] designed a Nephele, a dynamic resource allocation framework in IaaS cloud to schedule and execute tasks. The jobs are expressed as DAG. A job manager transforms the job graph received from the user to the execution graph. Nephle ensured the cost effectiveness execution by allocating and deallocating the resources using just-in-time allocation strategy. Nephele applied feedback data to detect both the computational and input/output bottleneck for refinement of recurring jobs scheduling strategy. Nephele keeps a track of instance allocation times, in case the resource can be reutilized for the same instance type. It is not deallocated and reassigned the tasks and thus reducing the processing cost and overall resource utilization. Bientinesi et al. [20] introduced cloud workflow management system: SwinDeW-C (Swinburne Decentralized Cloud Workflow) of large-scale workflow applications. The models consist of

SwimDeW-C coordinator peers with abilities to provision resources when needed for managing the QoS requirements, data management and security issues. The workflow is submitted to SwimDeW-C peers after the evaluation of the workflow for its non-functional QoS requirements. When the tasks are successfully allocated, then it further deploy the workflow instance. At run time, the coordinated peers executed various workflow management activities to achieve satisfactory performance, but still there is a scope of improvement. The final and the intermediate results are presented to the user through SwimDeW-C web portal.

De Oliveira et al. [21] proposed SciCumulus, a cloud middleware to support Multiple Task Computing (MTC) paradigm to perform multiple parallel workflow tasks on multiple cloud processors. The middleware promotes control workflow components as an abstract view to the scientists of the complex cloud environment. The SciCumulus provides a computational infrastructure for executing and supporting workflow parallelism with parameter sweep and data parallelism. The SciCumulus requires the scientists to enter the details of the workflow and to specify activities to be parallelized, parameters to be explored and various other requirements. It handles the deployment and execution through a layered architecture. The solution presented in SciCumulus is not applicable to a heterogeneous cloud environment. Oliveira et al. [22] proposed an architecture SimiFLow for similarity based comparison and clustering using a bottom-up approach. It considered modeling of experiment lines. The implementation is done in the GExpline tool for managing experiment lines. After the calculation of similarity, SimiFlow generates clusters of the concrete workflow. For obtaining a modeled workflow, scientist does not have any guidance. However, similarity can be easily judged from already modeled workflows. SimiFlow used three cartridges, first cartridge for importing workflow, second for comparing workflow and third for calculating the similarity. In the future, the author intends for the execution of comparison and clustering of the workflow. SimiFLow can be enhanced by developing clustering algorithms.

Wei et al. [23] proposed Manjrasoft ANEKA a framework which is used for the development, management, and deployment of the cloud applications. The service-oriented framework is deployed on the heterogeneous resources and service which coordinate the execution of applications. The middleware also provides advanced management capabilities such as reporting, remote deployment, control of application and

infrastructure, monitoring of the resources in cloud and integration with other cloud technologies to ensure that applications are executed under specified Quality of Service (QoS). Aneka offers an extensible API for the development of applications which are distributed in nature and integration of new capabilities in the cloud. It also supports different types of cloud: public, private or hybrid. Aneka provides the organization to achieve end-to-end performance, scalability and high availability by conforming to the Service Level Agreement (SLA)and providing the desired Quality of Service (QoS).

2.3 Review on Task Clustering Techniques

Chen and Ewa [24] analyzed the transient failures in workflow management systems and categorized into job failure and task failure. Further, author introduced techniques of Selective Reclustering (SR) and Dynamic Clustering (DC). The author also addressed the issues of *run-time imbalance* and *dependency imbalance* in task clustering. These balancing methods reduced the workflow applications runtime performance as per the number of clusters at each level decided by the users. It is unable to exploit the parallelism between tasks. The main focus is on the occurrence of failure and improvement of the workflow makespan in a faulty environment. The robustness of the methods is evaluated using the variance of patterns of failure, overhead, and runtime. The workflow clustering also needs to be dynamically adjusted. Chen et al. [25] examined the cause for Dependency Imbalance and Runtime Imbalance in clustering of tasks. It introduced quantitative metrics for evaluation of the severity of these problems. A series of balancing tasks are proposed to address the imbalance problem. The two new metrics introduced are Horizontal Runtime Variance (HRV) for dealing with Runtime Imbalance and Impact Factor Variance (IFV) for dealing with dependency imbalance. Horizontal Impact Factor Balancing (HIFB) technique and the Horizontal Distance Balancing (HDB) technique are the clustering methods introduced to solve the workflow balancing issues. The methods are implemented in WorkflowSim simulator using two real workflows. The aim of the author is to address the overhead problem and to extend the HDB method for horizontal clustering.

The intermediate strategy for data placement addressed the problem of security in sensitive data [26]. The security overhead model is proposed to measure overheads in security produced by sensitive data and for the dynamic placement of data of scientific

workflows. The data security is ensured in the model based on data confidentiality, the authentication process, and data integrity. Security services are measured quantitatively by the data model. The selection of an appropriate data center performed using Ant Colony Optimization (ACO) based algorithm. Li et al. [27] introduced a Particle Swarm Optimization (PSO) data transmission model for developing a cost-effective data placement technique. It deals with the mapping of datasets onto different data centers. Mostly the workflow tasks are seen as an individual process while ignoring shared datasets among workflows. The author focused on reducing data transmission costs for multiple workflows. The introduced strategy is found to be generic and can be achieved at a much lower cost. In the future, it aims to introduce quantum-based PSO for better convergence.

Energy-aware scientific workflow scheduling algorithm EARES-D was proposed and evaluated on cloudsim simulator to meet Service Level Agreement (SLA) response time by minimizing energy consumption and CO2 emissions [28]. Within the acceptable performance, limitations to reduce more energy consumption it has applied Dynamic Voltage and Frequency Scaling (DVFS) [29] and DNS [30] schemes. Further it aims to run experiments on Saluki (local private cloud) established by Eucalyptus. Wang et al. [31] proposed a solution for separating the role of workflow composer and virtual machine administrator by introducing four heuristic algorithms namely Static Maximal VMI number, Task Scheduling Algorithm for short process time, Adaptive Task schedule algorithm and Greedy Task schedule algorithm for WFaaS approach. It relies on the fact that one VM is available for one package while considering the factors of monetary cost, process time and Price/performance Ratio (PPR) with scale-out any physical Virtual Machine (VM). Proper configuration of the host are required to minimize the cost and PPR. The algorithms proposed processed only DAG workflows that are based on performance and ignored the factors of overhead and cost VMI instantiation. Chen et al. [32] proposed metrics to address the problem of load balancing in clustering. It has compared the performance of methods with two algorithms. It addressed the problems of load imbalance and data dependency. The work is an extension of the top-down and bottom-up approach and both horizontal and vertical task clustering techniques are considered. For the qualitative measure of dependency imbalance, the metrics proposed are Impact Factor Variance(IFV) and distance variance. In the

future, the author aims to normalize the value of proposed metrics and also implement the metrics on asymmetric structures.

The portfolio clustering algorithm is designed to changes among multiple clustering algorithms and make the selection according to the dynamic load. Chen et al. [32] conducted the theoretical analysis of the impact on the runtime performance of scientific workflow executions of transient failures. A general modeling framework presented for task failure that uses a parameter estimation process based on Maximum Likelihood. Three fault-tolerant clustering strategies introduced to improve workflow execution performance in faulty execution environments. For the optimization of the makespan of workflow, a dynamic task clustering method is introduced. On the basis of factors of resource size and structure of workflow, a level-based autonomous Workflow-and-Platform Aware (WPA) task clustering technique is introduced [33]. It aims at reducing overhead in resource systems and wasting. The overheads in systems are classified into four types of delays: delays in the queue, delays in the workflow engine, delays in job postscript and delays in data transfer. The author worked on the factor of wastage of resources and minimal queuing overhead. Evaluation of the proposed algorithm takes place on actual application workflows such as Cybershake, LIGO, SIPHT.

Horizontal Clustering (HC), Horizontal Impact Factor Balancing (HIFB), Horizontal Runtime Balancing (HRB) and Horizontal Distance Balancing (HDB) used for evaluate the proposed technique. The designed method delivered consistent performance in all the experiments. Therefore, it can be used in workflow structures taking into account the factors of varying runtime and distributing time. All the experiments showed that the proposed approach delivers consistent performance. It can be applied in workflow structures with communication time distributions and varied runtime. Further, aims to identify the appropriate size of resource for optimal task performance. The goal is to implement the workflow engine approach. Zhou et al. [34] presented an approach to reuse and re-purpose workflow. It used the semantic similarity metric between the workflow layer hierarchies. It adopted a clustering technique based on graph skeletons to group layer hierarchies into clusters based on ranking technique. Semantic similitude calculation is performed for activity names, the text of activity discipline, for two activities. It identifies core workflow representation similarities in each cluster. Therefore the workflows for reuse and repurposition are ranked and recommended. In the future,

it aims to consider the text syntactic and more similarity calculation conditions.

2.4 Review on Data Placement Techniques

Zhao et al. [35] and Wang et al. [36] presented an inventive review of literature in the field of data placement. Still in the field of data placement research is persistently growing. The integration and evaluation of existing research present in the data placement field lie a need for a methodological survey of the literature. A complete survey in methodical form for discovering and evaluating research challenges on the basis of existing research in the field of data placement is done.

2.4.1 Classification of Data Placement Techniques

Data placement schemes are classified into different categories as shown in Figure 2.1.

Correlation-based Data Placement

Kosar and Linvy [37] proposed the scheduler for data placement has the ability to queue, schedule, manage and monitor data placement jobs. They also applied the technique of check-pointing jobs which provides complete automation for processing of data in the heterogeneous systems. It possessed the ability to recover from network, software and storage system failures without human intervention. It performs a dynamic adaption of data placement jobs at the execution time to the system environment. Doraimani and Iamnitchi [38] presented a filecule grouping technique for managing data in science Grids while saving the time for locating the data and grouping the file based on its size. LRU-bundle algorithm designed for file staging. Fedak et al. [39] introduced a BitDew programming interface for data management operations as replication, placement and fault tolerance. This architecture relies on independent services to transfer, store and schedule the data.

Yuan et al. [40] proposed an algorithm for the run time and build time stage. In the initial stage dependency between all the data sets is calculated and dependency matrix is built. For the partitioning and clustering of data sets, the Bond Energy Algorithm (BEA) is used. These partitions are shared among different data centers. These data centers are partitioned using k means algorithm. After the generation of intermediate data, the

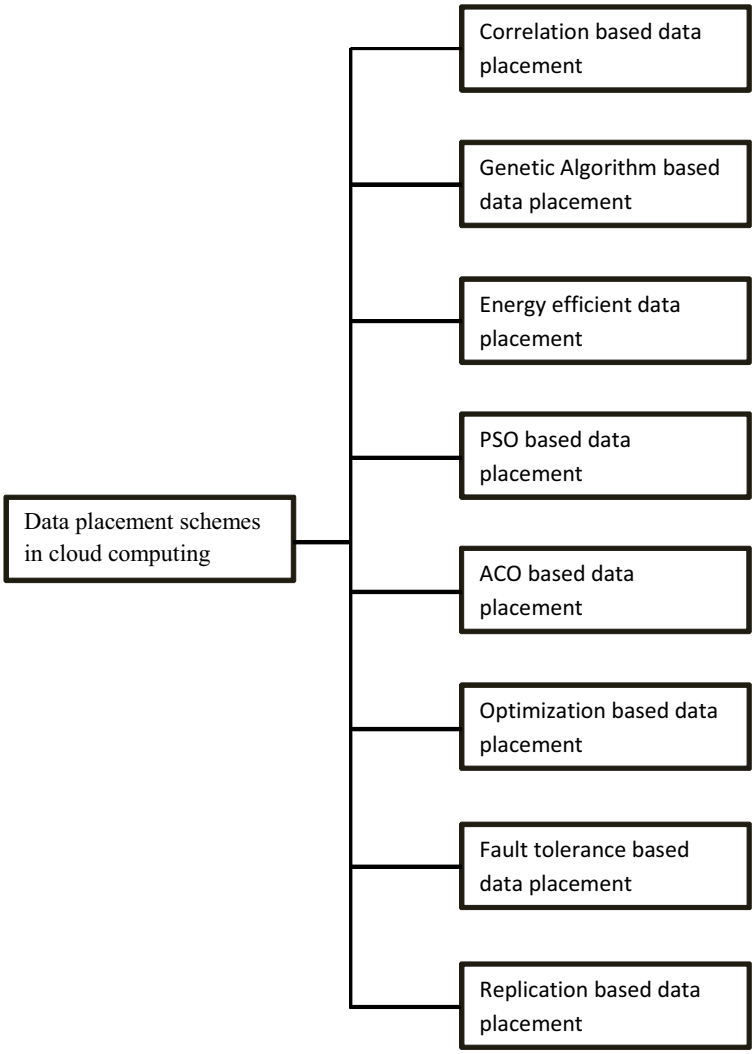


Figure 2.1: Data placement schemes in Cloud computing

newly proposed clustering algorithm deals with new datasets. The dependencies for each data center are judged and then accordingly data is moved. The factors of data movement and gathering of data at one point are covered up. Two algorithms proposed are as follows:

1. Build-Time Stage Algorithm: In this stage dependencies between the tasks is calculated and dependency matrix is built. For the transformation of dependency matrix to clustered dependency matrix, BEA is used. Global measure (GM) for clustered dependency matrix is defined in Eq. 2.1.

$$GM = \sum_{i=1}^{n} \sum_{j=1}^{n} DM_{ij}(DM_{i,j-1} + DM_{i,j+1}) \tag{2.1}$$

Then further partitioned datasets are mapped to data center using binary parti-

tioning algorithm as per Eq. 2.2.

$$PM = \sum_{i=1}^{p}\sum_{j=1}^{p} CM_{ij} * \sum_{i=p+1}^{n}\sum_{j=p+1}^{n} CM_{ij} - \left(\sum_{i=1}^{p}\sum_{j=p+1}^{n} CM_{ij}\right)^2 \tag{2.2}$$

This process continues recursively.

2. Run-Time Stage Algorithm: Using K means algorithm, newly generated data sets are clustered at this stage based on the intermediate dependency. The dependency for all datasets with all data centers is calculated. At last the dataset placement is performed depending on the maximum dependency on a data center.

Table 2.1: Objectives of correlation based data placement schemes

Schemes	**Energy-aware**	**Cost-aware**	**Resource-aware**	**Application-aware**	**Factors**
[38]	-	✓	-	-	Data dependency, File Size
[39]	-	✓	✓	✓	Data Transfer
[37]	-	✓	-	✓	Data dependency
[40]	-	✓	-	-	Data dependency, Data Movement, Execution Time
[36]	-	✓	-	-	Load balance, Execution Time

Table 2.2: Properties of coorelation based data placement schemes

Schemes	**SLA support**	**Security**	**Cloud/ Grid**	**Algorithm**	**Clustering Algorithm**	**Tool/ Programming Language**
[38]	✓	-	Grid	Greedy Request Value	-	Java
[39]	✓	-	Grid	Information Dispersal	-	Java
[37]	✓	-	Grid	-	-	Stock Server
[40]	✓	-	Cloud	Bond Energy	K means	SwinDeW-C
[36]	-	-	Cloud	DCCP	-	cloud application platform (CApp)

Dynamic Computation Correlation Placement (DCCP) proposed for the method of

data placement that is based on dynamic computation correlation [36]. The datasets with highly dynamic correlation placed at the same data center. The factors considered are datacenter capacity load and datacenter input/output load. During the execution of computations, data sets processed are stored on local data centers thus leading to the reduction of execution time. This leads to Input/Output and statistical load balancing. Table 2.1 represents the objectives of correlation based data placement schemes and Table 2.2 represents the properties of correlation based data placement schemes.

Genetic Algorithm Based Data Placement

Er-Dun et al. [41] proposed an algorithm for reduction of movement of data among data centers leading to load balancing in data centers. The heuristic and genetic are combined together for improving the ability to local search and reducing the search time. The heuristic idea is implemented in gene operations and initial population selection. The integer encoding rules are applied and the placement process is represented by a gene. The ineffective fragments of genes that cannot be coded at data centers is decided by using the encoding rule. Fitness function depicts advantages as well as disadvantages of genetic individuals. It shows the degree of dependence between data sets and center load balance. D_{dc} is the total data dependency in the d_c data center defined in Eq. 2.3.

$$D_{d_c} = \sum_{ds_i,\, ds_j \, in \, d_c} D_{ij} \tag{2.3}$$

The fitness function is defined in Eq. 2.4.

$$fitness = \frac{1}{|DC_u|} \left(\sum_{dc \; \varepsilon \; DC_u} D_{dc} \right) \tag{2.4}$$

Where DC_u is data centers set which consist of at least one data set. Larger the value depicts better gene. The overloading of data centers is adjusted by a non-normal chromosome. If the gene is found to be invalid, it refers to the overloading of data centers. So this gene is to be neglected. If the invalidity is depicted at the initial stage then the new gene should be generated. If this occurs at the cross stage then genes to be re-crossed and if it occurs at the mutation stage then genes to be re-mutated.

Optimization of genes is performed for reducing the movement of data. If the data set is fixed location data set then calculate the dependency of each data set with a data center as per formula. If the data set size exceeds the storage capacity of the data center, then the dataset is placed on another data center.

Table 2.3: Objectives of Genetic algorithm energy based data placement schemes

Schemes	**Energy-aware**	**Cost-aware**	**Resource-aware**	**Application-aware**	**Factors**
[41]	-	✓	✓	✓	Data movement, Load balance
[42]	-	✓	✓	-	Storage Capacity, Data Transfer
[43]	-	✓	✓	-	Data dependency, Data movement, Cost

Table 2.4: Properties of Genetic algorithm energy based data placement schemes

Schemes	**SLA support**	**Security**	**Cloud/Grid**	**Algorithm**	**Clustering Algorithm**	**Tool/ Programming Language**
[41]	✓	-	Cloud	Heuristic genetic	-	Cloudsim
[42]	-	-	Cloud	Optimal genetic	-	-
[43]	-	-	Cloud	Hashing	K-Means	-

Guo et al. [42] introduced a mathematical technique on the basis of the genetic algorithm. The crossover rate(P_c), size of population (G) and mutation rate(P_m) is determined. After the initial population generation of BG, the fitness value is obtained for each individual. The fitness value of $F = 1/\Gamma(B_t)$ is indicated using roulette wheel selection. At last individuals are selected using roulette wheel selection [44]. Crossover and mutation are performed on the selected matrix [45]. The individuals not adhering to the requirements of storage capacity are abandoned.

In addition, Zheng et al. [43] introduced a genetic algorithm-based data placement technique to reduce time and data scheduling costs between centers. Authors considered the factors of dependency among slices of data and distributed the cost of transaction

for the placement of data using a genetic algorithm. It minimizes the transaction cost while balancing the load. The author used a roulette wheel for realization model. The fitness function is represented in Eq. 2.5.

$$Fitness = \frac{\sum_{DCT_n} C_{ij}}{|DCT|} \tag{2.5}$$

Where $|DCT|$ represents the total data centers, DCT_n specify the particular data center numbered n and C_{ji} is the total cost of cooperation between data sets j and i. The objectives of Genetic Algorithm energy based data placement schemes is mentioned in Table 2.3 and properties of Genetic algorithm energy based data placement schemes are mentioned in Table 2.4.

Energy Efficient data placement

In literature, many authors proposed energy saving methods at a hardware level such as processor speed adjustment, voltage settings, enlarging memory, etc. [46, 47, 48, 49, 50]. However, these methods are unable to reach maximum energy optimization, saving of energy by these methods is comparatively less than turning the computer off. These methods are only limited to a single node.

Xiao et al. [51] proposed a heuristic algorithm for data placement and two node scheduling techniques in order to save consumption of energy during the execution of tasks. In the proposed algorithm data blocks are kept rational for finding the minimum set of nodes containing the collection of blocks of data. The goal of energy saving is achieved by turning on minimum nodes required covering the maximum data blocks. A greedy algorithm employed for covering data block with the computing node. The author addresses two goals in it. First is the power consumption upper bound is known, accordingly execution time of task requests by node scheduling is minimized. Second, the execution time of tasks is known as per the execution time of task requests by node scheduling is minimized. This optimization of batch scheduling achieves energy saving effect by providing a solution for time constrained and power restrained issues in cloud data centers. Each node $s \, \varepsilon \, S$, in node s, q is the number of data blocks, p is the number of data blocks which meet the job requests, so node cover rate defined as per Eq. 2.6.

$$\gamma(s) \; = \; \frac{p}{q} \tag{2.6}$$

If node s and node s_t have the same data block replica, then node s and node s_t are data-exchangeable. The computation of resource utilization of nodes as per Eq. 2.7.

$$U = (e \times U_{cpu}) + ((1-e) \times U_{disk}) \tag{2.7}$$

where U_{cpu} stands for utilization of cpu, u_{disk} stands for utilization of disk and e is scale factor [51].

Pinheiro et al. [52] proposed Popular Data Concentration (PDC) algorithm. It dynamically migrated mostly used data on the disk to a different subset of disks in an array. The purpose is to transfer the load to few disks so that other disks can be sent to the mode of energy saving. Maheshwari et al. [53] addressed conservation fo energy for a cluster of nodes that execute Map Reduce jobs. Reconfiguration of clusters is performed on the basis of the current workload. When the average utilization rises, clusters are turned on or off.

Table 2.5: Objectives of energy efficient based data placement schemes

Schemes	**Energy aware**	**Cost aware**	**Resource aware**	**Application aware**	**Factors**
[51]	✓	-	✓	-	Power usage, Execution time
[52]	✓	✓	-	-	File characteristics, workload characteristics
[53]	✓	-	-	-	Power consumption, cost
[54]	✓	✓	-	✓	Execution cost, power usage

Table 2.6: Properties of energy efficient based data placement schemes

Schemes	**SLA support**	**Security**	**Cloud/ Grid**	**Algorithm**	**Clustering Algorithm**	**Tool/ Programming Language**
[51]	✓	-	Cloud	Dynamic Algorithm	-	Cloudsim
[52]	-	-	File server	Popular data concentration	-	Execution simulator
[53]	-	-	Grid	Cluster reconfiguration	-	Gridsim
[54]	-	-	Cloud	EnCloud algorithm	-	iVIC with Phython

Li et al. [54] proposed EnaClous approach that enables dynamic live application placement considering the energy efficiency. In this approach, the virtual machine is used for encapsulation of an application that performs scheduling of an application and lives migration of an application for saving energy. Bin packing algorithm is used for application placement. The author proposes an energy-aware heuristic algorithm. For dealing with the varying resource demands, the investigator presented an over provision technique.

Beloglazov et al. [55] proposed an architectural framework based on resource allocation and principles for energy efficient green cloud computing. It introduced a virtual node placement scheme and decides to turn on some physical nodes of the active virtual nodes, but the data placement problem is not considered.

The objectives of Energy efficient based data placement schemes are specified in Table 2.5 and the properties of energy efficient based data placement schemes is specified in Table 2.6.

PSO based data placement

Yin et al. [56] designed the optimal assignment of tasks in a reasonable time are obtained using the proposed hybrid particle swarm optimization approach. PSO based algorithm is represented to solve Task Assignment Problem (TAP). PSO iteration embedded with hill climbing heuristic for convergence. The factors considered are execution and communication costs.

Guo et al. [57] proposed a method based on particle swarm optimization leading for cost reduction and transfer time execution. The investigator works on the on-demand method as the standard for computing. This refers to paying per on per hour basis with no long term commitments. According to the charging standard of Amazon. P_{out_k} is defined as pricing of the data center from DC_k, p_{in_t} as pricing of data transfer to DC_l, P_k is the processing pricing of standard on-demand and T_p is the time of execution of tasks. The task performed on data center k is i. Cost of data processing calculates as per the Eq. 2.8.

$$C_p = T_p \times P_k \tag{2.8}$$

Cost of data transfer is represented as per Eq. 2.9.

$$C_t = \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{m} \sum_{l \neq k} x_{ik} \times x_{jl} \times (DT_{ij} \times P_{out_k} + DT_{ij} \times P_{in_l}) \tag{2.9}$$

Fitness function is sum of cost of data transfer and processing represented as per Eq. 2.10.

$$C = C_p + C_t \tag{2.10}$$

According to the best fitness value, data is placed at the data center.

Li et al. [58] proposed a cost-effective data placement scheme for multi-data centers. The data sets are transferred from one data center to another during the execution of workflows. So mapping of these data sets to an appropriate data center is the issue addressed by the author. The focus is on reducing the cost of data transfer by considering multiple workflows. Let $DC = \cup_{i=1,2,...|DC|} dc_i$ be data center set, G_i is a single workflow and datasets of multiple workflows (MWs) are represented as $DS = \cup_{i=1,2,...n} ds_i$ Data placement of multiple workflows $MWS = \cup_{i=1,2,..n} G_i$ is represented with following Eq. 2.11.

$$M_{multi} = \cup_{i,j=1,2,....,|DS|} d_i \rightarrow dc_j \tag{2.11}$$

where $dc_j \varepsilon DC \forall d_i \varepsilon DS$, d_i is stored in unique data center in M_multi.

Guo et al. [59] introduced PSO based algorithm embed in Small Position value (SPV)[60]. The aim is to optimize the scheduling of tasks in cloud computing for minimization of processing cost. The algorithm relies on a small position value. The focus is on reducing execution and transferring time. The fitness function for an algorithm is taken as a combination of cost and time $CT = T + C$ [57].

Pandey et al. [61] presented heuristic based Particle Swarm Optimization (PSO) to schedule applications on cloud resources. Consideration is given to the factors of cost of data transmission and computation. Mapping of tasks to the resources is accomplished depending upon varying the cost of computing and communication.

Zhao et al. [35] clustered datasets on the basis of dependency using the proposed hierarchical clustering based technique. A new factor as the size of the dataset is introduced. This leads to a reduction in data movement. In the partitioning matrix, improved dichotomy algorithm is used. For the mapping between data groups and servers

PSO based algorithm is used. The objectives of PSO based data placement schemes is represented in Table 2.7 and properties of PSO based data placement schemes are represented in Table 2.8.

Table 2.7: Objectives of PSO based data placement schemes

Schemes	**Energy-aware**	**Cost-aware**	**Resource-aware**	**Application-aware**	**Factors**
[56]	-	✓	-	-	Execution and Communication cost
[57]	-	✓	-	-	processing time, transferring time, processing cost, transferring cost
[58]	-	✓	✓	-	Data dependency, Data Transmission Cost
[59]	-	✓	✓	-	Execution time, Transferring time, Cost
[61]	-	✓	-	-	Execution cost, Communication cost
[35]	-	✓	-	-	Data dependency, Data Movement

Table 2.8: Properties of PSO based data placement schemes

Schemes	**SLA support**	**Security**	**Cloud/Grid**	**Algorithm**	**Clustering Algorithm**	**Tool/ Programming Language**
[56]	✓	-	-	Hybrid-PSO	-	-
[57]	✓	-	Cloud	PSO	-	Matlab
[58]	✓	-	Cloud	PSO	-	-
[59]	✓	-	Cloud	PSO-SPV	-	Cloudsim
[61]	✓	-	Cloud	PSO	-	-
[35]	-	-	Cloud	PSO	Bond Energy (BEA)	Visual Studio 2010

ACO based data placement

Liu et al. [62] presented data placement scheme for intermediate data placement after considering the factor of security. The security of data is based on aspects of data integrity, authentication access, and data confidentiality. A model of security quanti-

tatively measures the services of data center security. Ant Colony Optimization(ACO) based technique is used for dynamic selection of data center. In this study, the datasets are divided into the flexible location and fixed location data sets according to the location of stored data. For the evaluation of security services, Degree of Data Security Deficiency (DDSD) is presented. The objectives of ACO based data placement schemes are listed in Table 2.9 and properties of ACO based data placement schemes are described in Table 2.10.

Table 2.9: Objectives of ACO based data placement schemes

Schemes	**Energy-aware**	**Cost-aware**	**Resource-aware**	**Application-aware**	**Factors**
[62]	-	-	✓	-	data security, data transfer time
[58]	-	✓	✓	-	Data dependency, Data Transmission Cost
[59]	-	✓	✓	-	Execution time, Transferring time, Cost
[61]	-	✓	-	-	Execution cost, Communication cost

Table 2.10: Properties of ACO based data placement schemes

Schemes	**SLA support**	**Security**	**Cloud/Grid**	**Algorithm**	**Clustering Algorithm**	**Tool/ Programming Language**
[62]	✓	✓	Cloud	ACO	-	Cloudsim
[58]	✓	-	Cloud	PSO	-	-
[59]	✓	-	Cloud	PSO-SPV	-	Cloudsim
[61]	✓	-	Cloud	PSO	-	-

Optimization based data placement

Agarwal et al. [63] proposed system considers the factors of WAN bandwidth, data center capacity limit, data interdependency, user mobility and application changes. The proposed approach analyzes the data centers logs. It used an iterative optimization algorithm based on access patterns of data, locations of clients. Further, it outputs recommendations of migration back to cloud service. For scaling the large volume of

data logs, this technique works in scope [64]. This algorithm works in three phases. Mapping of each client to a set of geographical candidates is performed using weighted spherical mean calculation as per Eq. 2.12.

$$wsm((w_i,\vec{x}_i)_{i=1}^{N}) = interp\left(\frac{w_n}{\sum w_i},\vec{x}_N, wsm(w_i,\vec{x}_i)_{i=1}^{N-1}\right) \tag{2.12}$$

It handles the complexities of data inter-dependencies and shared data. In the second phase, improvement in the placement is done by applying the proposed approach. In the third phase, the mapping of data to the data center is performed by considering the factor of the storage capacity of the data center.

Cataly urek et. al. [65] aims to place data files into and assigning tasks to the sites of execution for reducing the cost while considering weights. To accomplish this, the workflow modeled as a hypergraph. A heuristic based on hypergraph partitioning is proposed for generating appropriate placement of data and assignment of the task. According to the proposed technique, computational and storage loads are distributed evenly according to some pre-decided ratios. The hypergraph partitioner is implemented by modifying PaToH [66].

In the multilevel framework, net costs are incorporated into PaToH in three phases. In the first phase of coarsening of net costs if performed. In the phase of initial partitioning, the algorithm modification of GHGP is performed to be used with target weights and net costs [66]. In the last phase of refinement, heuristic FM is modified for accurate calculation of the vertex move gains and cutsize [67].

Yuan et. al. [40] proposed an algorithm for each build time and run time stage. In the initial stage, the dependency between all the data sets is calculated and dependency matrix is built. For the partitioning and clustering of data sets, the Bond Energy Algorithm (BEA) is used. These partitions are distributed among different data centers. The data centers are partitioned using the k-means algorithm. After the generation of intermediate data, the newly proposed clustering algorithm deals with new datasets. The dependencies for each data center are judged and then accordingly data is moved. The factors of data movement and gathering of data covered up a single point. This strategy allocates application data among data centers automatically and reduces the movement of data.

Pandey et al. [68] proposed the Non-linear Programming (NLP) model for minimization of retrieval of data and execution cost of workflows. The proposed technique aims at minimizing the computation cost and data transfer cost on the computing resource. For a case study, the intrusion detection application is considered. While using the storage and compute resources, the NLP model is applied to intrusion detection application. The objectives of optimization based data placement schemes are mentioned in Table 2.11 and the properties of optimization based data placement schemes are mentioned in Table 2.12.

Table 2.11: Objectives of optimization based data placement schemes

Schemes	**Energy aware**	**Cost aware**	**Resource aware**	**Application aware**	**Factors**
[63]	-	-	✓	-	storage capacity, data interdependency, WAN bandwidth, IP address
[65]	-	✓	✓	-	Communication cost
[40]	-	✓	-	-	Data dependency, Cost
[61]	-	✓	-	-	Execution cost, Communication cost
[68]	-	✓	-	-	execution cost
[69]	-	✓	✓	-	execution cost, execution time, data dependency

Table 2.12: Properties of optimization based data placement schemes

Schemes	**SLA support**	**Security**	**Cloud/Grid**	**Algorithm**	**Clustering Algorithm**	**Tool/ Programming Language**
[63]	✓	-	Cloud	iterative optimization	Cloud	-
[65]	-	-	Cloud	combinatorial algorithm	-	C programming language
[40]	✓	-	Cloud	Bond Energy	K-Means	SwinDeW-C
[61]	✓	-	Cloud	PSO	-	-
[68]	✓	-	Cloud	NLP	-	Amazon CloudFront

Fault-tolerance Data Placement

Li et al. [70] proposed the strategy based on the clustering algorithm of consistent hashing and minimum distance. The data is clustered efficiently with the clustering algorithm by placing item-based and user-based data. For addressing the effect of outliers and noises, cluster centers and threshold is updated. Item-based algorithm and CBR strategy fill a sparse matrix of users [71]. The consistent hashing algorithm is used for improving fault toleration and scalability.

Table 2.13: Objectives of fault tolerant based data placement schemes

Schemes	**Energy-aware**	**Cost-aware**	**Resource-aware**	**Application-aware**	**Factors**
[70]	-	✓	✓	-	Fault tolerance, Execution Time
[37]	-	✓	-	✓	Data dependency, Fault tolerance

Table 2.14: Properties of fault tolerant based data placement schemes

Schemes	**SLA support**	**Security**	**Cloud/ Grid**	**Algorithm**	**Clustering Algorithm**	**Tool/ Programming Language**
[70]	✓	-	Cloud	Consistent Hashing	K means	VMware
[37]	✓	-	Grid	-	-	Stock Server

A Data placement scheduler, Stork provide an ability to queue, schedule, manage and monitor the data placement jobs. It also applies the technique of checkpointing jobs. This approach provides complete automation for processing the data. The proposed system makes the data transfer fully automatic in the heterogeneous systems. It possesses the ability to recover from network, software and storage system failures without human intervention. It performs a dynamic adaption of data placement jobs at the execution time to system environment [37]. The proposed technique uses checkpointing technique along with scheduling and managing the jobs in data placement. The technique is best suited for the hetrogenous systems. Table 2.13 represents the objectives of fault-tolerant based data placement schemes and Table 2.14 represents properties of fault-tolerant based data placement schemes.

Replication-based Data Placement

Bharathi and Chervenak [72] presented a heuristic based on ordering jobs cleaned up in workflow and reduce time taken for execution. A genetic algorithm based approach vary the amount of storage and number of processors for generating schedules with low cost. Tightly coupled data staging approach is used and also introduced DPS and workflow manager models to work together for fulfilling the requirements of data management.

Wei et al. [73] introduced Cost-effective Dynamic Replication Management (CDRM) scheme for reducing cost by capturing the relations between the availability of data center and replica number. Its aims to increase the availability of data for the storage system of the cloud for balancing the load and improving performance while considering failures. CDRM instincts proposed a model for maintaining a minimum number of replicas according to requirement availability. Placement of replica is on the basis blocking capacity and probability of data nodes. CDRM dynamically redistributes according to changing workload and capacity of the node.

The author proposed a Two-level DHT (TDHT) approach to apply Lazy update and minimize the cost of communication [74]. Trading between availability and safety is performed. This approach is integrated with DHT, known as Global DHT (GDHT).

Myint et al. [75] used the replication strategy to achieve network bandwidth availability, reliability, and adequate usage. The PC cluster system applied for cloud storage system implementation.

Huang et al. [76] proposed a new interconnection network MyHeawood for cost-effective data placement. It is based on different hashing functions on a hierarchical network. It consists of a small switch and dual NIC server. Data placement strategy is based on the hashing function composed of different hash functions. Hash functions are composed of a hash key for computing server address for a master replica. After that, on the basis of the address of master replica, remaining replicas are allocated in a different layer. Three replicas are used for solving overhead of storage caused by multiple replicas. This solves the issue of reliability in the cloud. Further, Pandey [77] improved this technique with a replication method.

Casas et al. [78] designed a Balanced and Reuse-Replication Scheduling (BaRRS) method for computing in the cloud. This algorithm divides the scientific workflow among multiple workflows for balancing through parallelization. The main objective is

to deals with the reuse of data and different replication techniques for optimization of transferable data. It considers the execution time of the task, patterns of dependency among tasks and size of files for adapting to current techniques of replication and data reuse. At last, it selects the optimal solution on the basis of monetary cost and execution time. Table 2.15 represents objectives of Replication-based data placement schemes and Table 2.16 represents properties of Replication-based data placement schemes.

Table 2.15: Objectives of replication based data placement schemes

Schemes	**Energy-aware**	**Cost-aware**	**Resource-aware**	**Application-aware**	**Factors**
[72]	-	✓	-	-	Makespan, execution cost, storage limit
[37]	-	✓	-	✓	Data dependency, Fault tolerance
[73]	-	✓	-	-	CPU power, memory capacity, network bandwidth, access latency, load balance
[74]	-	✓	-	-	Response Latency, Cost
[76]	-	✓	-	✓	Cost
[78]	-	✓	-	✓	Execution time, Data dependency, Task Size

Table 2.16: Properties of replication based data placement schemes

Schemes	**SLA support**	**Security**	**Cloud/Grid**	**Algorithm**	**Clustering Algorithm**	**Tool/ Programming Language**
[72]	-	-	Grid	Coarse and fine grained genetic algorithm	-	Cluster viz16
[37]	✓	-	Grid	-	-	Stock Server
[73]	✓	-	Cloud	CDRM	-	-
[74]	✓	✓	cloud	uTLA	-	PlanetLab (Plab) platforms
[76]	-	-	Cloud	Hashing	-	Cloudsim
[78]	-	-	Cloud	BaaRS	-	VMware-ESXi-based (version)

2.5 Review on Scheduling Algorithms

Kang et al. [79] investigated the issue of minimization of financial cost for allocation of virtual resources and proposed a heuristic based algorithm. It consists of two phases of VM packing and Multiple requests to a single resource (MRSR). It considers the factor of quality of service assurance. It experimented the phases with an OpenStack based cloud platform and found 30%. Rodriguez et al. [80] introduced a technique based on swarm optimization (PSO) [81] to deal with resource provisioning and scheduling in the cloud. The main aim is to reduce the execution cost of workflow along with deadline constraints. It includes four basic principles of a model of heterogeneity, pay-as-you-go, elasticity and resource dynamics. The factors such as VM boot time and performance variation are also considered. It performed the evaluation on CloudSim with four other workflows and reached a conclusion that the proposed algorithm out-performed SCS [82] and IC-PCP state-of-the-art algorithms [83]. In each case, the designed approach succeeded, while IC-PCP failed to meet deadlines and is capable of producing a schedule with a lower cost of implementation. It analyzed that selecting a resource pool initially enhances the performance and will work on different options for selecting it. It aims to extend the genetic algorithms approach and implement the approach in a workflow engine to make deployment easier.

On the basis of cloud service platform, for future manufacturing, introduced the architecture of a scientific workflow management system [84]. It devised a new scheduling method Max Percentages (MP) for static heterogeneous resources within the system, resulting in increased overall system performance. The algorithm's performance is compared to other algorithms such as classic algorithms such as Min-Min, DCP, GA and Max-Min. The effective performance is provided by the MP algorithm than the other algorithms, but it ignores the security factor. Wu et al. [85] formulated the problem of budget-constrained workflow scheduling. It developed MED-CC, a prototype for a generic workflow system incorporating existing technologies of the workflow. It introduced MED-CC, a heuristic Critical-Greedy algorithm. The performance of the algorithm is shown over existing methods. Its performance was compared with ScaleStar and HBCS, which showed 10 percent improvement over ScaleStar and 25 percent better than HBCS. Zhao et al. [86] designed the system for the heterogeneous environment

to improve the heuristic data placement algorithm. It presented clustering based on data dependency and recursive portioning in the initialization phase. It considered the factors of data size and fixed position. It extended the tree-to-tree heuristic data placement technique for the frequently occurring movements of data on high bandwidths in centers. The comparison is made with two classical strategies and results depicted that the proposed strategy reduced the size and time for data transmission during execution. It introduced a heuristic data layout method works in two phases. In the first phase, the abstraction of both the datasets and the cloud system are done as tree-structured models considering the factors of data correlation and network bandwidth. It extended the heuristic data allocation method for making data movements frequent on high bandwidth networks so as to reduce communication time of data. Results of simulation show that during execution, the introduced strategy reduces the data movement and time required for processing. Further placement factors such as each server's computing capacity and load balancing can be considered in the future. In terms of system reliability and response time, replicating the user data can also be used for good performance.

Bryk et al. [87] investigated the issue of scheduling workflow sets while considering infrastructure costs and deadline constraints as a service cloud. It addressed the ignored file transfers between workflow tasks that have a high impact on workflow ensemble execution. A simulation model was proposed to handle file transfer between tasks. It dynamically calculated the bandwidth and supported a large number of replicas resulting in different congestion levels being simulated. By data caching and file locality, the scheduling technique is introduced to reduce the transfers. It proposed both static and dynamic scheduling and resource provisioning algorithm. It analyzed in a virtual machine to task variance factors or temporary delay. The Storage and Workflow-Aware DPDS (SWA-DPDS), Storage-Aware DPDS (SA-DPDS) and Storage-Aware SPSS (SA-SPSS) scheduling algorithms are developed. They are based on the function of runtime prediction. It introduced file location-aware scheduling algorithm that produced better schedules by taking file location into consideration when submitting VMs task caches are examined. VM is selected on which the task is predicted to be completed earlier while estimating runtime and file transfers. The performance of the algorithms was good. Additional improvements can be made to estimate the file transfer process. It is also possible to introduce safety margins in SPSS algorithm. The storage

billing model is still under development for storage. It is possible to extend this work to the hybrid cloud landscape.

Ebrahimi et al. [88] introduced meta-heuristic based Big Data Placement Technique (BDAP) to improve workflow execution by reducing data transfers between multiple virtual machines in the cloud. Scientific workflows formalize the problem of data placement across multiple virtual machines. It proposed a data placement algorithm, taking into account the initial input data set and intermediate data sets obtained during the workflow run. In the distributed environment, it evaluated the proposed technique and assessed its solution for effective data placement at suitable virtual machines in the cloud in minimal time. BDAP clusters and stores the most interdependent datasets in the same virtual machine. It is generated after this random set of data placement schemes. A heuristic function applied in the second step to calculate and compare generated schemes and return the best scheme. The two constraints that are considered while placing are non-replication constraint and virtual machine storage capacity. In the future, it aims to implement data replication for reducing data movement. It also found that multiple workflows can be considered simultaneously.

Casas et al. [89] proposed a Balanced and File Reuse-Replication Scheduling (BaRRS) algorithm for deploying scientific workflows. It includes three strategies for balancing queues, reuse of files. It considers the factors of time of execution and monetary cost. It analyzed the features of scientific workflows applications like dependency pattern, the execution time of task and sizes of files. BaRRS is compared to a different scheduling method using four different scientific workflows with different dependence patterns and sizes of data files. It delivers promising results. Li et al. [90] proposed SCAS, a PSO method dependent on factors of security and cost for scheduling. It addressed the multi-constraint and multi-dimensional optimization issue. It aimed to reduce execution cost while complying with limitations of deadline and risk. In addition, experiments are conducted using Cloudsim framework to demonstrate the algorithm performance.

2.6 Summary

The detailed understanding of different task clustering techniques has been discussed based on levels, labelling and similarity between tasks of workflow. Apart from task clustering techniques, this chapter provides an insight into numerous data placement

schemes to place the datasets at appropriate location in data center with metrics of size of data set, bandwidth of data center and storage capacity of data center. This chapter also specifies the number of scheduling methods to schedule the scientific workflow in order to minimize the execution time of workflow with different parameters such as user budget, deadline, cost, etc. The prominent workflow management systems are also discussed in detail.

It has been concluded from the literature survey that still there is a scope of improvement in QoS and resource management for scientific workflows in cloud environment. The optimization of data movements and appropriate scheduling algorithms can overcome the fundamental issues of workflow execution in cloud data centers. Task clustering can reduce the scheduling overhead, but the clustering overhead is again a crucial factor which need to contemplate while designing the clustering techniques.

CHAPTER 3

HYBRID BALANCED TASK CLUSTERING FOR SCIENTIFIC WORKFLOW IN CLOUD COMPUTING

3.1 Introduction

In past years of scientific discovery, the computational workflow continues to be popular among various disciplines of science, including astronomy, physics, biology, seismology, chemistry, and others [91].

A workflow is a series of activities representing business processes or computational science with existing dependencies between them. These dependencies need to be satisfied with the achievement of a goal [92]. Business workflow is a control-flow driven activity including constructs for specifying conditions, paths and also involve human interaction. It implements the company's services or products. The scientific workflow involves large scale data and/or complex computations, therefore, utilizes computing resources and storage capacities [93]. It does not involve control-flow, but it is data-driven and still exceptions (e.g. Askalon) are persisting [94].

A large amount of data processing is required by scientific workflows that consist of millions of uncommon tasks [95]. For example, the Cybershake workflow [96] containing 800,000 tasks are executed on TeraGrid [97]. These loosely coupled applications represent a considerable amount of data and computation [98]. Existing applications such as Condor [99] do not consider overheads in the system, fault occurrence or restructuring of a workflow.

Workflow restructuring technique such as task clustering is introduced to improve the scalability of the system and reduce overhead system costs [100, 101, 102]. It is a process of merging smaller tasks into a larger job [100] which is the single unit of execution in a workflow management system. After the task clustering on sceintific workflow application, the execution units are reduced. Which in turns leads to an increase in application computation and reducing system overheads.

However, various existing methods used an approach for the optimization of workflow structures. For example, Horizontal Clustering (HC) [100] merge the tasks at different levels of workflow horizontally. For a single task, the horizontal level is defined as the largest distance from the start task of the Directed Acyclic Graph (DAG) to this task. The user controls the granularity of clusters that are defined as the number of tasks. It defines either the total clustered jobs per horizontal level or jobs per task group. Such techniques ignored the dynamic characteristics of distributed environments [103].

Many methods are introduced for reducing the system overhead and clustering the tasks either horizontally or vertically but none of the technique employs both kinds of clustering simultaneously. The structure of the workflow plays a significant role in clustering of tasks of a workflow.

In the present study, the proposed Hybrid Balanced (HYB) task clustering algorithm takes into account both the number of jobs available for clustering tasks and workflow structure. Further, the tasks with the parent-child relationship are clustered vertically and other tasks in the workflow are clustered horizontally according to horizontal runtime balancing and horizontal distance balancing methods. Hence, this helps in reducing systems overheads and faster execution of tasks with minimum wastage of resources. The important points considered in the proposed work are:

- Minimum tasks overheads: The overheads are reduced to the minimum, while the tasks with a single parent-child relationship are clustered into one job. Hence, the

dependency time of tasks reduced to a significant level.

- Minimum resource wastage: The proposed algorithm ensures that the dependent tasks are provided with the data required as early as possible in order to avoid an increase in waiting time and wastage of resources.

The remainder of the chapter is arranged as follows. The overview of related work is outlined in section 3.2. Section 3.3 describes the system architecture of clustering. Section 3.4 describes the proposed algorithm. Section 3.5 reports the performance evaluation, results of the proposed technique along with available basic clustering techniques. Section 3.5 presents the summary of chapter.

3.2 Related Work

3.2.1 Load Imbalance

Load balancing is a critical topic in distributed computing. To balance the computational load dynamically among different resources, the transfer of some jobs is required from one resource to another in a period of time. This is called task reallocation [104]. A reallocation algorithm is proposed by Zhang et al. for tuning the parallel jobs submitted to the resource manager. The batch scheduler sends submission and cancellation requests. It dynamically migrates processes from overloaded computational nodes to less loaded nodes in a multi-cluster environment [105]. However, it exhibits the limitation to maintain balance only with some idle nodes. In our case, we consider more tasks then available compute resources. To handle load imbalance, Zhen et al. [106] presented techniques to split tasks dynamically and consolidate them to fill idle compute nodes. Similarly, Ying et al. [107] present a load balancing algorithm based on collaborative task clustering. Sahni et al. introduced level-based autonomous Workflow-and-Platform Aware(WPA) task clustering method that considers the factors of the size of the resource and also the structure of workflow [33].

In comparison to the techniques discussed above, proposed work merges the tasks based on their runtime distribution also considering the data dependencies. Also, an approach to dynamically select the order of clustering whether vertical or horizontal is proposed.

3.2.2 Granularity

In scientific workflows, the techniques to control the granularity of tasks is also addressed. A label-based and level-based clustering approach is proposed by Singh et al. [100]. Considering the same horizontal level tasks are clustered according to level-based clustering. The number of tasks in a cluster is specified by the user. In another approach of label-based clustering, the labeling of tasks is accomplished manually by the user. This method is more prone to error due to manual interaction. A task grouping and ungrouping algorithm are proposed by Ferreira et al., where information about an application and resources is not known in advance [108]. This work does not consider data dependencies but reduces queuing and scheduling overhead. An algorithm is proposed by Muthuvelu et al. [109] that group tasks based on their runtime to resource capacity. Muthuvelu et al. proposed another method [110] to determine the granularity of task based on CPU time, resource constraints and task file size. Muthuvelu et al. also introduced an online scheduling algorithm to cluster tasks based on the user's budget, application deadline, and resource network utilization [111]. Ng et al. [112] aimed to increase the performance in the scheduling of tasks by introducing a factor of bandwidth. Further, Liu and Liao et al. [113] proposed a technique for executing fine-grained jobs by grouping tasks considering the processing capacity of available resources and bandwidth. It presented a method to reuse and repurpose a workflow and uses the semantic similarity metric between a workflow's layer hierarchies. This technique ranked the clusters. The similarity computation used is dependent on syntactic variations [114].

Because the execution of a scientific workflow involves a large number of processes. This can lead to high failure levels. A general model of task failure is proposed by Chen et al. using estimates based on maximum likelihood to improve the performance of scientific workflows execution time [115]. The combination of horizontal and vertical clustering does not take advantage of this framework.

3.2.3 Structural Similarity

Koohi et al. presented a method based on Shuffled Frog Leaping Algorithm(SFLA) for the encoding of workflows through workflow representations by exploiting set descriptors [116]. Zhao et al. [114] proposed a method for reusing and repurposing of a

workflow by calculating the semantic similarity between layers of different workflows. The hierarchies in the workflow are grouped into clusters. Silva et al. proposed a simi-Flow architecture for supporting clustering of workflows based on similarity [117].

3.2.4 Data Dependency

The proposed techniques significantly decrease the overhead and scheduling impact of queuing time but the factor of data dependency is still ignored. The horizontal clustering of tasks increased problems of dependency imbalance and runtime imbalance among tasks. To overcome these problems, Chen et al. introduced three new methods as Horizontal Impact Factor Balancing(HIFB), Horizontal Runtime Balancing(HRB) and Horizontal Distance Balancing(HDB) [32]. In these algorithms, only horizontal clustering is performed. In a workflow, there can be tasks with single parent single child relationship. In these kinds of tasks, vertical clustering can prove to be more advantageous then horizontal clustering technique. A general model of task failure is proposed by Chen et al. using estimates based on maximum likelihood to improve the performance of scientific workflows execution time [115]. This framework fails to take advantage of the combination of horizontal and vertical clustering considering the single parent and child relationship in the nodes of a workflow. The deciding factor is unexplainable in research so as to cluster tasks vertically or horizontally in order to maintain parallelism.

The existing work suffers from one or the following drawbacks

- The data dependencies between tasks not considered.
- The runtime imbalance and dependency imbalance not considered.
- The structure of workflow not considered.
- The maximum parallelism between tasks not exploited.

This work proposes a Hybrid balanced task clustering technique to perform clustering while maintaining the parallelism of the system considering the single parent single child relationships in the nodes of a workflow. Hybridizations of horizontal and vertical clustering technique have been achieved using impact factor based clustering

technique. The cluster size is dynamically set as per job runtime to maintain the parallelism. Dependency variance is calculated using distance metrics. Horizontal and vertical clustering is performed as per the available resources so that the parallelism is not affected. Hybrid clustering improves scientific workflow performance in cloud computing and provides an additional benefit to data placement.

3.3 System Architecture

3.3.1 Workflow

The Workflow Management Coalition (WfMC) defined a workflow as a method where tasks are transferred from user to user according to protocols [2]. By WfMC's definition, the work process is a progression of organized exercises and business process calculations that is a unique representation of the undertakings to be carried out in that business process. They are utilized to join a few diverse computational procedures into a solitary lucid procedure. The business applications can now be seen as perplexing work processes, which comprise of different changes performed on the information desired to accomplish the goal. Workflows offer awesome points of interest in isolating capacities from applications. In this manner, it offers the data segmentation framework to arrange and incorporate the workflows in cloud data centers.

To recognize the interfaces within this structure, WfMC introduces its reference model that empowers items to operate interactively at different levels. The workflow management system is characterized as in Figure 3.1.

- **Workflow Engine:** A service of software providing the runtime environment with a specific end-term goal of making, supervising and executing cases of work processes.
- **Process Definition:** Specifies the information about the process and the workflows related to it.
- **Workflow Interoperability:** This interface makes interoperability possible between different processes of a workflow.
- **Invoked Applications:** Interfaces to strengthen cooperation with an assortment of IT applications.

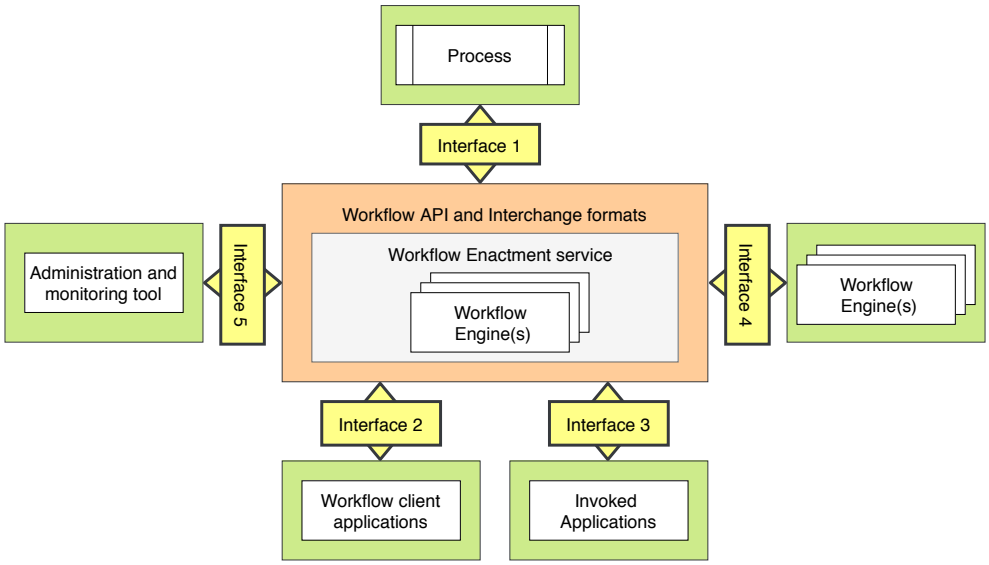


Figure 3.1: WfMC reference model [2]

- **Workflow Client Applications:** It is an interface to strengthen the client interface connection.
- **Administration and Monitoring:** It is an interface to provide an observing framework and metric capabilities to encourage the administration of application situations for composite work processes.

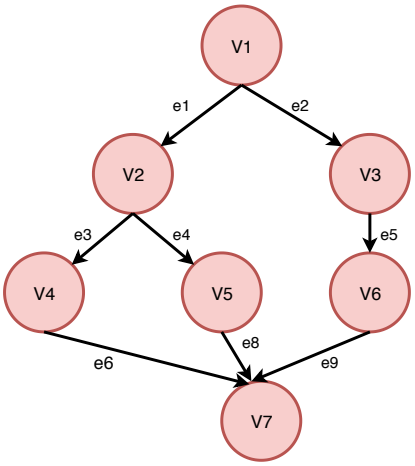


Figure 3.2: Workflow model

3.3.2 Workflow Model

A $Wf=(V_i, E_i)$ workflow application is represented as a Direct Acyclic Graph (DAG) where $V_i=v_{i1},v_{i2}..v_{in}$ is a set of vertices representing tasks and E_i represents control

edges or data dependence between them. A dependency e_{ij} is the precedence constraint of the form (v_{i1},v_{j1}) ,where v_{i1},$v_{j1} \in V_i$ and $v_{i1} \neq v_{j1}$. This refers to that the child task can only complete its execution until the parent task has completed the execution. An example of workflow is shown in Figure 3.2

3.3.3 Workflow Execution Environment

A workflow is submitted for execution to the workflow management system that resides on a user interaction machine. The execution machine for a workflow is a grid, physical cluster [118], a dedicated parallel system [96], a virtual environment such as the cloud [119] or it can also be a local machine. Figure 3.3 shows environment of execution for scientific workflows. The components of this environment are listed below:

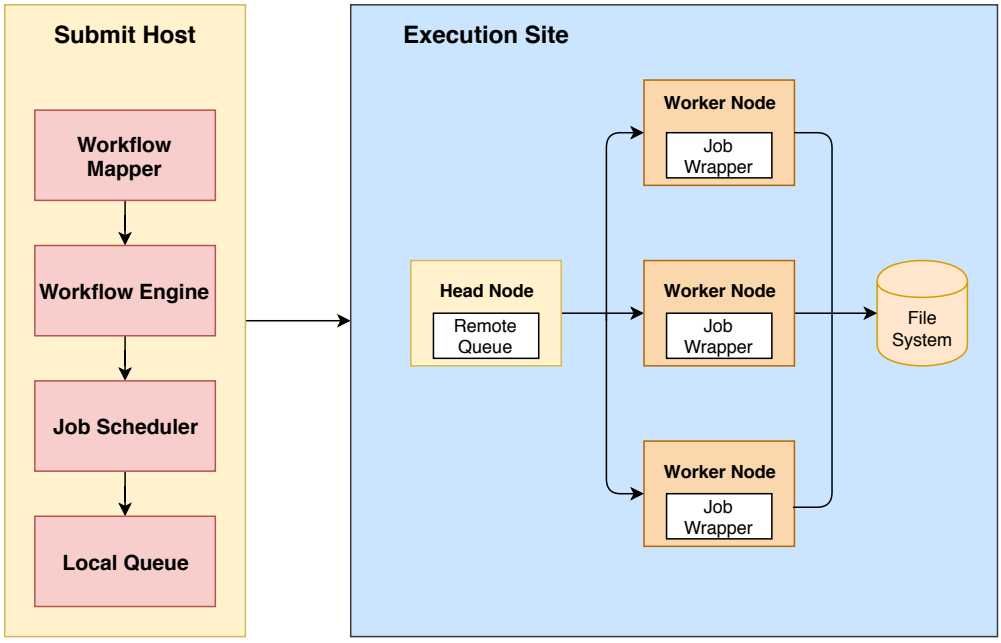


Figure 3.3: Workflow management system

Workflow Mapper: On the basis of abstract workflow generates an executable workflow. The abstract workflow is submitted by the workflow user. This component maps the workflow to appropriate computational resources. The restructuring of workflow is performed for performance optimization.

Workflow Engine: It executes the jobs as per dependencies defined by the workflow. It manages jobs by tracking their status. Job Scheduler is given the jobs whose parent jobs have been completed.

Local Queue and Job Scheduler: It manages workflow jobs and monitor the performance on remote and local resources. The Job Scheduler can be applied to different scheduling algorithms such as HEFT[120] and MinMin[121] and improve the overall runtime of workflow executions.

Job Wrapper: It takes tasks out of clustered jobs to be executed at the nodes of the worker. All of these components work cooperatively with each other to perform workflow preparation and execution.

Table 3.1: Symbols used in this work

Abbreviation	Definition
WF	Workflow
LVL	Level in a workflow
J	Number of jobs at a horizontal level
LT	List of tasks at a level
DP	Depth of workflow
JB_i	Empty Job
LC	Empty list of clustered jobs
LST	Sorted List of jobs
CT	Child Task
IF	Impact Factor
TLM	Merged Task List

3.4 Proposed Hybrid Balanced Task Clustering Algorithm

In this section, the proposed Hybrid Balanced Task Clustering Algorithm (HYB) is discussed which is not dependent on the input of the user. It is able to cluster the tasks vertically with single parent single child relationship and merge the tasks horizontally according to the number of resources. The system overhead is reduced while involving the best utilization of resources. The flowchart of the proposed technique is depicted in Figure 3.4. The symbols used in this work are explained in Table 3.1.

3.4.1 Problem Formulation

The two major issues that are undertaken by the clustering algorithms are dependency imbalance and runtime imbalance. Runtime imbalance refers to the unequal distribution

at the same horizontal level of the runtime of tasks. While dependency imbalance refers to clustering tasks at a level without taking into account the inter-task dependence factor.

This increases the waiting time for input at the next level and thus the delay in execution. The problem is also referred to as data locality. Generally, runtime imbalance leads to dependency imbalance and dependency imbalance leads to runtime imbalance. The structure of workflow is also an important factor while clustering the tasks. As horizontal clustering is always performed for tasks that may increase the problems of runtime imbalance and dependency imbalance. So instead of performing task clustering horizontally at each level of workflow, vertical clustering can also be performed of the tasks where single parent single child relationship exists between tasks besides performing clustering horizontally. This may lead to an improvement in the execution of workflow while further decreasing the delays. Considering the above challenges of clustering, the introduced method aims to obtain appropriate hybrid clustering while merging the tasks vertically with a similar impact factor and remaining tasks horizontally according to the number of available resources. Thus reducing the overall execution time. For the realization of the desired clustering, the proposed hybrid balancing algorithm as follows.

3.4.2 Research Methodology

This technique prefers to cluster the tasks with single parent single child relationship. The problem of dependency imbalance is catered by measuring the impact factor of tasks in the workflow. The Impact Factor (IF) of task t_n is denoted using Eq. 3.1.

$$IF(t_n) = \sum_{t_a \subset child(t_a)} \frac{IF(t_b)}{parent(t_b)} \tag{3.1}$$

where $child(t_a)$ refers to a set of child tasks t_a and $parent(t_b)$ is the number of parent tasks of t_b. Impact factor applied to determine the similarity of tasks or jobs [122].

The proposed algorithm ensures that the tasks with one parent-child relationship are clustered into one cluster. The remaining tasks are then clustered horizontally. The parent-child relationship is depicted by matching the impact factor of tasks vertically. The tasks with a similar impact factor are grouped into one cluster. This help in reducing the execution time of workflow and system overheads. For instance, Figure 3.5 shows a workflow of five levels consisting of one task at level one, two tasks at level two, four

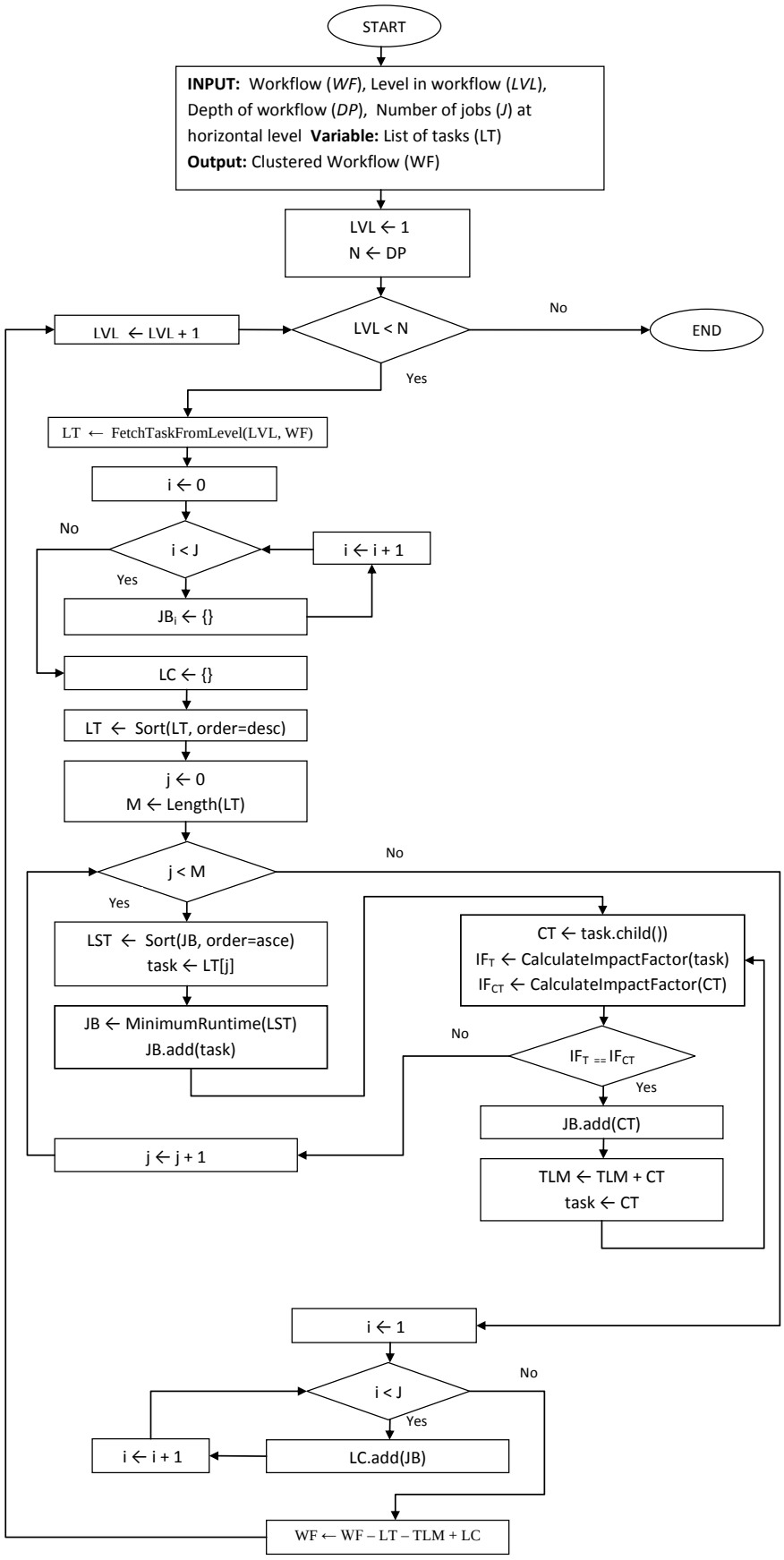


Figure 3.4: Flowchart of Proposed Hybrid Balanced Clustering Algorithm

tasks at level three, three tasks at level four and one task at level five.

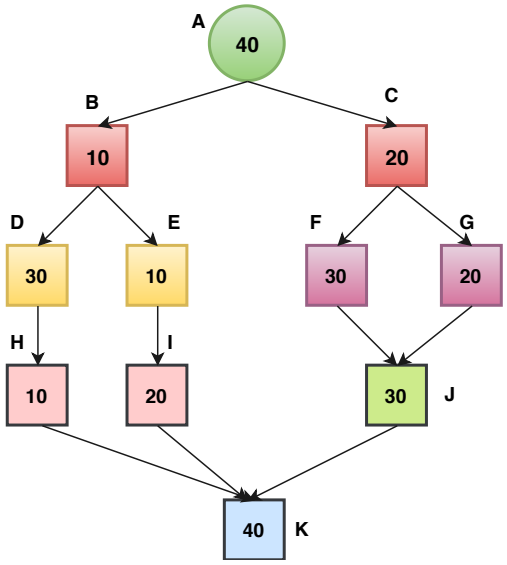


Figure 3.5: A sample workflow

Algorithm 1 presents the pseudo code of proposed Hybrid Balanced task clustering algorithm designed with the combination of horizontal distance balancing [32], horizontal runtime balancing [32], and task impact factor [32]. The number of jobs is input from the user. This algorithm addresses the problem of load imbalance and also considers the tasks with asymmetric structure. The algorithm begins with the first level of the workflow as in Figure 3.5 by selecting tasks from each level (Lines 2-3). Tasks are clustered into a job and returned by merge procedure (Line 4). The merge procedure merges the tasks vertically and horizontally. In merge procedure, the tasks are sorted according to decreasing order of runtime (Line 13). At the third level of workflow, there are four tasks D(30s), E(10s), F(30s), G(20s). The execution time for task D is 30s, similarly, for tasks E, F and G is 10s, 30s, 20s respectively. The task list LT is maintained as per decreasing order of runtime of tasks LT=D,F,G,E. Then horizontal runtime balancing adds the task D to the job with the shortest runtime as in Figure 3.6.

Step 2: The tasks are arranged as per the minimum distance with task D. The distance between the tasks is calculated as number of edges between the tasks. The distance of all nodes from node D is calculated as $Distance$=$0(D),4(F),4(G),2(E)$. The impact factor of task D included in a job is matched vertically to depict parent-child relationship exists or not. If the relationship exists, the child task is added to the job J1 where task D resides (Lines 20 - 22) as shown in Figure 3.7 and Figure 3.8.

Algorithm 1 Hybrid Balanced Task Clustering algorithm.

Require: WF: workflow; J: jobs per horizontal level ; LVL: Level of workflow; DP: depth of workflow

1: **procedure** HYCLUSTERING(Wf)
2: **for** $LVL < DP(WF)$ **do**
3: $LT \leftarrow$ FETCHTASKSFROMLEVEL(WF, LVL) ▷ Divide WF on the basis of depth
4: TLM,$LC \leftarrow$ MERGE(LT, J) ▷ Group of clustered job returned
5: $WF \leftarrow WF - LT + LC - TLM$ ▷ Dependency merging
6: **end for**
7: **end procedure**
8: **procedure** MERGE(LT, J)
9: **for** $i < J$ **do**
10: $JB_i \leftarrow \{\}$ ▷ NULL JOB
11: **end for**
12: $LC \leftarrow \{\}$ ▷ NULL JOB LIST
13: Sort LT in descending runtime order
14: **for all** *task* in LT **do**
15: $LST \leftarrow$ sort list of JB_i as per least distance with *task*
16: $JB \leftarrow$ the job with minimum runtime in LST
17: JB.add ($task$)
18: **while** $IF_T = IF_{CT}$ **do** ▷ CT is child task
19: JB.add(CT)
20: $TLM \leftarrow TLM + CT$
21: $task \leftarrow CT$
22: **end while**
23: **end for**
24: **for** $i < J$ **do**
25: LC.add(JB)
26: **end for**
27: **return** LC,TLM
28: **end procedure**

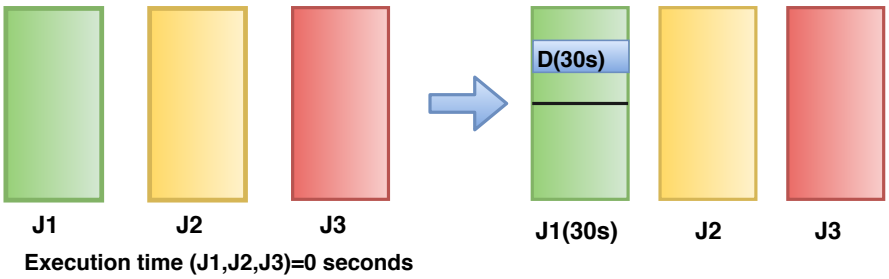


Figure 3.6: Merging of clusters in a job

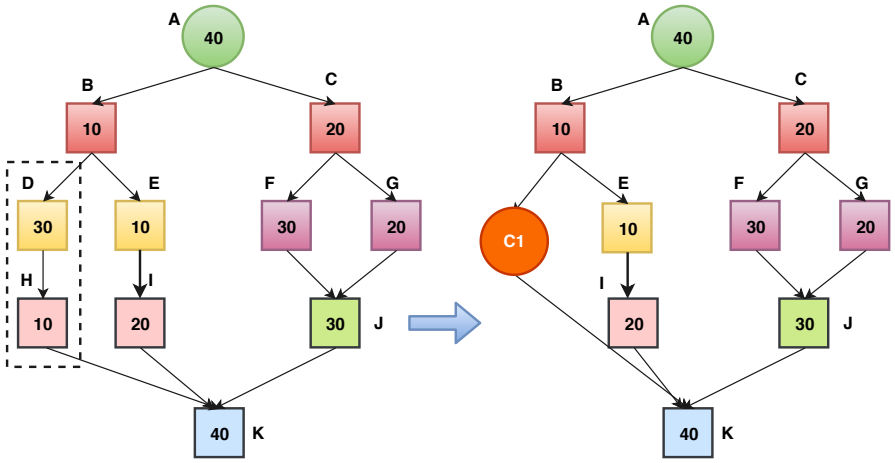


Figure 3.7: Clustering to avoid runtime imbalance and dependency imbalance

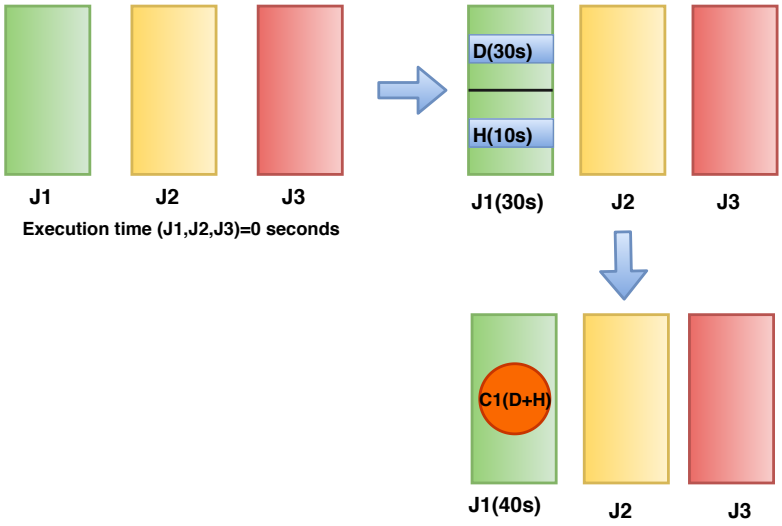


Figure 3.8: Merging clusters into a job

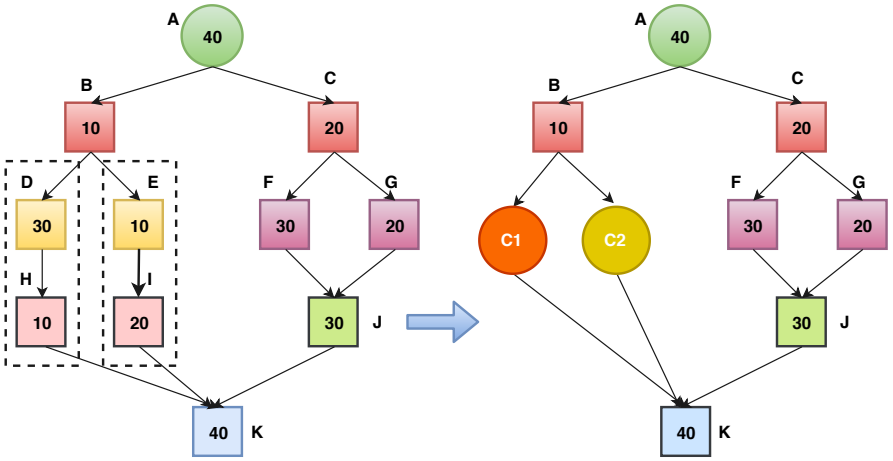


Figure 3.9: Clustering to avoid runtime imbalance and dependency imbalance

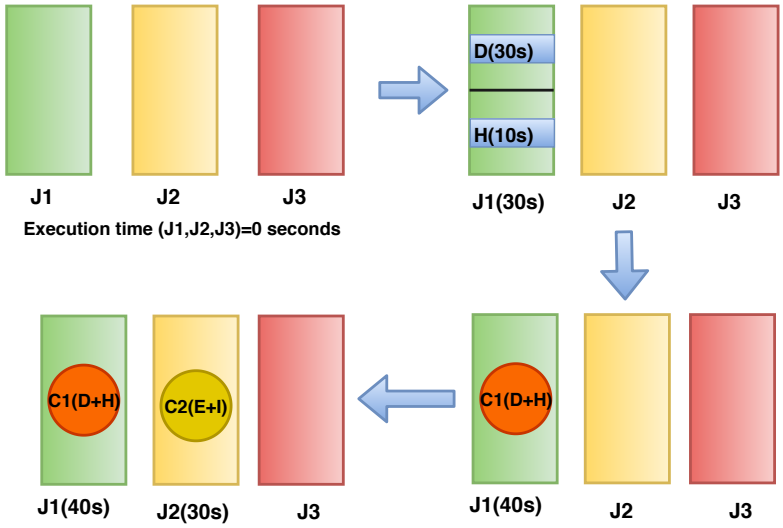


Figure 3.10: Merging clusters into a job

Step 3: Now after the clustering of tasks D and H. The tasks in a task list(LT) are task F, G, E. The horizontal distance balancing is performed on tasks E, F, and G on the basis of shortest distance between them and targeted task D. The distances are 4,4,2 respectively for tasks F, G, E. The candidate E with minimal distance 2 is selected. Now again the impact factor of task E is matched vertically and if the impact factor matches with the task vertically then both are clustered into another cluster c2 as shown in Figure 3.9 and merged as a job J2 as shown in Figure 3.10

Step 4: Now the tasks left in the task list LT are F and G. Similarly the process is repeated for the tasks F and G. However, these tasks do not have a parent-child relationship with any task. So they cannot be clustered vertically. Therefore, horizontal clustering is performed for the tasks F, G forming cluster c3 as shown in Figure 3.11

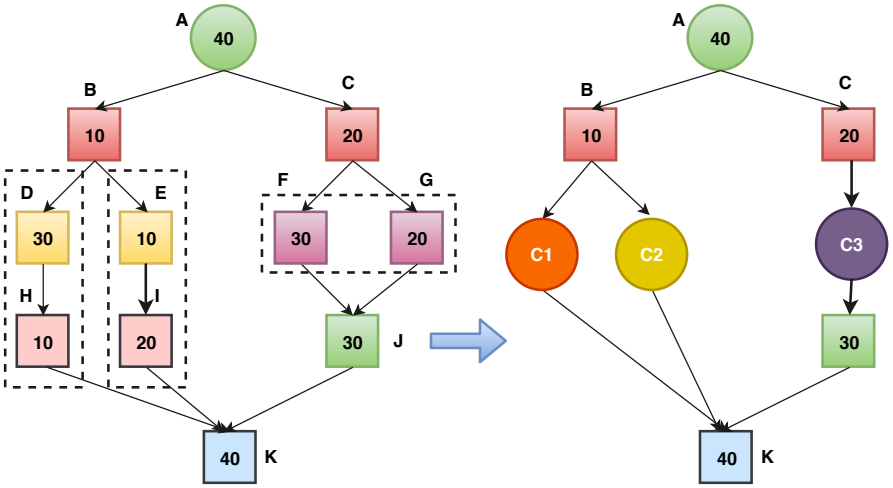


Figure 3.11: Clustering to avoid runtime imbalance and dependency imbalance

and merging into job J3 as shown in Figure 3.12. The steps are repeated for different number of tasks at levels of workflow. The tasks are merged into a cluster and assigned as a job to execute. This reduces the scheduling overhead. The clusters are formed at each level of workflow and then scheduled onto data centers.

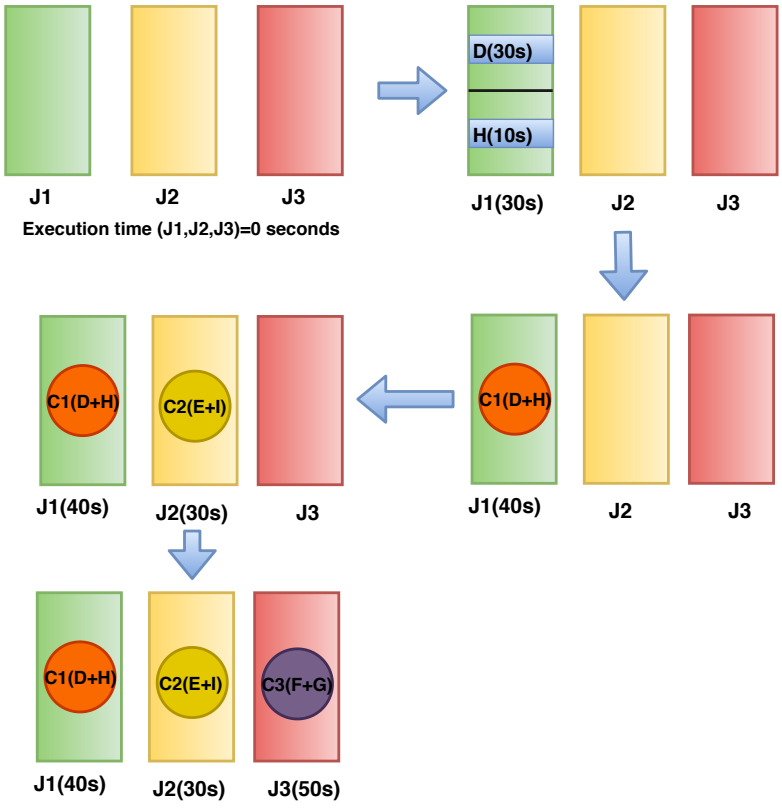


Figure 3.12: Merging clusters into a job

3.5 Experimental Evaluation

3.5.1 Scientific Workflow Applications

Montage [3] is one of the astronomy applications used to create large-scale image mosaics of the sky. The images as input are projected into a sphere and then the overlap is calculated for each image. These input images are projected to the precise orientation while preserving the constant background emission level for all images. In a final mosaic, reprojected images are finally added. The final resulting image describes the sky part under investigation in detail. The Montage workflow is represented by Figure 3.13. The size of the workflow depends on how many images are required to make the mosaic of the sky.

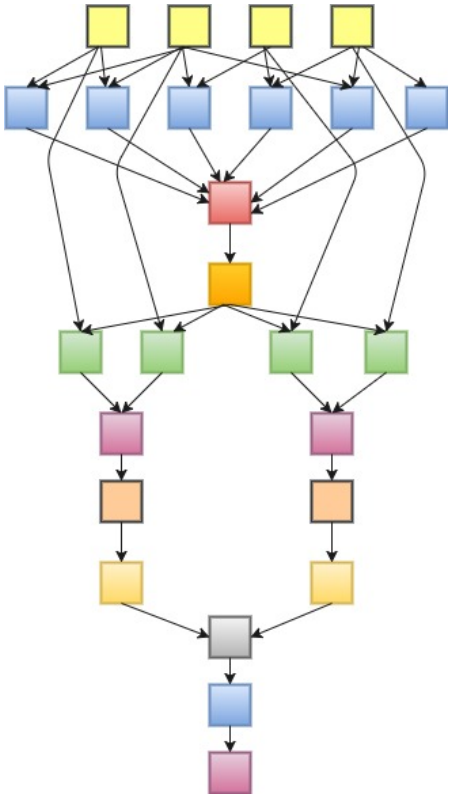


Figure 3.13: A simplified version of the Montage workflow [3].

Cybershake [4] is seismology application that is used for calculating Probabilistic Seismic Hazard curves for various geographic regions in Southern California. Hence, used in the identification of all ruptures within a 200KM radius. It also changes rupture into multiple rupture variations that differ in locations of hypocenters and distributions

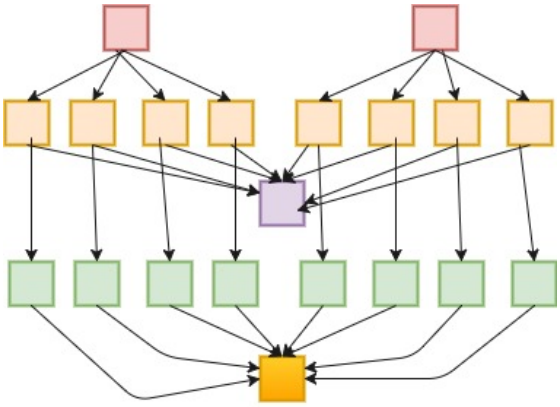


Figure 3.14: A synthetic version of the CyberShake workflow [4].

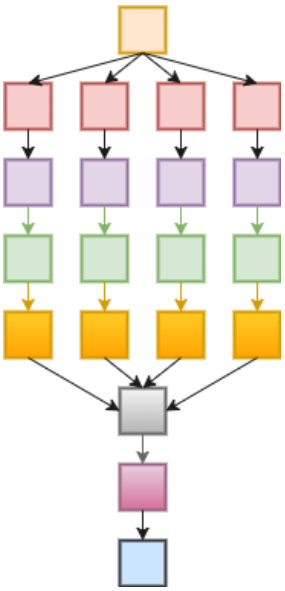


Figure 3.15: A synthetic version of Epigenomics workflow with multiple branches [5].

of slips. Then synthetic seismograms are calculated for rupture variance. After that, the peak intensity is extracted and added with the original rupture probability for production probabilistic hazards for the location. Figure 3.14 shows a Cybershake workflow illustration.

Epigenomics [5] is the CPU-intensive application. Initially, the data is obtained from the Illumina-Solexa Genetic Analyzer in the form of DNA sequence lanes. Multiple DNA sequences are generated by each Solexa machine. Then the workflow performs the mapping of DNA sequences. Hence, creating a map showing the sequence density. Figure 3.15 shows a simplified epigenomics structure.

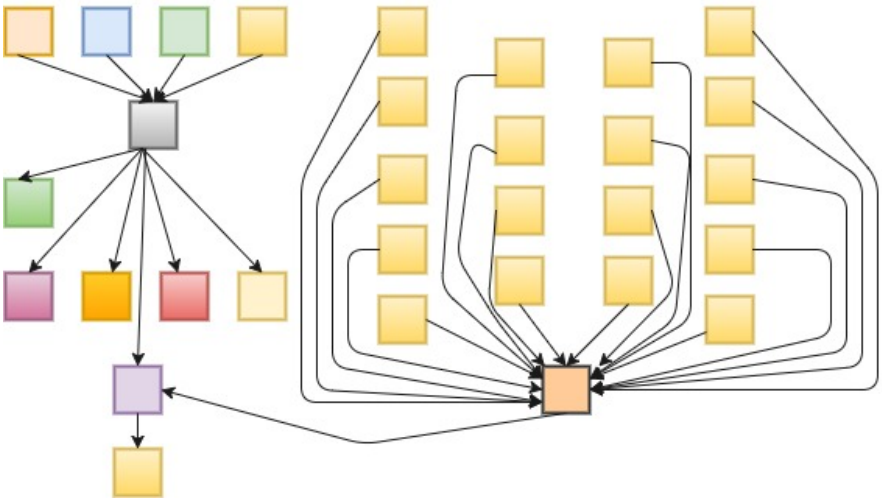


Figure 3.16: A simplified version of the SIPHT workflow [6].

SIPHT [6] is responsible for researching small untranslated RNAs (sRNAs). It is responsible for the regulation of different processes such as virulence or secretion in bacteria. This predicts ρ-independent transcriptional terminators. A simplified version of the SIPHT workflow is shown in Figure 3.16.

LIGO [7] Workflow data is collected by large-scale interferometers in the Laser Interferometer Gravitational Wave Observatory (LIGO) to search for gravitational wave signatures. The objective of the observers is to measure and detect waves as relativity predicts. It is a workflow that is data-intensive. A workflow version is shown in Figure 3.17. In this workflow tasks are divided into different groups, where there is a cluster of interconnected tasks as shown in Figure 3.17.

3.5.2 Balanced Task Clustering Algorithms

The techniques of task clustering are divided into two different categories of horizontal clustering and vertical clustering. A hybrid balanced task clustering method is introduced in order to overcome the limitations of these techniques.

3.5.3 Description of Baseline Balanced Clustering Algorithms

The clustering methods considered in order to study the impact of the proposed clustering technique are Horizontal Impact Factor Balancing (HIFB), Horizontal Distance Balancing (HDB) and Horizontal Runtime Balancing (HRB) [32].

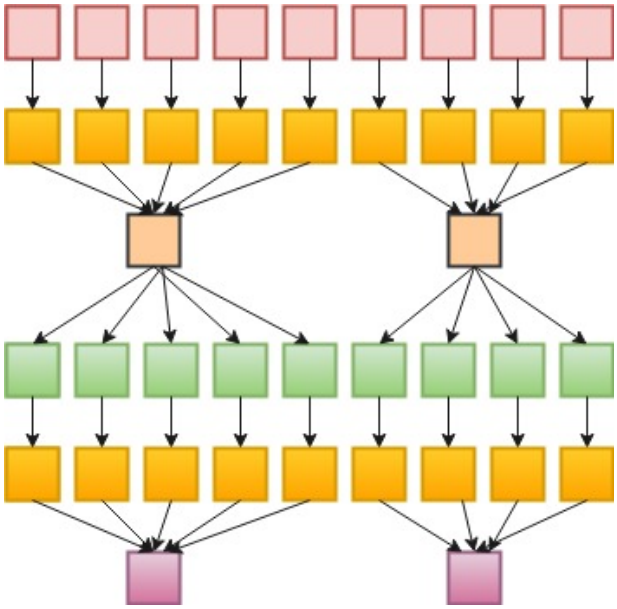


Figure 3.17: A simplified version of the LIGO Inspiral workflow [7].

Horizontal Clustering (HC): Tasks in this technique are merged into a job at the same horizontal level of workflow. The horizontal level for a single task is considered to be the greatest distance from the first workflow task beginning with the first task. The first task is a parent-free task.

As shown in Figure 3.18, the tasks t2, t3, and t4 are combined together into a cluster c1, thus creating just one job j1. The horizontal clustering is performed for the tasks at the same level. Thus reducing the overhead.

Vertical Clustering (VC): In this algorithm task at the same vertical level of a workflow are merged and make a single job. The tasks with one parent and one child relationship are merged.

As shown in Figure 3.19 the tasks t2, t5, t8 exhibit parent-child relationship as t2 is a parent of t5 and further t5 is parent t8. So these tasks are clustered together into a cluster c1, thus creating a job1. Similarly, t3, t6, t9 and t4, t7, t10 are clustered into clusters c2, c3 thus creating combined jobs job2, job3 respectively. The overheads are also combined into overheads o2, o3 and o4.

Horizontal Runtime Balancing (HRB): In this algorithm runtime of tasks is equally distributed between jobs. The problem of runtime variance is addressed at the same horizontal level. In this greedy method, the jobs are sorted according to the ascending order of runtime. The new task is joined to the job with a minimum runtime. This method

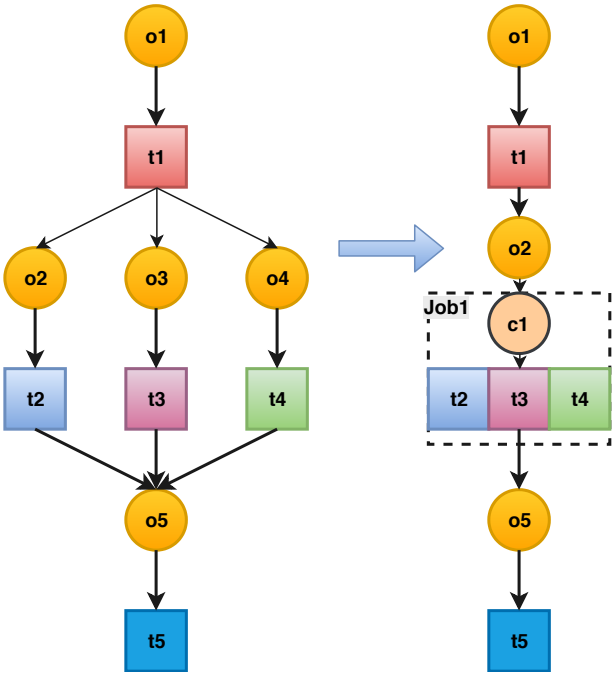


Figure 3.18: Working of Horizontal Clustering

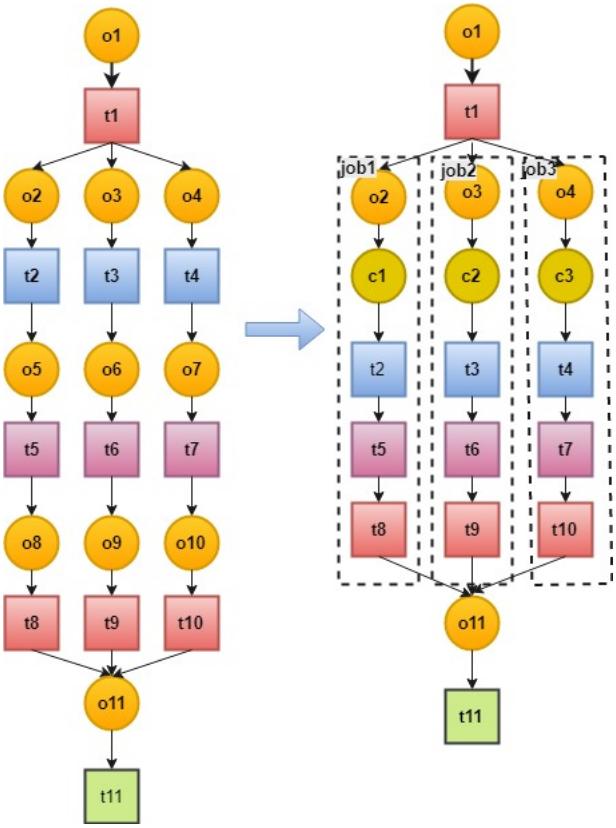


Figure 3.19: Working of Vertical clustering

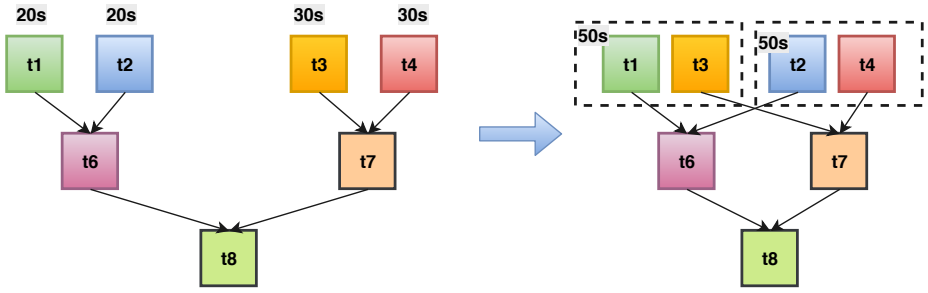


Figure 3.20: Working of the Horizontal Runtime Balancing (HRB) method

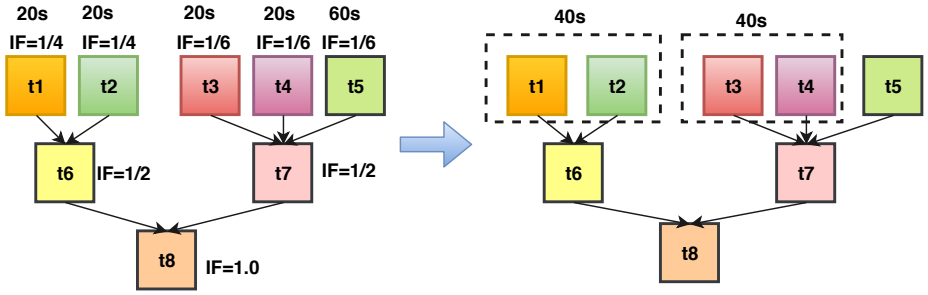


Figure 3.21: Working of Horizontal Impact Factor Balancing (HIFB) method

further raises the dependency imbalance among tasks as the factor of data dependency is not considered while clustering.

As shown in Figure 3.20, there are four tasks t1, t2, and t3, t4 with runtime 20s and 30s respectively. According to this clustering method tasks t1, t3 are combined into one cluster and t2, t4 are combined into another cluster. This balances the runtime among tasks but leads to dependency imbalance among tasks.

Horizontal Impact Factor Balancing (HIFB): It overcomes the limitation of Horizontal runtime balancing algorithm of dependency imbalance. In this algorithm, the jobs are first sorted by considering the similarity of impact factor (IF) in increasing order. Then the shortest job is selected using Horizontal Runtime Balancing. It groups the jobs sharing the same position in the workflow.

As shown in Figure 3.21 the tasks t1, t2, and t3, t4, t5 have same impact factor. Then using HRB the tasks t1, t2, and t3, t4 are combined into clusters c1, c2.

Horizontal Distance Balancing (HDB): Jobs are sorted in this clustering technique taking into account the distance from the target job. After that Horizontal Runtime Balancing method is executed for selecting the shortest job. The tasks with minimum distance are merged, thus reducing data transfer.

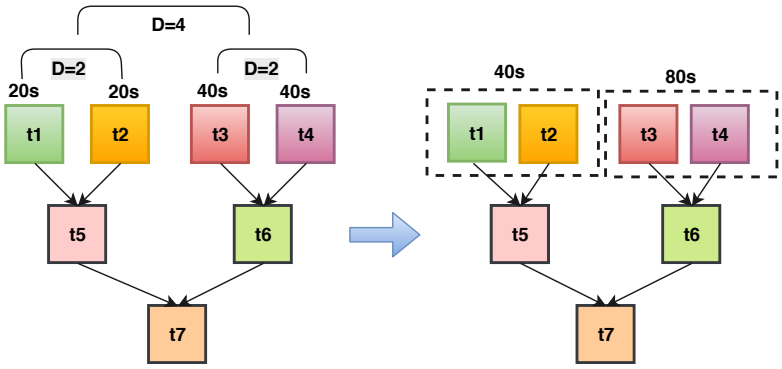


Figure 3.22: Working of Horizontal Distance Balancing (HDB) method

As in Figure 3.22 the tasks t1 and t2 are closest as compared to the distance between t1, t3, and t1, t4. So t1, t2 are combined into one cluster and t3, t4 are combined into another cluster.

The performance of the above discussed algorithms has been evaluated using workflowsim on various datasets e.g. Montage, LIGO, SIPHT, Epigenomics, and Cybershake.

3.5.4 Experimental Evaluation

Experimental Setup The trace-based simulation method is adopted for evaluation. Real traces are used for evaluation of algorithms. In the simulation environment, different system parameters used are a number of virtual machines, system overhead, different workflows and sizes of data.

For executing applications of workflow, open source workflow simulator workflowsim [123] is used. It is used for modeling an execution environment in the cloud. It is an extension of cloudsim. The DAG model of workflow is executed in it. It performs clustering and scheduling of tasks. It also performs provisioning of resources at the workflow level.

In the experiment, the simulator WorkflowSim is extended to implement the proposed hybrid balanced task clustering technique. The virtual machine cluster of 20 single homogeneous core VMs is used. In some distributed environments, such as Amazon EC2, this cluster is a quota for a user. Each VM has a 512 MB configuration. For each virtual machine, the processing capacity is 1000 MIPS and the network bandwidth has been set to 15 MB/s by default.

The configuration setup for the experiment is described as in Table 3.2

Table 3.2: Configuration setup for the experiment

Parameter	**Value**
No. of virtual machines	20
Memory Capacity	512 MB
Processing capacity	1000 MIPS
Network Bandwidth	15 MB/s

An evaluation of the proposed technique was performed for identification of its ability. The method was assessed for enhancing scientific workflow execution time and reducing the load to minimum resources. The experiment is conducted on the following variables.

- Makespan: Total time is taken to execute on the available resources by the workflow application.
- Performance Gain: The clustering algorithm used over the execution of the workflow without clustering a task is defined as an overall improvement in the execution time of the workflow. It can be evaluated with Eq. 3.2

$$\mu = (time\,taken\,without\,clustering - time\,taken\,with\,clustering) \\ /(time\,taken\,without\,clustering) * 100 \tag{3.2}$$

  $\mu > 0$ for a clustering technique signifies that it leads to improvement of the execution time of a workflow. $\mu < 0$ refers to the negative impact of the clustering method. This negative impact leads to an increase in execution time of workflow.

Different scientific workflow applications are used in the experiments along with the fixed number of tasks for each as shown in Table 3.3.

3.5.5 Results and Discussion

Table 3.3 depicts the scientific workflow applications used for the experiment and the number of tasks considered for each.

Table 3.3: Scientific workflow for experiment and number of tasks

Workflow	**Number of tasks**
Cybershake	1000
Epigenomics	997
Montage	1000
SIPHT	1000
LIGO	1000

Then the makespan time for each scientific application is calculated as described in Table 3.4.

Table 3.4: Makespan time of workflow with or without clustering algorithms

	HC	**HRB**	**HIFB**	**HDB**	**HYB**	**without clustering**
Cybershake	1511.46	1303.58	1833.68	1445.07	1352.92	2222.05
Epigenomics	238974.2	218583.8	241833.1	238754	206077.6	253462
LIGO	13188.79	12768.49	15261.21	13015.6	11635.49	17234.23
Montage	1210	924.71	936	947	923.61	1927.69
SIPHT	21154.99	9537.03	22778.28	18539.23	9499.07	23453.58

Using the makespan time, performance gain of scientific applications Cybershake, Epigenomics, LIGO, Montage and SIPHT is calculated and compared as evaluated according to different baseline algorithms and proposed an algorithm in Table 3.5.

Table 3.5: Performance gain for various workflows

	Cybershake	**Epigenomics**	**LIGO**	**Montage**	**SIPHT**
HC	31.97903	5.715961	23.47329	37.23057	9.800593
HRB	39.11388	13.76072	25.91204	52.03015	59.33649
HIFB	17.478	4.588013	11.44826	51.44447	2.879305
HDB	34.96681	5.802842	24.4782	50.87384	20.95352
HYB	41.33435	18.69489	32.48616	52.08721	59.49842

Three sets of experiments are conducted on the scientific workflow applications.

Experiment 1 (Improvement in execution time): The makespan time of balanced clustering algorithms on different scientific workflow applications Cybershake, Epigenomics, LIGO, Montage, SIPHT is obtained from the experiments performed. The makespan time is mentioned in Table 3.4. According to the experimental evaluations the following results are depicted:

- **Cybershake**: In this application, Hybrid Balanced (HYB) task clustering algorithm improves the performance by 26% and 10% respectively from Horizontal Impact Factor Balancing (HIFB) and Horizontal Clustering (HC).

- **Epigenomics**: In this application, there is 14.78%, 13.68%, 13.76% improvement in execution time by Hybrid Balanced Clustering algorithm then Horizontal Impact Factor Balancing (HIFB), Horizontal Distance Balancing (HDB) and Horizontal Clustering (HC) respectively.

- **LIGO**: In this application, there is 23.75%, 11.77% , 10.60% and 8.87% improvement in execution time by Hybrid Balanced Clustering algorithm then Horizontal Impact Factor Balancing (HIFB), Horizontal Clustering (HC), Horizontal Distance balancing (HDB) and Horizontal Runtime Balancing (HRB) respectively.

- **Montage**: In this application, there is 23%, 2.46% improvement in execution time of workflow by Hybrid Balanced Clustering algorithm then Horizontal Clustering (HC) and Horizontal Distance Balancing respectively.

- **SIPHT**: In this application, there is 58%, 55% and 48% improvement in execution time by Hybrid Balanced Clustering algorithm then Horizontal Impact Factor Balancing (HIFB), Horizontal Clustering (HC) and Horizontal Distance Balancing (HDB) respectively.

Hence from experiment 1, it is concluded that the execution time taken by Hybrid Balanced (HYB) task clustering algorithm is lesser as compared to other baseline algorithms. Proposed technique shows the overall improvement in execution time.

Experiment 2 (Performance Gain): In this experiment, the performance gain (μ) of the proposed technique is evaluated with the other clustering algorithms. In this experiment, the proposed technique is evaluated for identification of the amount to

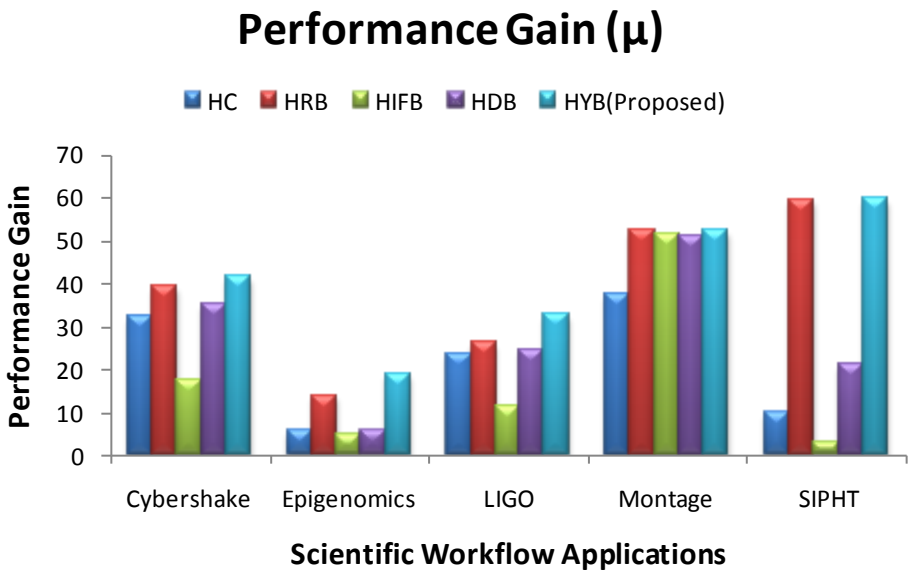


Figure 3.23: Comparison of performance Gain (μ) for various clustering methods

which it impacts the overall execution time of workflow. Figure 3.5 shows the performance gain μ for different clustering methods obtained from the experimental evaluation. From the experiments, it is depicted that all the methods of clustering retain a positive gain performance. The proposed technique further improves the execution time of different workflow applications. According to obtained results Montage, CyberShake and SIPHT workflows have better performance gains with a minimum gain of 52%, 41%, and 59% respectively. In comparison LIGO and Epigenomics having a minimum performance gain of 32% and 18% respectively. The performance gain variation is due to the granularity of the average workflow runtime of tasks. Considering the workflows the lowest value for the performance gain is for HIFB in SIPHT workflow. In all the scientific workflows, the proposed Hybrid Balanced (HYB) task clustering method performs better than the other balancing techniques. The clustering method Horizontal Distance Balancing (HDB) and Horizontal Impact Factor Balancing (HIFB) lacks in performance, because of the runtime as well as dependency imbalance between tasks. These methods groups the tasks into a cluster that should execute in parallel leading to an increase in execution time of workflow. The proposed hybrid balanced task clustering method delivers a good performance since it first checks for parent-child relationship and then clusters the tasks. Therefore, increasing the performance gain and decreasing the overall execution time of workflow as shown in Figure 3.23.

Experiment 3 (Varying Virtual Machines): In experiment 3, the number of virtual machines is varied while keeping the total number of tasks, each task consider as fixed 1000 MIPS.

First, the experiment is performed by using a horizontal clustering technique while varying the number of Virtual Machines (VMs) as shown in Figure 3.24

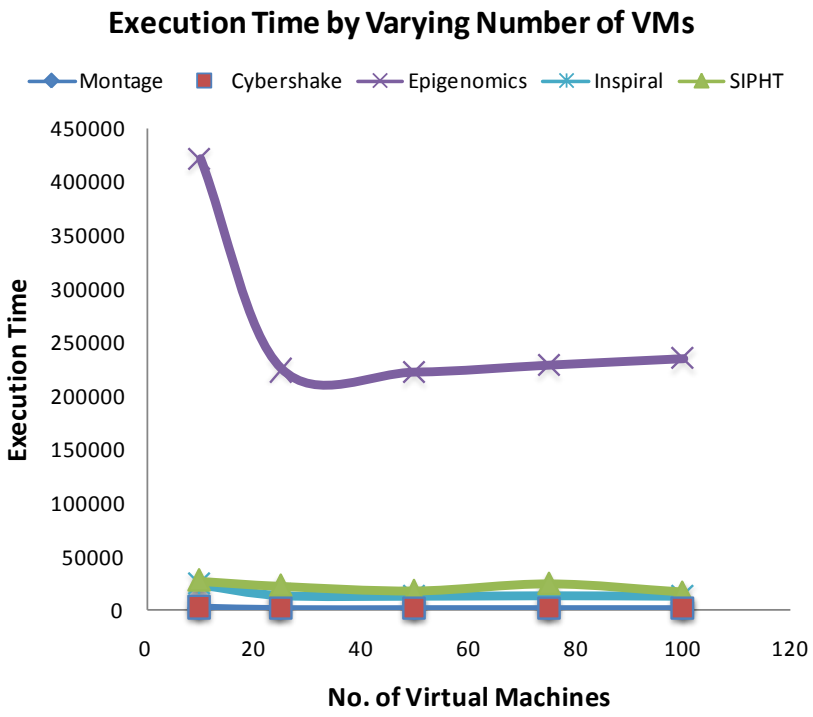


Figure 3.24: Execution time while varying virtual machines in Horizontal Clustering technique

The same experiment is conducted by using a hybrid balanced clustering algorithm and the result is depicted as shown in Figure 3.25.

While comparing both the results, it is found that horizontal clustering technique performs in a similar manner inspite of more number virtual machines available. While in the proposed algorithm, the execution time decreases as the number of virtual machines increases. Hence leading to an increase in performance of the execution of workflow applications. In some situations, the proposed technique performs as similar to baseline algorithms. This depicts the proposed technique also reaches a stagnation point but after more of the resource utilization in comparison to baseline algorithms.

As in Figure 3.24 the graph depicts the number of virtual machines increases rel-

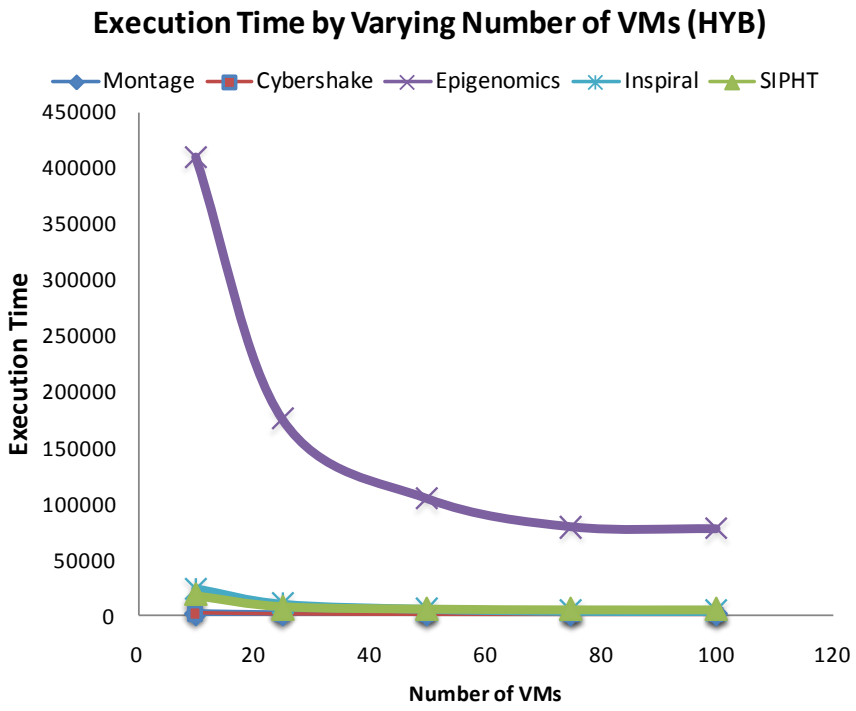


Figure 3.25: Execution time while varying virtual machines in Hybrid Balanced Clustering technique

atively decrease the execution time upto certain level after that it does not matter on adding more virtual machines. The reason is that the parallelization in the workflow depends on structure of scientific workflow. After certain level it does not matter on adding more virtual machines. The proposed technique is performing in a similar manner and adding up the virtual machines does not decrease the execution time.

3.6 Summary

The proposed Hybrid Balanced (HYB) task clustering method reduces the execution time of workflow execution and avoids wastage of resources. The proposed approach considers distance variance and merges the tasks with similar impact factor in the pipeline. A simulation experiment is carried out to evaluate the proposed algorithm compared to four methods of clustering: Horizontal Runtime Balancing (HRB), Horizontal Clustering (HC), Horizontal Distance Balancing (HDB), and Horizontal Impact Factor Balancing (HIFB). The results show that the proposed method of clustering can perform significantly better than the existing clustering methods for scientific workflow.

CHAPTER 4

A DATA PLACEMENT STRATEGY BASED ON CROW SEARCH ALGORITHM IN CLOUD COMPUTING

4.1 Introduction

In big data era, the increasing amount of data in scientific workflows arises the need for high-end computing and a massive amount of storage. The most complex task is to process this data. For a scientific workflow, relatively large data is to be stored in different data centers [124]. Data centers are facing a variable amount of visitors with the generation of data with the development of the Internet. This leads to an explosive increase in storage capacity requirement[125]. A large amount of scientific data is accumulated by various fields of research such as meteorology, astronomy, and bio-informatics. Processing of large scale data and obtaining valuable scientific discoveries are complex jobs. This lead to a rising need for high-performance computing [126]. This high-performance computing, also known as cloud computing entails the need for mass storage resources [127].

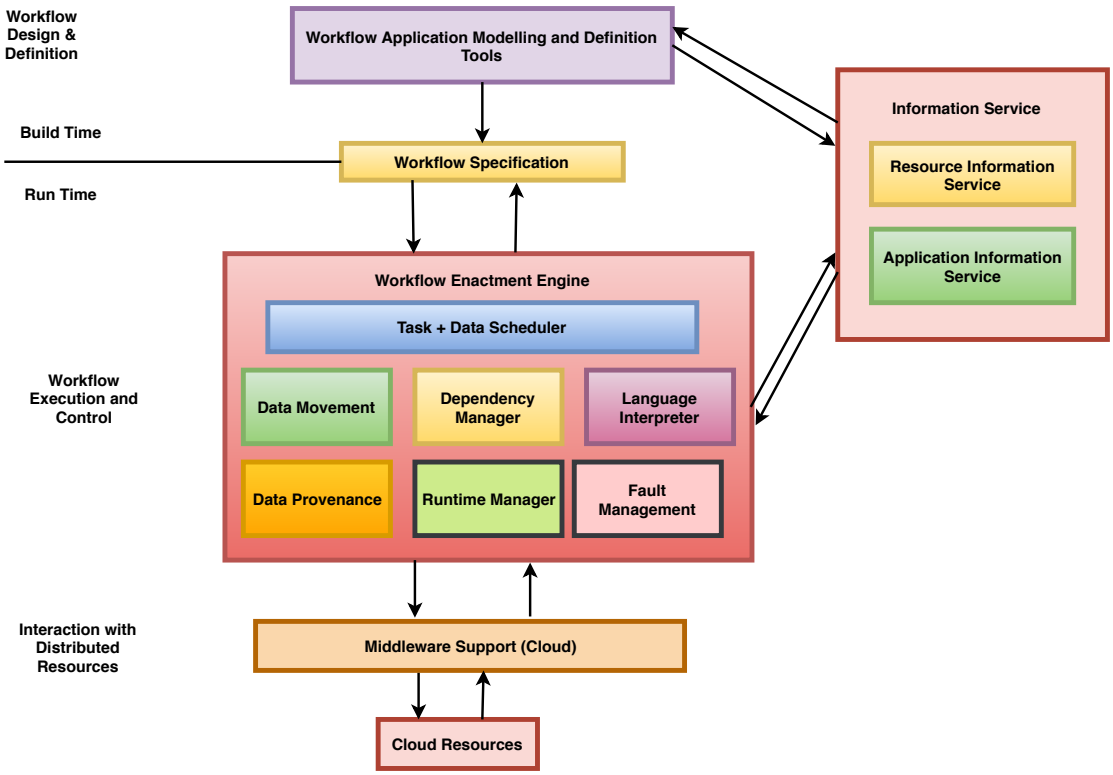


Figure 4.1: Cloud architecture for Workflow management system

The main process in cloud computing is to meet large volumes of data and store data at a suitable data center during the execution of the workflow. This case arises the need of data placement strategy [128]. Data placement includes all activities related to movements such as replication, staging, transfer, space de-allocation, space allocation, unregistering and registering metadata, retrieving and locating the data. There arises a question of how data placement is performed on cloud computing.

The clients work on data, evaluate it and pass on to the server. The request of the client is processed by the server. This is to-and-fro operation and, at the end, final output is to be stored on the storage device. The data placement strategy is responsible to determine this storage of output on a storage device.

Data storage reaches a terabyte magnitude scale in the cloud computing era. The diverse and complex data structures, high demand for type and level of service has induced great pressure on data management [129]. Cloud systems work on two kinds of applications data-intensive and compute-intensive. These are handled concurrently in the systems and generates a massive amount of output/intermediate data. The cloud architecture that manages data and workflows is as in Figure 4.1.

The designing in engineering is a decision making the process for building the products satisfying particular needs[130]. Mostly, design problems in engineering include complex objective functions with the varying number of decision variables [131]. The method of optimization finds an appropriate solution considering the decision variables. Different methods are used to solve design problems in engineering. The techniques offer valid solutions but fail when it comes to complex design issues. There may be a number of decision variables that can affect the objective function in some real design problems. There can be many local optima by the objective function, but the researcher may be interested in global optimum. So in these situations, appropriate optimization methods are required.

Promising performance is shown by metaheuristic algorithms for solving problems of real-world optimization. These metaheuristic algorithms use a specific tradeoff of local search and randomization [132]. These algorithms do not guarantee to find an optimized solution. These algorithms may not work most of the times. The best suitable for global optimization is metaheuristic algorithms [133].

In the latest trend, mostly nature-inspired metaheuristic algorithms are used for tackling complicated problems [132, 133]. These algorithms are surprisingly very efficient. Some of the famous metaheuristics algorithms are Harmony Search (HS) based on music improvisation process [134], Genetic Algorithm (GA) based on natural selection [135], Particle Swarm Optimization (PSO) based on the social behavior of fish schooling and bird flocking [136], a cuckoo search algorithm based on the brood parasitism of some cuckoo species [137, 138], Firefly Algorithm (FA) based on the flashing light patterns of tropic fireflies [139], Bat Algorithm (BA) based on echolocation behavior of microbats [140], etc. In the recent era, only a few characteristics inspired by nature are used, and more algorithms can be developed. One of the motivations of this research is to propose a metaheuristic method inspired by a user-friendly nature to place data sets in suitable data centers to achieve promising results.

Data placement technique has been designed on the basis of nature-inspired metaheuristic technique, Crow Search Algorithm (CSA). The characteristics of cloud data centers are investigated. The proposed technique integrates initial/generated and fixed location datasets and divides into two stages: stage-in and stage-out. For the heuristic data placement, CSA is used for reducing the time complexity of scientific workflows.

The number of experiments are conducted by varying different parameters such as a ratio of shared and fixed location data sets, number of tasks and size of datasets.

Compared to the traditional data placement techniques for scientific workflows, the experimental results depict that our approach reduces the data transfer cost. The primary contributions of this chapter are:

- A Crow Search Algorithm (CSA) based data placement algorithm is designed for distributing datasets according to the levels of the workflow. The algorithm persists lower complexity and finds the appropriate data center for placing unshared and shared datasets flexibly.

- Proposed algorithm consists of both build-time and run-time stages. All initial data sets are allocated to data centers at the build-time stage by bond energy algorithm, then all intermediate data sets are distributed in the run-time stage by the proposed method.

The remaining chapter is organized as follows. In Section 2, background for data placement and stages of data placement is represented. It also defines the different categories of data placement. Section 3 defines the used techniques for data placement along with its parameters. Section 4 describes a novel proposed data placement algorithm. Section 5 explains the experimental results. Finally, in section 6 presents a summary of this chapter.

4.2 Related Work

The most critical issue in scientific workflows is the placement of intermediate generated datasets. Scientific workflow applications require a large number of datasets from various organizations in the cloud computing environment, and these datasets need to be distributed across different data centers. The scientific applications are executed in hybrid cloud [141]. Two or more then two public, private or community clouds composes hybrid cloud. A popular Amazon Web Services (AWS) cloud environment charges large amounts per gigabyte for data transfer from and into AWS. The data centers belonging to various cloud providers will, therefore, include the cost of moving data between these data centers during the execution of the workflow[141]. The massive data movements for processing of scientific workflow bear a high cost.

Yuan et al. [40] proposed a data placement algorithm for a run time as well as for build time stage. In the initial stage dependency between all the data sets is calculated and dependency matrix is built. The partitioning and clustering of data sets are performed by Bond energy algorithm (BEA). These partitions are shared among different data centers. The introduced algorithm deals with the intermediate generated datasets. The dependencies for each data center are judged, and then accordingly data is moved. The factors of data movement and gathering of data at one point are covered up. Er-dun et al. [41] proposed an algorithm for reduction of movement of data among data centers leading to load balancing in data centers. The heuristic and genetic schemes are combined together for improving the ability to local search and reducing the search time. The heuristic idea is implemented in gene operations and initial population selection. The integer encoding rules are applied and a gene represents the placement process. The encoding rule determines the ineffective fragments of genes that cannot be coded at data centers. Fitness function depicts the advantages and disadvantages of genetic individuals. It shows the degree of dependence between data sets and center load balance. Xu et al. proposed a mathematical data placement technique on the basis of a genetic algorithm. The crossover rate(P_c), size of population (G) and mutation rate (P_m) is determined [42]. It is determined after the initial population generation and the fitness value of each individual [44]. Crossover and mutation are performed on the selected matrix [45, 142]. The individuals not adhering to the requirements of storage capacity are abandoned. In addition, Zheng et al. [43] introduced a genetic algorithm-based data placement technique to reduce time and data scheduling costs between centers. Guo et al. suggested a scheme based on particle swarm optimization leading for cost reduction and transfer time reduction [57]. The investigator works on the on-demand method as a standard for computing. This refers to paying per on per hour basis with no long term commitments as per the charging standard of Amazon. Guo et al. [59] introduced Particle swarm optimization based algorithm embed in Small Position value (SPV) [60]. The aim is to optimize the scheduling of tasks in cloud for minimization of cost of processing. The focus is on reducing execution and transferring time. The fitness function for an algorithm is taken as a combination of cost and time $CT = T + C$ [57].

Pandey et al. [61] presented heuristic based particle swarm optimization (PSO) to schedule applications on resources of cloud. Consideration is given to the factors of cost

of data transmission and computation. Mapping tasks to resources are accomplished by varying the cost of computing and communication.

Zhao et al. [35] clustered datasets on the basis of dependency using a proposed hierarchical clustering based technique. A new factor as the size of the dataset is introduced. This leads to a reduction in data movement. In the partitioning matrix, improved dichotomy algorithm is used. For the mapping between data groups and servers, PSO based algorithm is used.

Promising performance is shown by metaheuristic algorithms for solving problems of real-world optimization. These algorithms use a tradeoff between local search and randomization [132] and do not guarantee of optimization solution to be found. These algorithms may succeed most of the time but not every time. The best suitable for global optimization is metaheuristic algorithms [133]. At present, very limited nature-inspired algorithms are used and even more algorithms can be proposed. One of the main purposes of the research work is to introduce a user-friendly metaheuristic technique for placement of data sets at appropriate data centers for obtaining promising results. In this work, a crow search algorithm (CSA) based data placement strategy is developed with an aim to achieve a lesser cost of transmission among data centers.

4.3 Data Placement Process

In the computing systems, the data placement process is defined as the movement of input data of data-intensive applications from a remote site to the execution site, then a movement of output data from execution to another remote site or same site. To prevent the risk of disk full at the execution site, there is a requirement to allocate space and deallocating the space when a job is done.

The data placement steps are presented as DAG (Directed Acyclic Graphs) and dependencies between them are represented by an arc. The steps of data placement are as shown in Figure 4.2.

4.3.1 Data Placement Stages

- Stage-in: This data placement stage is also known as build-time data placement. This involves getting the provenance information of the dataset and pre-clustering the similar data items before uploading the data to the appropriate data center [8].

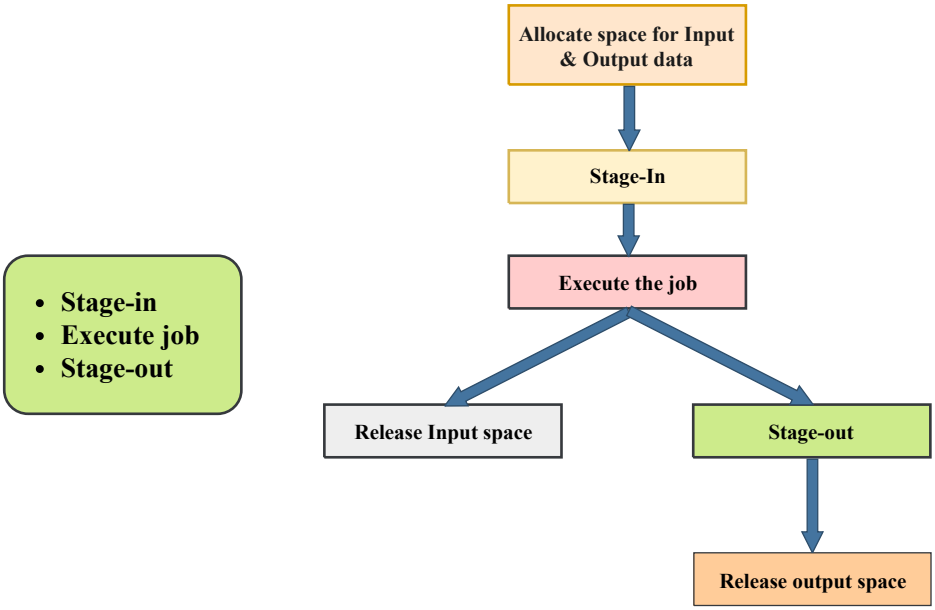


Figure 4.2: Data Placement Process [8].

- Stage-out: This data placement stage is also known as runtime data placement. During the execution of workflows, intermediate data is generated. This data may be the input for the subsequent tasks, so placing and managing this kind of data is again a complicated task [8].

4.3.2 Need for Data Placement

- Management of budget for storage: Some scientists perform their task without considering the budget whereas there are some that consider budget as one of the QoS parameters. So, it depends upon vendor to vendor. These kinds of vendors prefer the cloud service providers that adhere to the QoS parameters provided by them [75][57].

- Effective distribution of data onto a storage device: There is a large variety of storage devices available with different parameters like storage capacity, transmission speed, etc. So making the data evenly and considerably distributed among these storage devices is also an essential task [128].

- Data placement of data-intensive jobs is a major challenge in the cloud environment. Improper data placement policy increases data movement in the data center which leads to increased cost and delayed services.

- Placement of runtime datasets is still a challenge in a complex scientific workflow. Storage and computation capacity is a bottleneck in some situations while transferring the intermediate datasets.

- No placement policy considers the cached dataset (e.g. fixed position datasets) which affects the performance at a significant level.

With the drastic increase in data, requirements for both commercial and scientific applications are expected to reach several million terabytes scales. The matter of concern is about the increasing I/O needs and also about a number of users accessing the same dataset. In various fields like genomics and biomedical, more people will be accessing more datasets. The movement of a large amount of data for replication and processing becomes the necessity. This brings a problem of reliable and efficient data placement. There is a requirement of locating the data, moving the data to the place where it is required, replicating and staging the data. The storage is allocated and deallocated. As the scheduling and managing of computational and network resources is an important task, similarly scheduling the data placement activities is also crucial.

The above requirements of scientific workflows can be fulfilled by a cloud computing paradigm. Firstly, cloud computing environment consists of large scale commodity hardware which provides infinite storage space with lower execution cost. Secondly, it is easily accessible from anywhere via the internet. Researchers can carry out research from any region of the world. Thirdly, the services provided by the cloud uses a pay-as-you-go model, depending on the requirement of the resources of specific scientific applications. So, deploying scientific workflows on a cloud platform is the most cost-effective. Several successful scientific cloud workflow systems are Cumulus (https://cumulusnetworks.com), Nimbus (http://www.nimbusproject.org/ doc/ nimbus/ platform), and OpenNebula (http://opennebula.org).

Although, the cloud provides different features to execute the workflows in cloud with infrastructure as a service several challenges are faced by the scientific cloud workflow system. Executing and scheduling the tasks of scientific workflows is a tedious task. During scheduling, intermediate datasets are generated and placement of these datasets is the most critical issue in data placement.

4.4 Research Methodology

In the present study, the real-world workflow graphs or random workflow graphs are employed to represent different applications of the workflow using SwinDeW-C framework [143]. Directed Acyclic Graph (DAG) of different workflow applications is represented by this framework. This framework specifies the number of tasks in the workflow, the size of transfer between tasks of workflow, and the execution time of each task. These workflows have been used to measure the performance of the proposed CSA-based data placement algorithm.

In this chapter, on the basis of intelligent behavior of crow, Crow Search Algorithm (CSA) based data placement for scientific workflows in cloud computing is proposed. The introduced algorithm is based on the crow's behavior. Crow possesses the largest brain in comparison to their body size and are considered to be the most intelligent of all birds. They have remembrance power and can remember the places of hiding food up to several months later [144, 145]. The protocols of CSA are [146]:

- Crows live in congregates.
- Crows remember their hiding places positions.
- Crows follow one another to do rustling.
- Crows protect their hiding places of food from being stolen.

4.4.1 Proposed Crow Search Algorithm (CSA) Based Data Placement

The researcher presents a novel CSA-based data placement strategy for scientific workflows in cloud. The proposed algorithm is responsible for the placement of intermediate datasets at the data centers while ensuring minimal data movement between data centers. The two stages in the proposed technique are to build time data placement and runtime data placement. Firstly, the build time algorithm is used to distribute data sets to suitable data centers. After the build time stage, intermediate data sets are generated. Secondly, a runtime algorithm is used to distribute the intermediate datasets to data centers. The runtime stage uses the Crow Search Algorithm (CSA) based data placement method. The flowchart of CSA based runtime data placement is shown in Figure 4.3

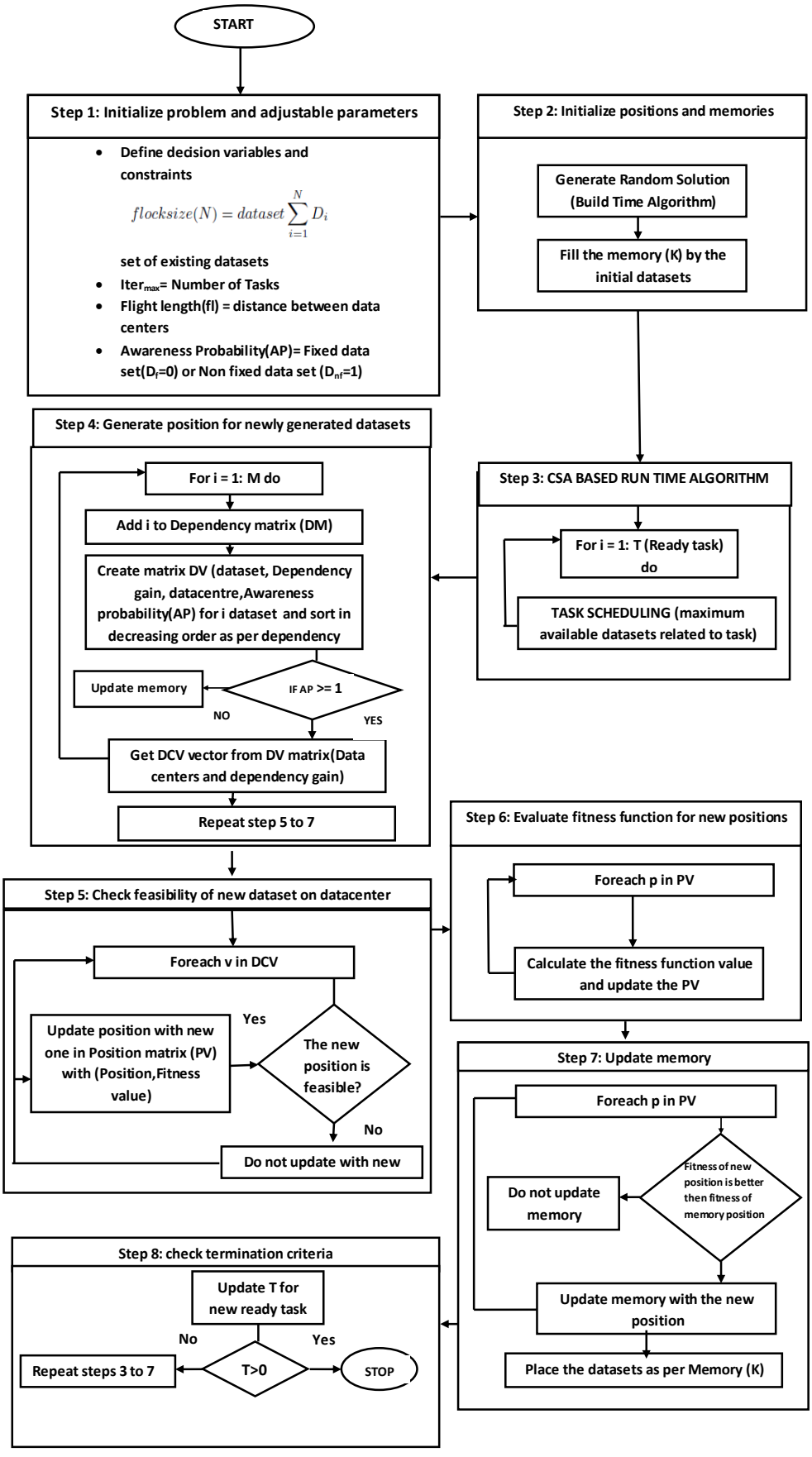


Figure 4.3: Flowchart of Crow Search Algorithm(CSA) based data placement.

4.4.2 Pseudo Code of Proposed CSA-based Runtime Data Placement

- **Step 1:** Initialize problem and parameters adjustable: Variables of decision and constraints are defined in this step.
  - $crows(cr) = number\,of\,datasets$
  - The flock size represents the number of existing datasets as per Eq. 4.1.

$$flocksize(N) = dataset \sum_{i=1}^{N} D_i \tag{4.1}$$

  - $iter_{max}$ represents the number of tasks in the workflow.
  - Flight Length (FL) of a crow is defined as the distance between data centers.
  - Awareness Probability (AP) is defined as awareness about fixed and non-fixed datasets. For fixed location data sets $D_f = 0$ or Non-fixed data sets $D_{nf} = 1$.

- **Step 2:** Initialize crow(datasets) position and memory, data for clustering algorithm allocation. In a search space, N crows are randomly positioned. Build time algorithm is performed in this step. In build time stage, the initial datasets are randomly positioned to datacenters. The data is represented by the matrix model. Pre-clustering of the data sets is performed by K-means clustering algorithm. The matrix is transformed and data sets are allocated to different data centers as per the initial partitions defined for use in the runtime stage. This algorithm works in two steps: 1) Setting up and clustering dependency matrix 2) Partitioning and distributing datasets [147] to an appropriate data center. The memory of the crows(dataset) is initialized with the data center as shown in Figure 4.4. The tasks are scheduled according to the placement of initial datasets and intermediate data sets are generated. Scheduling is an important issue for data-intensive applications [148, 149]. The workflow tasks are periodically monitored and ready tasks are scheduled onto the data centers

Dataset	d1	d2	d3	d4	d5	d6	d7	d8
Datacenter	dc2	dc2	dc2	dc1	dc1	dc2	dc2	dc2

Figure 4.4: Data Placement Map of initial datasets.

- **Step 3:** The runtime stage is performed in this step. The runtime stage is responsible for placing the data sets at an appropriate data center according to CSA based data placement algorithm. The ready tasks are selected one by one with all the available datasets required by it. The ready task is a task if all the datasets required by it are available for its execution. Now, it is decided by the system which data center to allocate to these datasets. In our work, the focus is to place intermediate generated datasets by CSA based approach during the runtime stage of execution. For example, task $t1$ is the ready task with datasets related to as d1, d2, d3, d4.

- **Step 4:** In this, the position for newly generated datasets is defined. The newly generated datasets for a selected task t1 is added to the dependency matrix. Dependency Gain (dg) is calculated from the dependency matrix for each of the datasets d1, d2, d3, d4 related to the task t1. The dependency between dataset d_j and datacenter dc_i is defined as the sum of dependencies of d_j with all the datasets in the data center dc_i.

  Suppose d_v is newly generated dataset. t_v is a set of tasks that will use d_v. Firstly, the dependencies of newly generated dataset d_v are calculated with all the other available datasets in the system. This newly generated dataset is added to the dependency matrix(DM) for d_v as represented in Eq. 4.2

$$DM_{vj} = DM_{jv} = Dependency_{vj} = Count\{T_v \cap T_j\} \tag{4.2}$$

  The decision variable matrix(DV) is created for newly generated dataset d_v, containing information for each dataset related to a task as per Eq. 4.3.

$$DV(d_v) = \begin{vmatrix} dataset & dg & dc & AP \\ d1 & 0 & dc2 & 0 \\ d2 & 4 & dc2 & 1 \\ d3 & 3 & dc1 & 1 \\ d4 & 5 & dc1 & 1 \end{vmatrix} \tag{4.3}$$

  The matrix represents the dependency gain(dg) of d_v with dataset d1 is 0 and d1 is placed at datacenter dc2. Similarly, representation is done for d2, d3 and d4

datasets. Then the DV matrix is sorted as per decreasing order of dependencies as shown in Eq. (4.4)

$$DV(d_v) = \begin{bmatrix} dataset & dg & dc & AP \\ d4 & 5 & dc1 & 1 \\ d2 & 4 & dc2 & 1 \\ d3 & 3 & dc1 & 1 \\ d1 & 0 & dc2 & 0 \end{bmatrix} \tag{4.4}$$

Awareness Probability (AP) of each dataset is evaluated. If AP < 1 then it symbolizes the dataset to be a fixed location dataset. Hence memory is updated with the particular data center. Dataset d1 is a fixed location dataset with awareness probability less than 1. Therefore, the memory of the crow is updated with the location dc2 for dataset d1. If AP $>= 1$ then the information for the datasets are stored in a decision variable vector(DCV) as represented in Eq. (4.5).

$$DCV(d_v) = \begin{vmatrix} dc & dg \\ dc1 & 5 \\ dc2 & 4 \\ dc1 & 3 \end{vmatrix} \tag{4.5}$$

Steps 5 to 7 are repeated for each dataset.

- **Step 5:** The feasibility of a new crow(dataset) position is assessed on all DCV datacenters. The feasibility to store dataset d_v is evaluated for each data center dc1, dc2. The feasibility to store dataset on a data center depends on the amount of storage used of the data center. The maximum storage used of a data center is represented as γ_{max}. γmax (percentage threshold) specifies whether or not the data center is overloaded and is also an experience parameter, γini represents initial storage usage parameter. Therefore, how much storage a data center can use for runtime data dc_j is $cs_j * (\gamma_{max} - \gamma_{ini})$. The γmax value depends on the system's complete workload. If the system workload is substantial, the value of γmax must be set to a larger value and if it is smaller, the value of γmax is set to small to prevent the collection of too many datasets in one data center.

The datacenter dc_s for the newly generated dataset d_v is decided, if $cs_s\gamma + s_v < cs_s\gamma_{max}$ is true, where s_v is dv and γ is dc_s as a percentage of current storage usage.

Position Matrix is created with the position value (PV) and fitness. The position value is updated by the data center if the data center is feasible else position value is not updated with the data center value. For example, feasibility to place newly generated dataset is evaluated with dc1 and dc2. If both dc1, dc2 have the storage capacity available then the value of dc1, dc2 is updated in position value matrix as in Eq. 4.6

$$PV(d_v) = \begin{bmatrix} dc & Fitnessvalue \\ dc1 & - \\ dc2 & - \\ dc1 & - \end{bmatrix} \tag{4.6}$$

- **Step 6:** The fitness function of a new position is evaluated using CSA. The goal of the proposed CSA-based data placement algorithm is to reduce global data transmission, hence the fitness function defined as per Eq. 4.7.

$$fitness = \frac{1}{\sum_{ts_i \epsilon T} minDatatransmissionAmount_{ts_i}} \tag{4.7}$$

where $minDataTransmissionAmountts_i$ refers to the minimum data movements required to execute the ts_i task. The calculation is performed on a protocol that the smaller dataset is always moved to larger storage.

Using Eq. 4.7 the fitness values are calculated for each data center dc1, dc2. These values are updated in the Position matrix(PV) as Eq. 4.8.

$$PV(d_v) = \begin{vmatrix} dc & Fitnessvalue \\ dc1 & 0.5 \\ dc2 & 0.1 \\ dc1 & 0.5 \end{vmatrix} \tag{4.8}$$

- **Step 7:** In this step fitness values of data centers, dc1 and dc2 are compared. The position of the newly generated dataset d_v is decided according to the best

fitness value. Accordingly, the memory of the crow is updated for the position of a dataset.

- **Step 8:** Check termination criteria.

  Steps 3 to 7 are repeated until maximum iterations $iter_{max}$ are reached and all the datasets of ready tasks are evaluated. When the termination criteria are met, in accordance with the fitness function, the best memory position which is the best dataset position on the data center is reported as the solution.

4.5 Comparison of CSA and PSO

CSA as PSO uses the population of seekers for searching search space. Using an element of the population increases the probability of finding an excellent solution. Other parameters of optimization algorithms need to be adjusted in addition to the number of iterations and population size. Parameter settings are the main drawback of optimization algorithms. The algorithms with fewer parameters are easier to adjust. There are only two parameters in CSA: a probability of awareness and flight length that need to adjust, while the maximum value of speed, inertia weight, social learning factor, and individual learning factor are four parameters in PSO. CSA is not a greedy algorithm but PSO is a greedy algorithm. A new position generated by the crow may not be the best position but it will move to a new position. It, therefore, increases the diversity of the various [150] solutions generated.

Good solutions are memorized in memory in CSA, but each particle moves in PSO towards the perfect location solution search in the group and on its own. So only the best solutions that have been found so far are used in CSA.

4.6 Experimental Evaluation

The simulation is performed on SwinDeW-C [149] (Swinburne Decentralized Workflow for Cloud), which is developed on the basis of SwinDeW-G [143] and SwinDeW [151]. There are 10 high-end PCs and 10 servers available. VMware software is set on the host and virtual machine as data centers are created to simulate a cloud computing environment.

Each data center consists of 8 storage-based virtual computing nodes. On these virtual data centers, SwinDeW-C streams the data between nodes. Web portal used on the application layer to deploy the application and upload the data.

SwinDeW-C is used in cloud applications on a large scale. SwinDeW-C's key components related to the data placement process are:

- **User interface module**: It is a kind of web portal that enables the users to deploy the workflows. For uploading the data of an application, uploading component is used and to monitor the execution of workflows monitoring component is used.

- **Data management module** : In SwinDeW-C the core part of data management is data placement component. The data catalog component is responsible to store information for different applications in data placement. Other essential components are data synchronization component, data replication, provenance data collection, and meta-data repository. These components are the supporting modules in data placement activity.

- **Other Modules**: Process repository is in Flow Management module that stores all instances of workflows executing. The scheduling of tasks is carried out by the scheduler in task management at the runtime stage. In addition, the Resource Management Module stores data center usage information and triggers the data placement adjustment process.

4.6.1 Experimental Setup

The proposed algorithm is evaluated on SwinDeW-C. The test workflows are generated randomly for execution on SwinDeW-C. Hence, the evaluation results are not dependent on any specific application. The number of existing and generated datasets are set to be the same for every test workflow. Every test workflow exhibit the same number of tasks and existing datasets. The assumption is that each task will generate only one dataset. The complexity of test workflow can be controlled by varying number of datasets. The random number of tasks will use every dataset. The tasks using the generated datasets are executed only after the task that provides its input to the task completed execution. The complexity between tasks and datasets is controlled by changing the range of random number. The factor of a number of fixed location datasets also impacts the

performance of the algorithm. The parameters for simulation settings are as in Table 4.1.

Table 4.1: Parameters for data placement

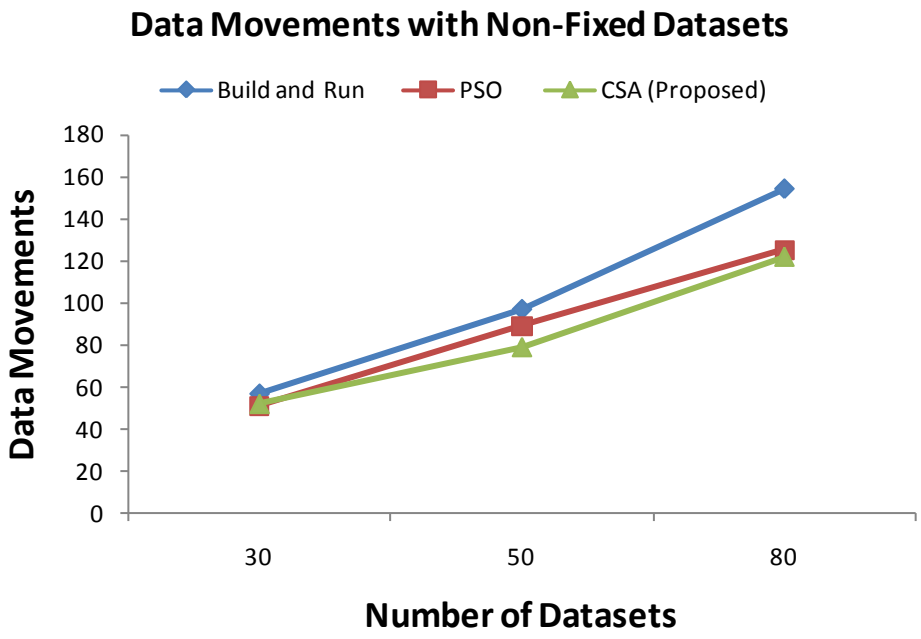
Parameter	**Value**
Data Centers	10-15
Storage Capacity	100GB-500GB
Upload Cost Rate	$0.08-$0.12/Gb
Download Cost Rate	$0.13-$0.17/Gb
Size of Dataset	700-800 MB

4.6.2 Results and Discussion

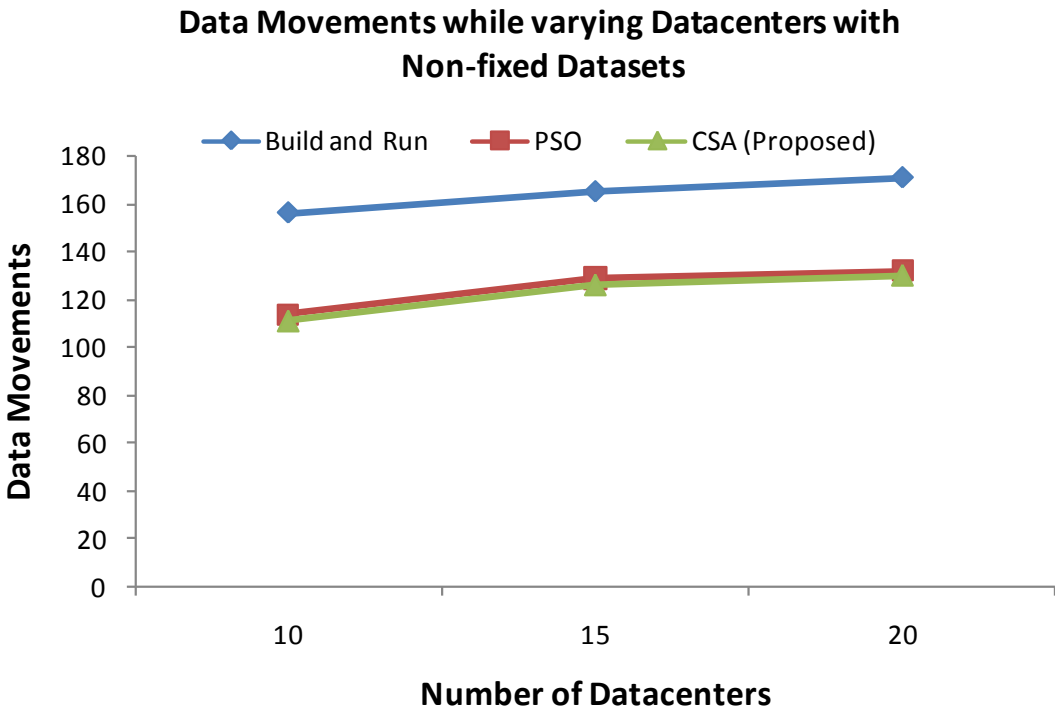
Figure 4.5 displays the data movement when workflows are executed on a different number of data centers without fixed location datasets and Figure 4.6 displays the data movement with 20% fixed location datasets. It can be seen an increase in data movement when the number of data sets have increased.

In an experiment, three types of test workflows are used with a varying number of datasets. The test workflow dataset count is fixed to 80 and the result is depicted in Figure 4.5(a) with varying number of datasets 30, 50 and 80. The same experiment is conducted with 20% fixed location datasets as shown in Figure 4.6(a). The test workflows with complexities are executed on a different number of data centers and the results are as shown in Figure 4.5(b) and the same experiment is conducted with 20% fixed location datasets with the result as in Figure 4.6(b).

Experiment 1 (Data Movements Without Fixed Location Datasets): From the result, conclusions can be derived that CSA based algorithm effectively reduces the total data movement in the execution of workflow in comparison to build and run an algorithm, and PSO based algorithm. The data movement by CSA is reduced by 3% in comparison to PSO and by 15% in comparison to build and run algorithm as shown in Figure 4.6(a). Figure 4.6(b) depicts the reduction in movement of data when an evaluation is performed by varying number of data centers as 10, 15, 20. The CSA based strategy shows the minimum data movement even if the number of data centers

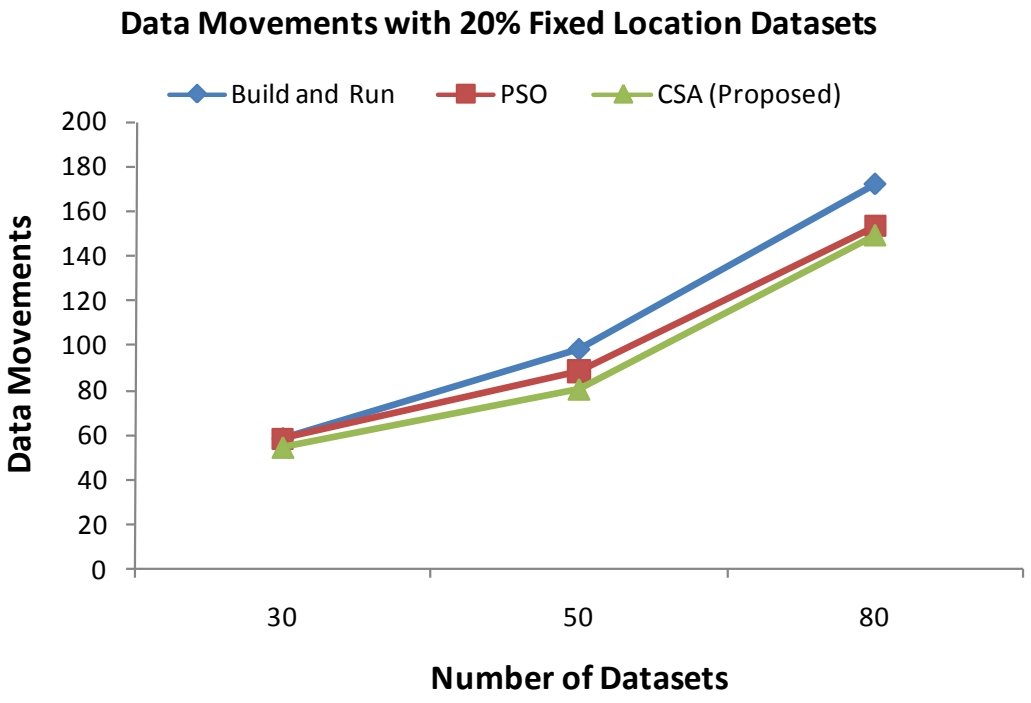


(a)

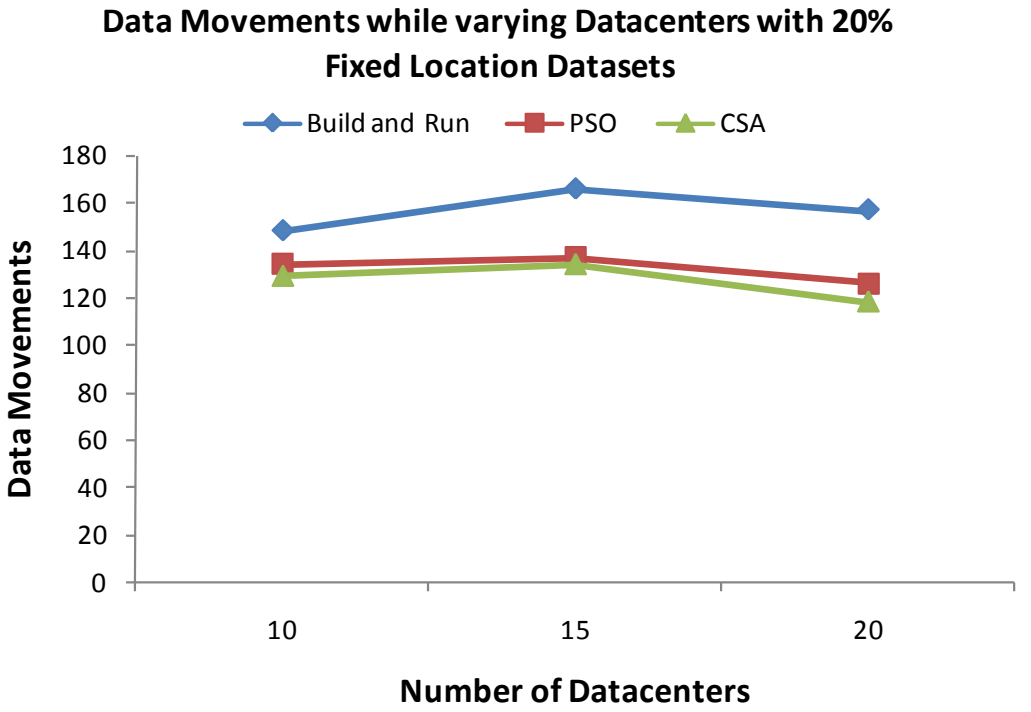


(b)

Figure 4.5: Data movement without storage limit without fixed location datasets (a) varying number of datasets (b) varying number of datacenters

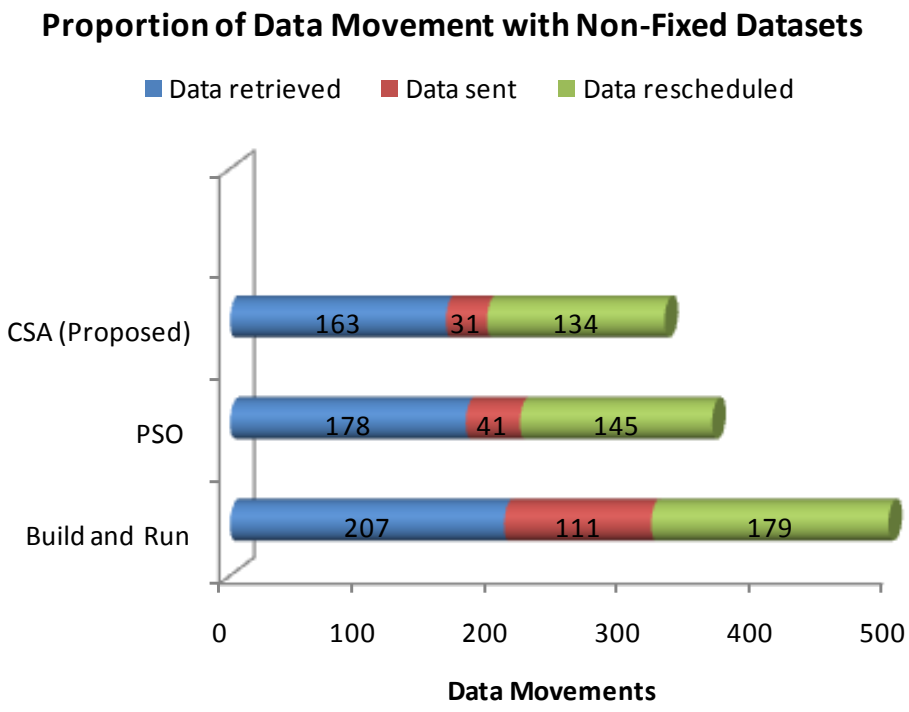


(a)

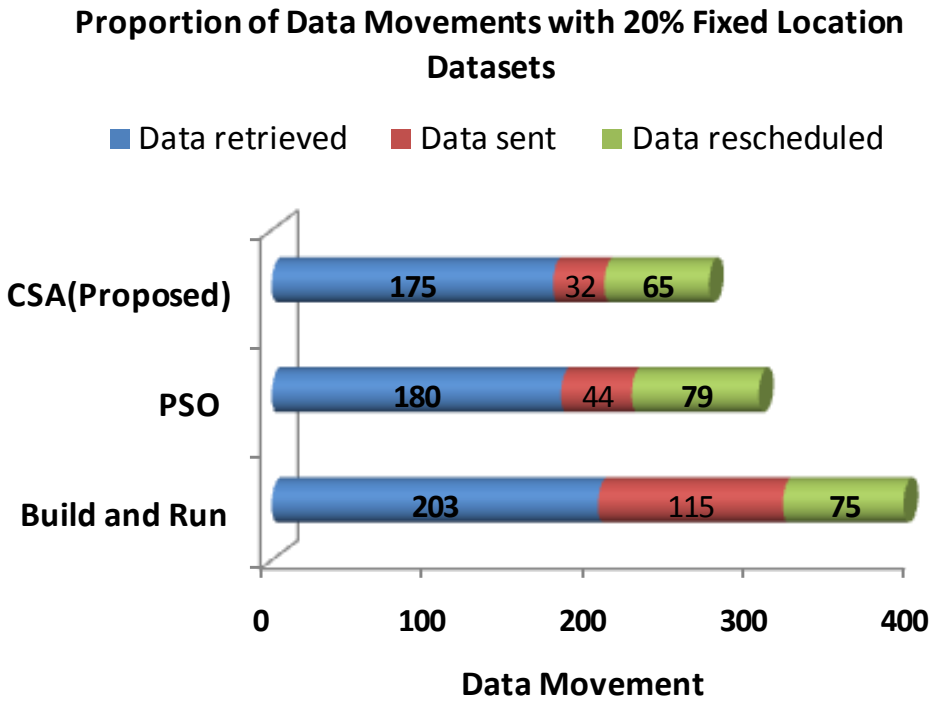


(b)

Figure 4.6: Data movement without storage limit with 20% fixed location datasets (a) varying number of datasets (b) varying number of datacenters



(a)



(b)

Figure 4.7: Proportion of three types of data movements (a) Non-Fixed datasets (b) 20% Fixed datasets

is increased.

Experiment 2 (Data Movements With Fixed Position Datasets): In this experiment, the researcher changed initial datasets to 20% fixed location datasets to validate the proposed technique. Figure 4.6 shows the results with 20% fixed location datasets. The experiment conditions are same as in Figure 4.5. The results depicted in Figure 4.6 shows that CSA based approach reduces the data movement by 2% as shown in Figure 4.6(a) with varying datasets and 5% from PSO based approach with varying number of data centers as shown in Figure 4.6(b). Similarly, CSA reduces the data movement by 10% with varying datasets as shown in Figure 4.6(a) and 13% with varying number of data centers as shown in Figure 4.6(b) from build and run algorithm. Therefore, a conclusion is derived that the proposed technique offers the minimum data movement and also works well with fixed location datasets.

Experiment 3 (Proportion of Three Types of Data Movement): Figure 4.7 shows that when the runtime algorithm is executed, a large amount of data is rescheduled. The least re-allocation of data occurred when using the CSA-based algorithm for data placement. Considering the case where there are no fixed location datasets, the result is shown in Figure 4.7(a). The number of data movements for CSA is 134 in comparison to PSO, build and run algorithm that posses data movements 145, 179 respectively. Hence, CSA reduced the data movement in comparison to other algorithms.

Considering the case when there are 20% fixed location data sets, then data movements for CSA, PSO, build and Run are 65, 79 and 75 respectively as shown in Figure 4.7(b). Hence, CSA posses the least amount of movement of data. The conclusion is CSA based data placement lower down the movement of data. Thus, the proposed technique decrease the execution time of workflow.

4.7 Summary

In this chapter, different characteristics of scientific cloud workflows have been evaluated and data placement technique using Crow Search Algorithm (CSA) is proposed. It automatically allocates the intermediate data among the data centers based on the factor of dependency with data centers. Simulations performed reveals that CSA based

run time data placement technique minimize the movement of data in workflow execution. The data movements also decrease in-spite of having fixed location datasets in the workflow. The experiments are also conducted by considering with 20% fixed location data sets. The approach is flexible to work with fixed and non-fixed location datasets.

CHAPTER 5

DATA PLACEMENT ORIENTED SCHEDULING ALGORITHM FOR SCIENTIFIC WORKFLOWS IN CLOUD: A BUDGET-AWARE APPROACH

5.1 Introduction

Cloud computing is an effective and reliable business computing model which distributes the incoming requests on the pool of shared resources and provides online physical machines. It transforms communication, storage and computer resources on-demand into ordinary utilities and commodities [152, 153]. Without owning a cloud physically, this feature brings large-scale computing. Infrastructure as a Service (IaaS) is most common among the cloud services which is capable of delivering or releasing pre-configured virtual machines. The client can access the virtual machines remotely with minimum renting cost [154].

One of the popular frameworks is to characterize data-intensive scientific workflow

applications that are hosted and deployed on various cloud providers such as Amazon EC2 [154]. It includes many data dependencies or inter-task control dependencies. It is represented in the form of a Direct Acyclic Graph (DAG) with tasks as nodes and edges as dependencies among tasks. The IaaS services provided by cloud service providers are purchased by the clients for the execution of scientific workflows while fulfilling the Quality of Service (QoS) parameter. The users are charged by the cloud provider as per the billing cycle of the deployed workflows. Hence, the aim is reducing monetary costs and makespan of execution of the workflow. Alkhanak et al. [155] described the cost-aware approach for Service Level Objective (SLO) challenges (e.g. makespan) and system functionality challenges. In this chapter, the researcher focus on the development of a new scheduling algorithm for workflows to minimize makespan and monetary and budget-compliant costs. The focus in this chapter is on designing the scheduling algorithm taking into account the user budget and deadline trade-off.

In order to optimize monetary costs in heterogeneous computing environments, significant research has been dedicated to the study of workflow scheduling. Heterogeneous Earliest Time First (HEFT) gave adequate performance when applied as a heuristic scheduling algorithm. It achieves the higher search efficiency by reducing the range of search space of task scheduling problem to minimum [156]. HEFT algorithm performs poorly with some Directed Acyclic Graphs (DAG) tasks. It gives high performance for scheduling small-scale task graphs, but with large-scale task graphs scheduling issues are unsatisfactory. In this chapter, HEFT algorithm is improved to achieve task scheduling performance.

Many indicators exist for evaluating task scheduling performance in cloud, such as load balance, makespan, green computing economic principles, Quality of Service (QoS), and average response time [157, 158]. The most important factors affecting system performance is the makespan. Less energy consumption with higher throughput rates can be considered as a smaller makespan. This research focus to propose a novel scheduling workflow algorithm while considering the user's budget. The algorithm tries to maximize resource utilization and find the best scheduling solution. The makespan is used as an important indicator of evaluation. The chapter's main contributions are as follows:

- A detailed design of the model with multiple VM categories and data centers.

- Design and development of Data Placement Oriented HEFT (DPO-HEFT) algorithm by taking budget and deadline into consideration.
- The experimental evaluation of extensive proposed DPO-HEFT using workflowsim and reak workflow traces.

5.2 Related Work

This section comprises the description of different scheduling algorithms used for scientific workflows. The benefits and problems in HEFT algorithms are analyzed. The task scheduling algorithm is proposed in this research by combining the advantages of different types of algorithms. The scheduling of tasks on virtual machines is a process of optimally scheduling the tasks with effective performance [159]. The task scheduling is a combinatorial problem with no definite solution found so far [160]. Mostly list scheduling algorithms are static. The list scheduling algorithms comprise of two phases. The first phase obtains an appropriate scheduling strategy by deciding the priority order and in the second phase, an algorithm is responsible for assigning tasks to the processor as per the order of priority. Different classic list scheduling algorithms are HEFT and Critical Path of Processor (CPOP) [156]. One of the most used heuristic algorithms is HEFT, that aims at minimizing the makespan. It is one of the most popular algorithms used for scheduling of scientific workflows, even if the number of heterogeneous computing resources is limited. Practically, it is a static algorithm which provides a mapping of the tasks to the resources are determined by the scheduler. It also determines the time taken to start each task before the execution of a workflow. The different resources available for this scheduler are not prone to be homogenous. In it, schedules are placed in the order of execution where tasks are classified. There are two major stages in HEFT. One stage of list-based heuristic is prioritization phase where priorities of all the tasks are calculated and another stage is to select the instances where the selection of tasks is performed as per the priority order and scheduling is performed to the best resource. The upward ranking method is used by HEFT for sorting all the tasks of workflow in decreasing order of priority. For each task, the upward rank is defined as the length of the longest path of a task to the exit task of a workflow. Both the task calculation time and edge communication time on the path are consid-

ered [156]. HEFT solves the only problem of mono-objective workflow scheduling. Different versions of HEFT are developed in different environments. One of the algorithm same as HEFT is Critical Path on a Processor (CPOP) algorithm. In the CPOP algorithm, to calculate the priority value of tasks both task's upward rank value and downward rank value are added. The sorting of the tasks is performed by priority value from highest to lowest. The tasks that possess the highest priority value are known as critical tasks. In the second phase, the assignment of the critical tasks to the critical processors is performed. The processor that provides minimum execution time for a task to the processor is known as a critical processor. The allocation strategy employed by CPOP is same as HEFT for non-critical tasks. Various other list scheduling algorithms are heterogeneous scheduling algorithm with improved task priority (HSIP) [161], the predict earliest finish time (PEFT) algorithm [162], the Median Deviation-based Task Scheduling (MDTS) algorithm [163] etc.

Zeng et al. [164] introduced solutions to schedule workflow in clouds with cost and security considerations. The problem of a multi-objective off-line scheduling problem for deterministic workflows consisting of adhering to budget and meeting deadlines are extensively studied. Although the problem of multi-objective offline scheduling problem consisting of meeting deadlines and respecting a budget has been extensively studied for deterministic workflows [165, 166].

Malawski et al. [167] introduced an algorithm for scheduling workflows while considering the limits of time and budget. However, the objective is different as it considers the ensembles of a workflow.

5.3 System Architecture

In this section, the system architecture of scientific workflow scheduling is described.

5.3.1 Cloud Data Center Model

In a cloud datacenter, a set of P types of instance series $VM_p = VM_p^1, VM_p^2.., VM_p^k..VM_p^{q_s}$ is provided. The instance series of three different kinds provided by Amazon EC2 (memory optimized, computing optimized and storage optimized) [168] are shown in Table 5.1, Table 5.2 , Table 5.3. It is denoted as VM_p^1, VM_p^2 and VM_p^3 in this research. Several characteristics specify a VM type VM_p^k, including processing capacity pc_p^k in

million floating point operations per second (MFLOPS), instance series type p, cost per hour CT_p^k , storage and memory space. VMs are charged in unit time (e.g., minute, hour) based on their lease time and amount of part-time unit used. It is rounded up to the next full-time unit. If the lease time unit is an hour, then the time of 2 hours and 1 minute is rounded up to 3 hours. Hence, it results in a three-hour lease cost calculation. No limit is imposed on the number of VMs and can be accessed in clouds.

Table 5.1: Amazon EC2: Optimized instance series of memory

	vCPU	**Memory(GiB)**	**Processing Capacity (MFLOPS)**	**Cost Per Hour**	**Storage(GB)**
r3.large	2	15	8800	$0.175	1x32SSD
r3.xlarge	4	30.5	17600	$0.350	1x80SSD
r3.2xlarge	8	61	35200	$0.700	1x160SSD
r3.4xlarge	16	122	70400	$1.400	1x320SSD
r3.8xlarge	32	244	140800	$2.800	2x120SSD

Table 5.2: Amazon EC2: Optimized instance series of computing

	vCPU	**Memory(GiB)**	**Processing Capac-ity(MFLOPS)**	**Cost Per Hour**	**Storage(GB)**
c3.large	2	3.75	8800	$0.105	2x16SSD
c3.xlarge	4	7.5	17600	$0.210	2x40SSD
c3.2xlarge	8	15	35200	$0.420	2x80SSD
c3.4xlarge	16	30	70400	$1.840	2x160SSD
c3.8xlarge	32	60	140800	$1.680	2x320SSD

Table 5.3: Amazon EC2: Optimized instance series of storage vs CPU

	vCPU	**Memory(GiB)**	**Processing Capac-ity(MFLOPS)**	**Cost Per Hour**	**Storage(GB)**
d2.xlarge	4	30.5	17600	$0.69	3x2000SSD
d2.2xlarge	8	61	35200	$1.38	6x2000SSD
d2.4xlarge	16	122	70400	$2.76	12x2000SSD
d2.8xlarge	32	244	140800	$5.52	24x2000SSD

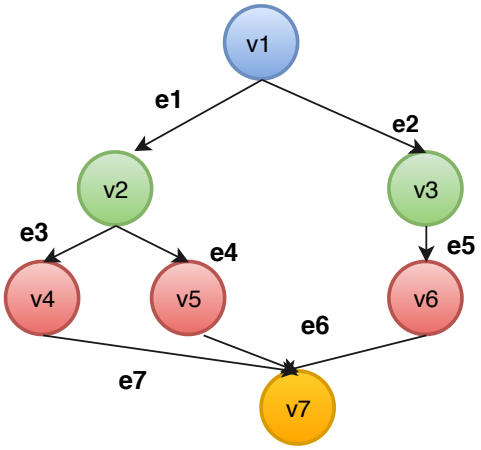


Figure 5.1: Workflow model

All processing units use only one data center. Due to the security issues, the units do not interact directly. For the execution of the task T on Virtual Machine (VM), the data of the predecessor task also needs to be on the same VM, unless the data is produced on some other VM. The researcher assumes a large bandwidth of data centers for all units of processing. The actual processing units are the VMs, which are classified with a set of parameters from different providers. A categorized (c) VM exhibits n_c processors. These processors can process one task at a particular time instant. VM speed sp_c is considered as a number of instructions handle per unit of time, a cost per unit of time $c_{l,c}$; and an initial cost $c_{ini,c}$ defined respectively. All the VMs consider the unlimited amount of time to boot before starting the execution of tasks. The start time of the task is not included in the cost of VM. Hence, the platform consists of a different set of possible c categories of VMs. Some assumptions taken are:

- Same bandwidth for each VM in all directions.
- Storage capacity of the VM is enough. So, there will be no problem with memory/space overflow.
- Time of initialization of VM is the same.

VMs exhibit different start-up times defined as $H_{start,vm}$. The ending time of VM is $H_{end,vm}$, when the data created by the last computed task is transferred to the data center. VMs are allocated contiguously. For the discontinuous allocations, VMs can be released and later on a new one can be allocated.

5.3.2 Workflow Model

The directed acyclic graph (DAG) represents a workflow application Wf where $Wf = (V_i, E_i)$. E_i represents edges control or data dependencies between them and V_i=v_{i1},v_{i2}..v_{in} is a set of vertices representing tasks. A dependency e_{ij} is the precedence constraint of the form (v_{i1},v_{j1}), where v_{i1},$v_{j1} \in V_i$ and $v_{i1} \neq v_{j1}$. This refers to that the child task can only complete its execution if the parent task has completed the execution. An example workflow is shown in Figure 5.1. To each dependency $(Task_i, Task_j)$ ε E represents the data in the size ($size(d_{Task_i,Task_j})$). The task is said to be ready task if all its predecessors have finished execution and the output data required by it is available.

5.3.3 System Model

First, the problem definition is refined. For the workflow scheduling, there are three layers in the cloud environment shown in Figure 5.2 [169]. All the preceding tasks are represented in a task graph layer. The virtual machine network is represented in the resource graph layer and set of network-connected data centers are represented in the cloud infrastructure layer. According to Abdelkader et al. the problem of task scheduling in heterogeneous systems is to correctly allocate tasks to machines for optimizing certain performance metrics such as resource utilization and execution time [170].

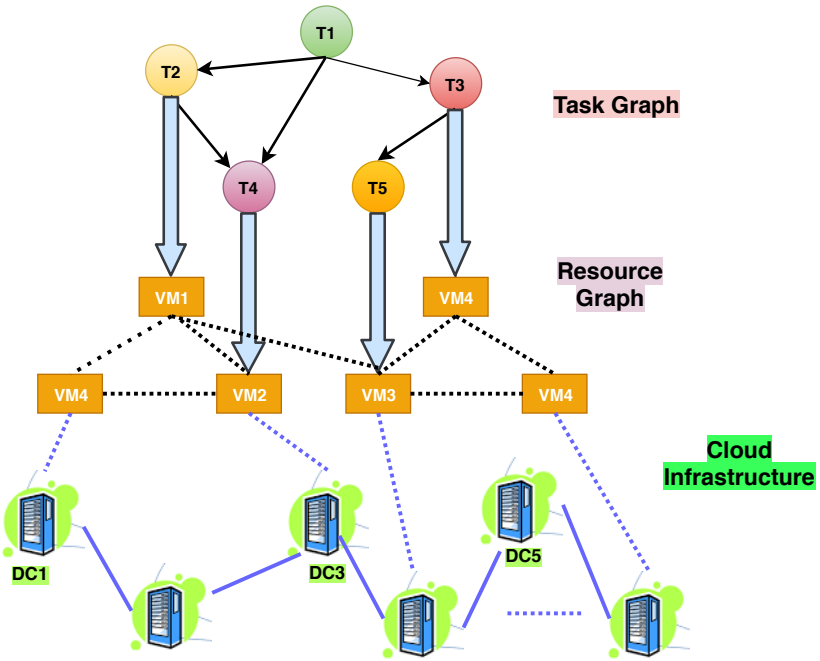


Figure 5.2: Model for scheduling scientific workflows in cloud

- Cost: The cost model represents different features of existing cloud providers (Amazon, OVH, Google) offerings. The complete cost of the entire execution of the workflow is the sum of the cost of using the VMs ($Cost_{vm}$) and the cost of using the datacenter ($Cost_{DCT}$). The cost ($Cost_{vm}$) of the usage of a VM of category (C_{vm}) is calculated as follows:

$$Cost_{vm} = (H_{end,vm} - H_{start,vm}) * C_{l,c_{vm}} + C_{ini,c_{vm}} \tag{5.1}$$

The start-up cost is $C_{ini,c_{vm}}$ as in Eq. 5.1. $C_{l,c_{vm}}$is a term proportional to usage duration $H_{end,vm} - H_{start,vm}$. The datacenter cost is based on a cost per time-unit $Cost_{l,DCT}$ to which transfer cost is added. The transfer cost is calculated by the amount of data transferred into datacenter $(size(d_{in,DCT}))$ and from the data center $(size(d_{DCT,out}))$. $d_{in,DCT}$ represents input data to tasks in the workflow and $d_{DCT,out}$ represents data output from tasks. H_{start} is the time when the first VM is booked and H_{end} is the last time, the last processed task data was sent to the data center. So the total cost of data centre (DCT) is calculated as Eq. 5.2.

$$Cost_{DCT} = (size(d_{in,DCT}) + size(d_{DCT,out})) * Cost_{trsf} + (H_{end,last} - H_{start,first}) * Cost_{l,DCT} \tag{5.2}$$

The total cost of processing the workflow where I_{vm} are the used VMs for execution is represented in Eq. 5.3

$$Cost_{wf} = \sum_{vm \varepsilon I_{VM}} Cost_{vm} + Cost_{DCT} \tag{5.3}$$

- Objective: The main goal is to meet the deadline while also considering the budget, given a deadline Dl and a budget BD as per Eq. 5.4.

$$Dl \geq (H_{end,last} - H_{start,first}) \quad and \quad BD \geq Cost_{wf} \tag{5.4}$$

The complex objective is to find a schedule that minimizes the makespan while considering the budget as in Eq. 5.5

$$min(H_{end,last} - H_{start,first}) \quad where \quad BD \geq Cost_{wf} \tag{5.5}$$

The goal is defined as per the above-mentioned definitions. The basic goal is to minimize the makespan and cost. Here, the approach designed to find a schedule with a minimum cost, where the cost must not exceed the budget. The conditions ensure successful execution of workflow within a specified constraint of a budget. The weight of a task is represented as $\bar{\omega_{task}} + \sigma_{task}$. It follows a Gaussian law with mean $\bar{\omega_{task}}$ and standard deviation σ_{task} which can be estimated through sampling method.

5.4 Research Methodology

This section introduces Data Placement Oriented-HEFT (DPO-HEFT), that is using the data placement as sub-module in HEFT and extending HEFT while considering the budget. The introduced algorithm accounts for the budget constraint as well.

Algorithm 2 DPO-HEFT

1: **procedure** $DPO-HEFT(wf, BD_{calc}, P)$
2: $\bar{sp} \leftarrow Mean_{Speed}(P)$
3: $bwd \leftarrow getbwd(P)$
4: $BD_{task} \leftarrow divide_{BD}(wf, BD_{cal}, \bar{sp}, bwd)$
5: $Sorted_{list} \leftarrow SortingTasksByRanks(wf, \bar{sp}, bwd, \bar{lat})$
6: $hole, newhole \leftarrow 0$
7: **for** $Task_j in Sorted_{list}$ **do**
8: $host \leftarrow getBestHost(Task_j, BD_{task}, p, NewHole)$
9: $hole \leftarrow newhole$
10: $sch[Task_j] \leftarrow host$
11: $schedule(Task_j, host)$
12: $update(Used(vm))$
13: **end for**
14: **end procedure**

The objective of the proposed algorithm is to gain the highest parallelism as per the budget of the user. The proposed algorithm works in five phases:

- Phase 1: Budget Distribution
- Phase 2: Workflow Clustering

- Phase 3: Deadline Distribution
- Phase 4: Task Selection
- Phase 5: Instance Selection

5.4.1 Phase 1: Budget Distribution

In this phase, the budget is distributed globally among different tasks on different levels of the workflow. According to the budget, for a given workflow, firstly fraction is reserved for the costs of the initialization of the datacenter and VM. Then, the remaining budget is divided into tasks. The task at each level is scheduled to an instance according to the budget assigned. The initial budget is denoted by BD_{ini}. For the estimation of the reserved amount the parameters are:

- Cost of Data Center: The duration of $H_{end,last} - H_{start,first}$ is to be estimated. The total duration is calculated as maximum work ($Work_{max}$) as represented in Eq. 5.6.

$$Work_{max} = \sum_{task \varepsilon wf} (\bar{\omega_{task}} + \sigma task) \tag{5.6}$$

  $H_{end,last} - H_{start,first}$ is represented as in Eq. 5.7

$$H_{end,last} - H_{start,first} = \frac{Work_{max}}{sp_1} + \frac{size(d_{in}, DCT) + size(d_{DCT}, out)}{bwd} \tag{5.7}$$

  The cost of input/output data is paid several times with C_t for the outside world and with $C_{l,DCT}$ as per the usage of the data center.

- Initialization of VM: We assume that the first category of VM is different for each task, so we budget the amount $s * C_{ini,1}$.

Choices are conservative. All in all, we reserve the corresponding budget amount and are left for the tasks with BD_{cal}. This reduced BD_{cal} budget is shared proportionally among tasks as in Algorithm 3.

In Algorithm 3, the time required to execute a task j including the transfer time is calculated as in Eq. 5.8.

Algorithm 3 Budget Division Algorithm

1: **procedure** $BUDGET_{div}(wf, BD_{cal}, \bar{sp}, bwd)$
2: $Work_{max} \leftarrow GetMaximumWork(wf)$
3: $dt_{max} \leftarrow GetMaximumTotalDataTransferred(wf)$
4: **for** $task$ in WF **do**
5: $budget[task] \leftarrow BD_{task} \leftarrow BD_{cal} \times \frac{\frac{\bar{\omega_{task}}+\sigma_{T_j}}{sp_1}+\frac{size(ds_{pred,T_j})}{bwd}}{\frac{Work_{max}}{\bar{sp}}+\frac{dt_{max}}{bwd}}$
6: **end for**
7: **end procedure**

$$time_{cal,T_j} = \frac{\bar{\omega_{task}} + \sigma_{T_j}}{s_1} + \frac{size(ds_{pred,T_j})}{bwd}$$
$$where \quad size(ds_{pred,Task_j}) = \sum_{Task_j',Task_j \varepsilon E} size(d_{task_j',Task_j}) \tag{5.8}$$

$size(ds_{pred,Task_j})$ is the size of input data from all its predecessors. The corresponding budget to the task j in proportion of entire workflow is allocated as in Eq. 5.9 :

$$BD_{task_j} = \frac{time_{calc,T_j}}{time_{calc,wf}} \times BD_{cal} \tag{5.9}$$

The time required to execute the entire workflow(wf) is calculated as in Eq. 5.10 where dt_{max} represents total volume of data within workflow. :

$$time_{cal,wf} = \frac{wf_{max}}{\bar{sp}} + \frac{dt_{max}}{bwd} \quad where \quad dt_{max} = \sum_{Task_j',Task_j \varepsilon E} size(d_{task_j',Task_j}) \tag{5.10}$$

The computed weights are divided by the mean speed $\bar{sp}$ of different categories of VM while the data sizes ($size(d_{task_j',Task_j})$) are divided by bandwidth(bwd) between data centres and VM. This step is performed in Line 4 of Algorithm 2.

5.4.2 Phase 2: Task Clustering in Workflow

In this, the proposed Hybrid Balanced (HYB) task clustering algorithm 1 is used for clustering tasks which are not dependent on the input of the user. It is able to cluster the tasks vertically with single parent single child relationship and the tasks horizontally as well. The system overhead is reduced while involving the best utilization of resources.

The proposed algorithm is explained in our previous work [171]. This phase gives the clusters at each level represented by *CLS*.

5.4.3 Phase 3: Deadline Distribution

The objective of this phase is to complete the execution of the workflow in time. As there are large and complex scientific workflows in different areas such as medical modeling, weather forecasting, climate modeling, etc. These are the examples of deadline sensitive applications and need to be executed in a specified time constraint. There are two kinds of deadline constrained application. One is the hard deadline constrained, in which once missed the deadline can result in disaster on life or environment. Another is the soft deadline constraints, in which small margins in deadlines are tolerable [172].

So to fulfill the objective of this phase, the deadline needs to be distributed as sub-deadlines among the tasks of the workflow. It involves distributing the deadline among different levels of the workflow. The tasks are distributed proportionally among each level based on the user's deadline (Dl). Each level is assigned a sub-deadline $Dl_{sub}(Level)$. For the achievement of deadline objective, an assumption is that every task completes the execution before the assigned deadline. The estimated initial deadline of a task is represented as in Eq. 5.11.

$$Dl_{sub}(Level) = max_{t_j \varepsilon CLS}[time_{cal,T_j}] \tag{5.11}$$

The time taken to execute each task in cluster $(time_{cal,T_j})$ is represented in Eq. 5.8. The deadline value for all the levels is estimated. Then, it is distributed among the tasks at a level. The proportion unit at a level is represented by Eq. 5.12.

$$\eta_{deadline} = \sum_{level=1}^{t_{entry}} Dl_{sub}(Level) \times CLS(level) \tag{5.12}$$

where $Dl_{sub}(Level)$ is sub-deadline of level and $CLS(level)$ represents clusters at level.

The deadline factor $(factor(deadline))$ is represented as per Eq. 5.13.

$$factor(deadline) = \frac{Dl - Dl_{sub}(Level)(1)}{\eta_{deadline}} \tag{5.13}$$

$Dl_{sub}(Level)(1)$ is the level consisting of an exit task. The remaining deadline subtracting the total deadline(Dl) is represented in the numerator of Eq. 5.13. The length of the deadline for each level is distributed as per Eq. 5.14.

$$Dl_{sub}(Level)(l) = factor(deadline) \times Dl_{sub}(Level) \times CLS(level) + Dl_{sub}(Level) \tag{5.14}$$

5.4.4 Phase 4: Task Selection

The selection of the tasks is performed in this phase according to the upward rank value [156]. The upward rank is based on mean communications costs and mean computation cost. The tasks are sorted according to decreasing order of rank values. If two tasks have the same rank then the task whose immediate successor is having highest upward rank is selected. This is performed in the Algorithm 2 (Line 5).

5.4.5 Phase 5: Instance Selection

In this phase, the selection of instance is performed according to two conditions, (a) placement of data sets and (b) budget of the user.

- Placement of data sets: In this phase, the CSA based data placement strategy for scientific cloud workflows is executed. This technique is a two-phase algorithm. Firstly, build time algorithm used for distributing the initial data sets to appropriate data centers. Secondly, a runtime algorithm is used to distribute the intermediate datasets to data centers. Runtime stage uses the CSA based approach. The algorithm is explained in our previous work [173].

- Sub-Budget: To host every ready task, sub-budget is used to select the best VM as per Algorithm 4. On a platform P, the best host of task j is the one that provides the best EFT (Earliest Finish Time) for $Task_j$, including the budget ($BD(task_j)$) allocated to task j. The P platform defines as the set of host candidates, consisting of VMs already used in addition to one fresh VM in each category. The total time of execution of task j on the host is calculated as in Eq. 5.15.

$$time_{exec(task_j,host)} = \delta_{new} \times time_{boot} + \bar{\omega_{task_j}} + \frac{\sigma_{task_j}}{s_{khost}} + \frac{size(ds_{in,task_j})}{bwd} \tag{5.15}$$

Algorithm 4 Best Host Selection Algorithm

1: **procedure** $BestHost(Task_j, budget[task], P, hole)$
2: $BD_{task} \leftarrow budget[task] + hole$
3: $New_{host} initialized \leftarrow category(cheapest)$
4: $Best_{host} \leftarrow vm, where \quad vm \varepsilon New_{vm} and c_{vm} = 1$
5: $minEFT \leftarrow EFT(task_j, Best_{host})$
6: **for** $eachhost(Used_{vm} \bigcup New_{vm})$ **do**
7: **if** $(EFT_{task_j,host}) < minEFT$ and $(Cost_{task_j,host} \leq BD_{task_j})$ and $(DataTransferCost_{task_j,host} \leq BD_{task_j})$ **then**
8: $minEFT \leftarrow EFT_{task_j,host}$
9: $bestHost \leftarrow Host$
10: $hole \leftarrow BD_{task_j} - C_{task_j}$
11: **end if**
12: **end for**
13: **end procedure**

In Eq. 5.15 , δ_{new} is a variable and depends on the type of host allocated. The value is 1, if $host \, \varepsilon \, New_{vm}$ else 0. If the host is the used vm , then some of the input data may be available. So the size used will be $size(ds_{in,Task_j})$ instead of $size(ds_{pred,Task_j})$.

For the computation of $EFT_{task_j,host}$, Earliest begin time(EBT) is accounted on that host (EBT_{host}) and $time_{exec(task_j,host)}$ is added. The earliest begin time depends upon the availability of the host and input data transfer to host any task$(task_j)$. All of the data produced by the predecessor of the task($task_j$) on another host is to be sent to the data center before submission to host as there are no direct communications between VMs. These transfers have the associated cost with it, that is added to $time_{exec(task_j,host)} \times c_{l,host}$ for the computation of the total cost $(C_{task_j,host})$ which is for the execution of task j on the host. The algorithm acquires any unused fraction of the budget used when assigning previous tasks: this is the variable *hole*. The *hole* acquires the leftover budget of the user. So DPO-HEFT is an extension of the HEFT algorithm that deals with provisioning of the budget while also considering the placement of data sets used for the tasks.

5.5 Experimental Evaluation

The proposed scheduling DPO-HEFT is evaluated through a set of experiments.

5.5.1 Experimental Methodology

The introduced algorithm is evaluated on simulator workflowsim. The model used is three different categories of VM along with costs from Google Cloud, Amazon, and OVH. The VM cost is dependent on the mean prices of instances. The payment for each used second is done for VM.

Five types of workflows: Montage, LIGO, Cybershake, Epigenomics, and SIPHT are used to perform the experiments. In the LIGO workflow, a large amount of input data is involved. There are many parallel tasks in it. In the Cybershake workflow, fifty percent of the tasks consist of input data. The first set of tasks in a workflow generate data in parallel. The generated data is used by a directly connected task. There are many interconnected tasks in MONTAGE workflow. So, it is difficult to maintain parallelism in it. The workflowsim simulator generates a benchmark for each type of workflow, with different task numbers as according to the workflow. The comparison of the results is performed with HEFT and HEFTBUDG algorithms. The results for the different number of tasks are reported.

5.5.2 Experimental Setup

Workflowsim [123] is used for the simulation of experiments. It supported the workflow applications for deploying and scheduling workflows. The workflow generator is used to produce 100 synthetic workflows. Synthetic workflows are obtained from the information collected from the actual execution of scientific workflows on the cloud. Once node number and the type of a desired synthetic workflow is set. The other remaining characteristics of this workflow are determined automatically, such as average data size and the edge number. Also, the task calculation and communication costs in this workflow are also determined. The process of determination involves the calculation of the cost of task and cost of communications between tasks [174]. The configuration setting is depicted in Table 5.4.

Table 5.4: Configuration setup for the experiment

Parameter	Value
No. of virtual machines	5-20
Memory Capacity	512 MB
Processing capacity	1000 MIPS
Network Bandwidth	15 MB/s

Application Model

The real world workflows are used [173] i.e Montage, SIPHT, LIGO, Epigenomics and Cybershake. These workflows are widely used to evaluate scheduling algorithms performance. The DAG features of these workflows include a number of edges and nodes, average data size and execution time of tasks. The structures of these five workflow applications are illustrated in Figure 5.3 for Montage, Figure 5.4 for CyberShake, Figure 5.5 for Epigenomics 5.7 and Figure 5.6 of SIPHT scientific processes. The workflows with a different number of tasks are selected for the experiment.

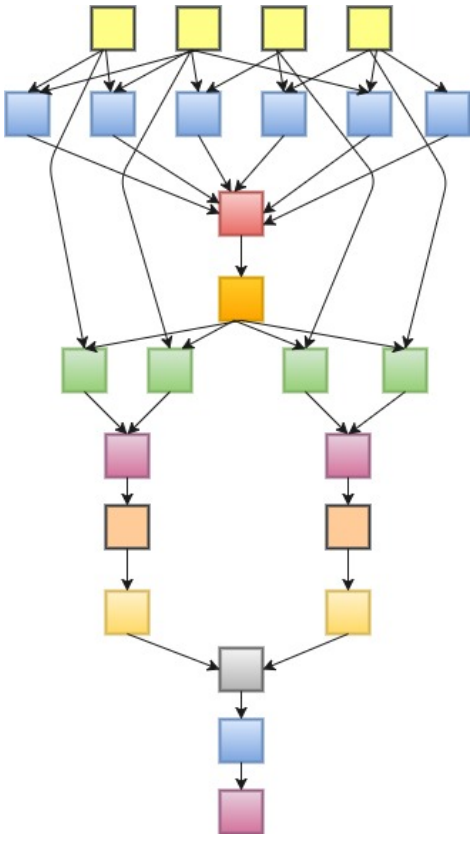


Figure 5.3: A simplified display of the Montage workflow [3].

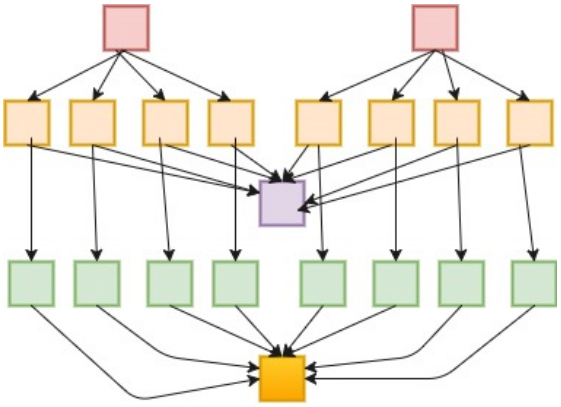


Figure 5.4: A simplified display of Cybershake Workflow [3].

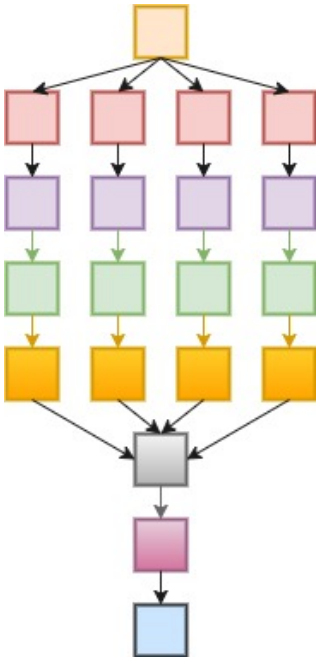


Figure 5.5: A simplified display of Epigenomics workflow [3].

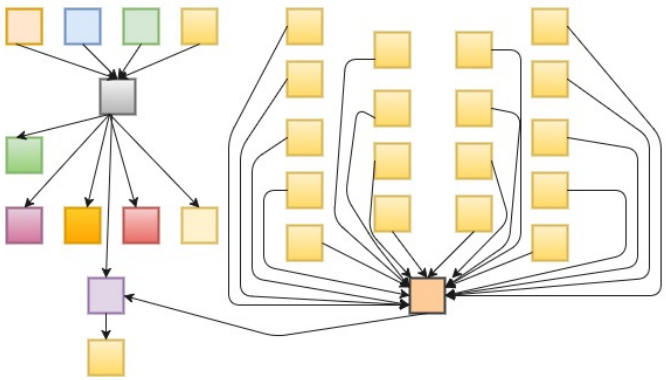


Figure 5.6: A simplified display of the SIPHT workflow [6]

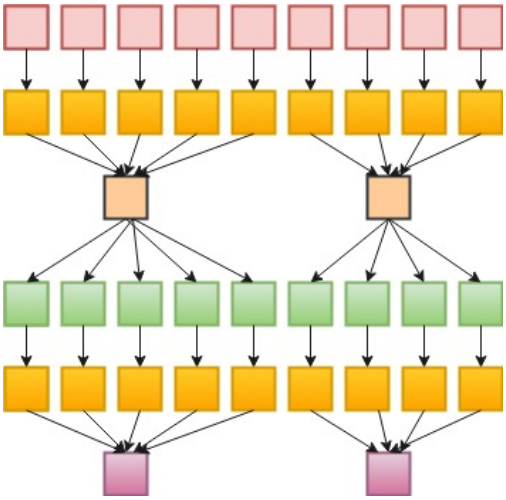


Figure 5.7: A simplified display of LIGO workflow [3]

Resource Model

A cloud model is considered with a data center and different types of VM. The characteristics of VM are modeled similarly as Amazon EC2 instances. The charging period considered for VMs is 60 minutes, as most of the cloud providers. Amazon EC2's pricing scheme is considered as a pricing model because of its widespread use. The on-demand instance of Amazon EC2 with purchasing option is used [175].

Comparative Algorithms

DPO-HEFT is compared with the algorithms described below:

- HEFT: The Heterogeneous Earliest Finish Time heuristic (HEFT) [156] is the most used algorithm for scheduling workflows with shorter makespan but it exhibits certain limitations.
- HEFTBUDG: This algorithm assigns the priorities to the tasks based on the budget allocated per task and then assigns VMs according to the priorities [176].

5.5.3 Results and Discussion

The budget is taken as user input. The limit is set to the budget. The minimum value of the budget is set to 5 virtual machines and maximum budget value is 20 virtual machines. The assumption considered is the maximum number of virtual machines that

a user can access is 20 and the minimum is 5 virtual machines. The makespan of each algorithm is calculated for different workflow applications. Five types of workflows are used to perform the experiments instantiated with a different number of tasks. The results for each scientific workflow application is represented in Figure 5.8, Figure 5.9, Figure 5.10, Figure 5.11 and Figure 5.12.

- **Montage Workflow :** Table 5.5 represents the execution time for Montage workflow using HEFT, HEFTBUDG and DPO-HEFT with varying budget and number of tasks as 25, 50, 100 and 1000. It is depicted from the Table 5.5 that for 25 tasks with the minimum budget the execution time of HEFT and HEFTBUDG is 123.8 seconds, 116.8 seconds respectively while DPO-HEFT completes the execution in 103.79 seconds. Similarly, considering the case with 1000 tasks and 0.75 budget, DPO-HEFT completes the execution in 1920.6 seconds while HEFT and HEFTBUDG complete the execution in 1941.56 seconds and 1928.41 seconds respectively. Hence, DPO-HEFT performs much better than other benchmarking algorithms with minimum time for execution. The number of tasks involved in workflow does not impact the results but the difference in execution time is higher in a small number of tasks in comparison to the larger number of tasks. The execution time results for Montage workflow analyzed from Figure 5.8.

Table 5.5: Execution time for Montage Workflow

	Number of tasks	**Budget: 0.25**	**Budget: 0.5**	**Budget: 0.75**	**Budget: 1.0**
DPO-HEFT	25	103.79	85.72	73.55	54.73
HEFT	25	123.8	114.03	92.63	73.06
HEFTBUDG	25	116.8	98.6	82.5	63.08
DPO-HEFT	50	222.32	154.4	109.77	91.32
HEFT	50	286.12	175.89	130.11	120.05
HEFTBUDG	50	270.45	160.12	118.1	98.65
DPO-HEFT	100	360.54	303.87	202.33	189.45
HEFT	100	380.98	322.14	224.8	208.19
HEFTBUDG	100	375.3	312.2	215.78	196.25
DPO-HEFT	1000	3434.33	2707.92	1920.6	1554.12
HEFT	1000	3454.25	2725.92	1941.56	1601.21
HEFTBUDG	1000	3421.68	2718.1	1928.41	1573.54

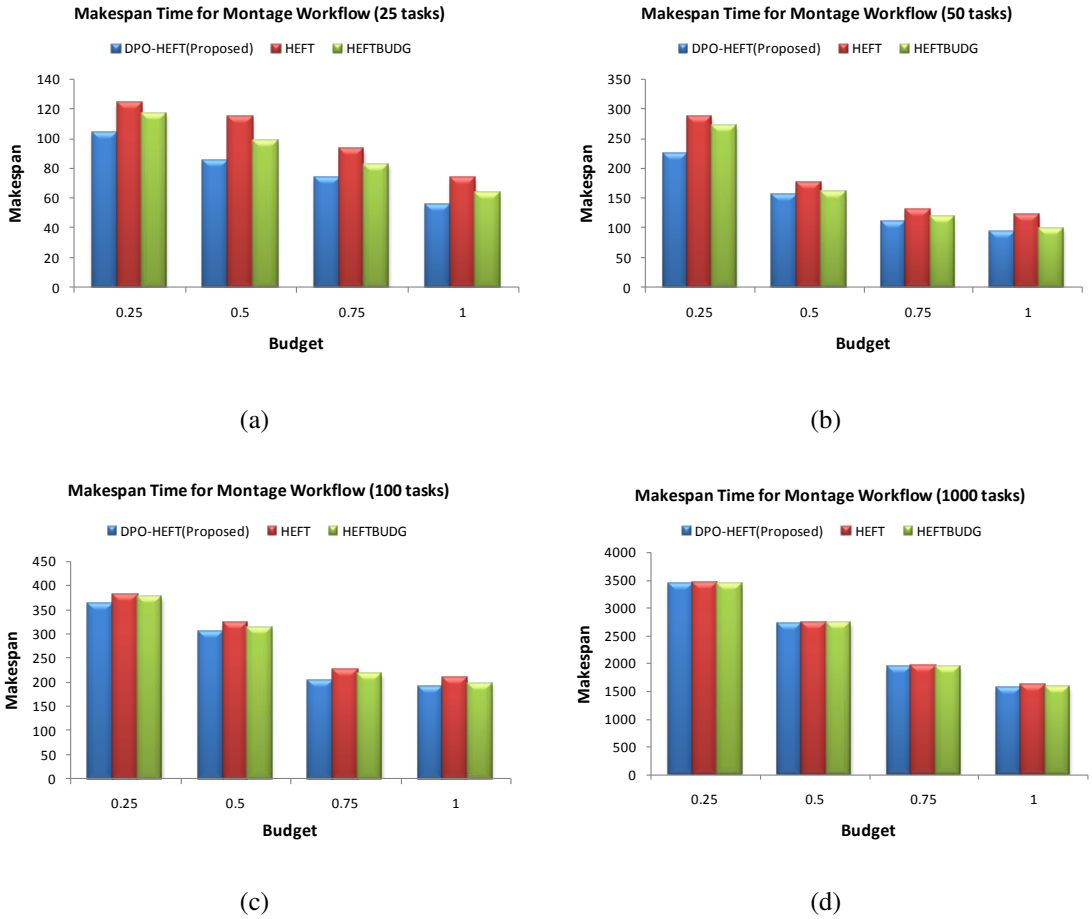
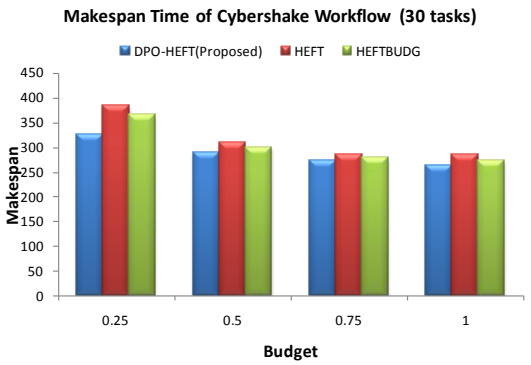


Figure 5.8: Makespan Time of Montage workflow with 25, 50, 100 and 1000 tasks

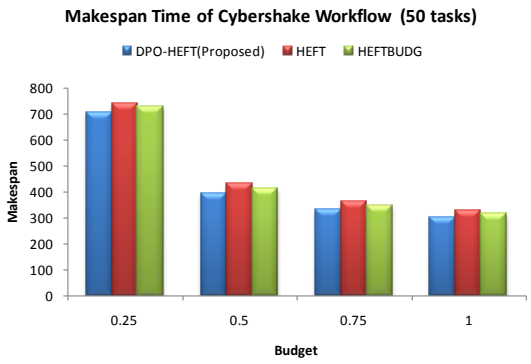
- **Cybershake :** The Table 5.6 represents the execution time for Cybershake workflow using HEFT, HEFTBUDG and DPO-HEFT with varying budget and number of tasks as 25, 50, 100 and 1000. It is depicted from the Table 5.6 that for 25 tasks with the minimum budget the execution time taken by HEFT and HEFTBUDG is 383.57 seconds, 363.87 seconds respectively while DPO-HEFT completes the execution in 323.64 seconds. Similarly, considering the case with 1000 tasks and 0.75 budget, DPO-HEFT completes the execution in 2615 seconds while HEFT and HEFTBUDG complete the execution in 2935 seconds and 2822.5 seconds respectively. Hence DPO-HEFT performs much better than the other state-of-art algorithms with minimum time for execution. The number of tasks involved in workflow does not impact the results but the difference in execution time is higher in a small number of tasks in comparison to the larger number of tasks. The execution time results for Cybershake workflow can be analyzed from Figure 5.9.

Table 5.6: Execution Time for Cybershake Workflow

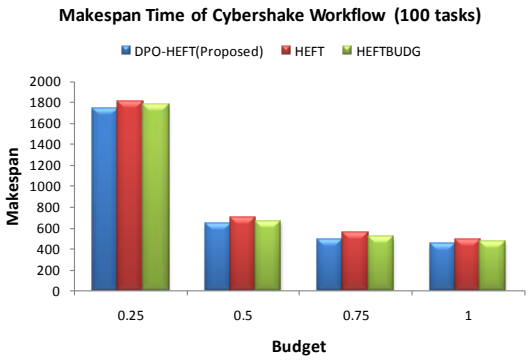
	No. of tasks	Budget: 0.25	Budget: 0.5	Budget: 0.75	Budget: 1
DPO-HEFT	25	323.64	288.03	270.76	262.5
HEFT	25	383.57	307.12	282.57	283.4
HEFTBUDG	25	363.87	299.14	278.45	271.2
DPO-HEFT	50	705.63	393.13	330.99	303
HEFT	50	737.56	432.57	360.98	325.4
HEFTBUDG	50	725.12	410.11	345.51	315.5
DPO-HEFT	100	1741.45	634.5	481.89	440.3
HEFT	100	1801.58	686.37	548.12	487.28
HEFTBUDG	100	1765.15	656.48	503.21	466.57
DPO-HEFT	1000	5354.44	3743.97	2615	2281.07
HEFT	1000	5675.54	3962.14	2935	2599.2
HEFTBUDG	1000	5570.4	3852.5	2822.5	2489.3



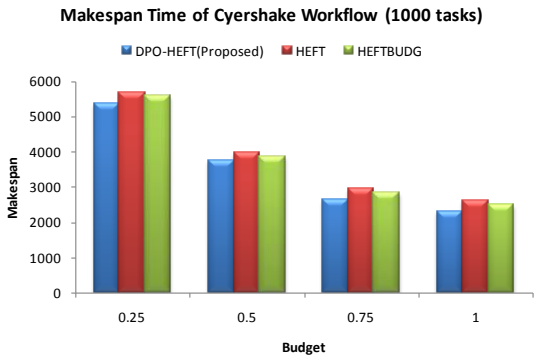
(a)



(b)



(c)



(d)

Figure 5.9: Makespan Time of Cybershake workflow with 30, 50, 100 and 1000 tasks

- **Epigenomics :** The Table 5.7 represents the execution time for Epigenomics workflow using HEFT, HEFTBUDG and DPO-HEFT with varying budget and number of tasks as 24, 46, 100 and 1000. It is depicted from the Table 5.7 that for 24 tasks with the minimum budget the execution time taken by HEFT and HEFTBUDG is 10424.21 seconds, 9813.5 seconds respectively while DPO-HEFT completes the execution in 9403.89 seconds. Similarly, considering the case with 997 tasks and 0.75 budget, DPO-HEFT completes the execution in 1716287.99 seconds while HEFT and HEFTBUDG complete the execution in 1916306.54 seconds and 1816294.53 seconds respectively. Hence DPO-HEFT performs much better than the other state-of-art algorithms with minimum time for execution. The number of tasks involved in workflow does not impact the results but the difference in execution time is higher in a small number of tasks in comparison to a larger number of tasks. The execution time results for Epigenomics workflow can be analyzed from Figure 5.10.

Table 5.7: Execution Time for Epigenomics Workflow

	Number of tasks	**Budget : 0.25**	**Budget : 0.5**	**Budget : 0.75**	**Budget : 1**
DPO-HEFT	24	9403.89	7608.45	5806.6	5585.93
HEFT	24	10424.21	8027.14	6525.9	5903.79
HEFTBUDG	24	9813.5	7817.8	6217.23	5792.57
DPO-HEFT	46	27578.87	15677.9	7927	7737.92
HEFT	46	31597.57	17696.14	9047.98	8656.21
HEFTBUDG	46	29582.5	16666.31	8839.41	8449.78
DPO-HEFT	100	150154.2	122402.4	82951.06	67904.16
HEFT	100	160175.61	129420.34	88977.27	73922.3
HEFTBUDG	100	152965.43	126413.89	85966.89	70915.84
DPO-HEFT	997	2023852.36	1844564.02	1716287.99	1548963.73
HEFT	997	2423873.15	2044583.21	1916306.54	1748982.4
HEFTBUDG	997	2223860.58	1944572.41	1816294.53	1648992.21

- **LIGO :** The Table 5.8 represents the execution time for LIGO workflow using HEFT, HEFTBUDG and DPO-HEFT with varying budget and number of tasks as 30, 50, 100 and 1000. It is depicted from the Table 5.8 that for 30 tasks with the

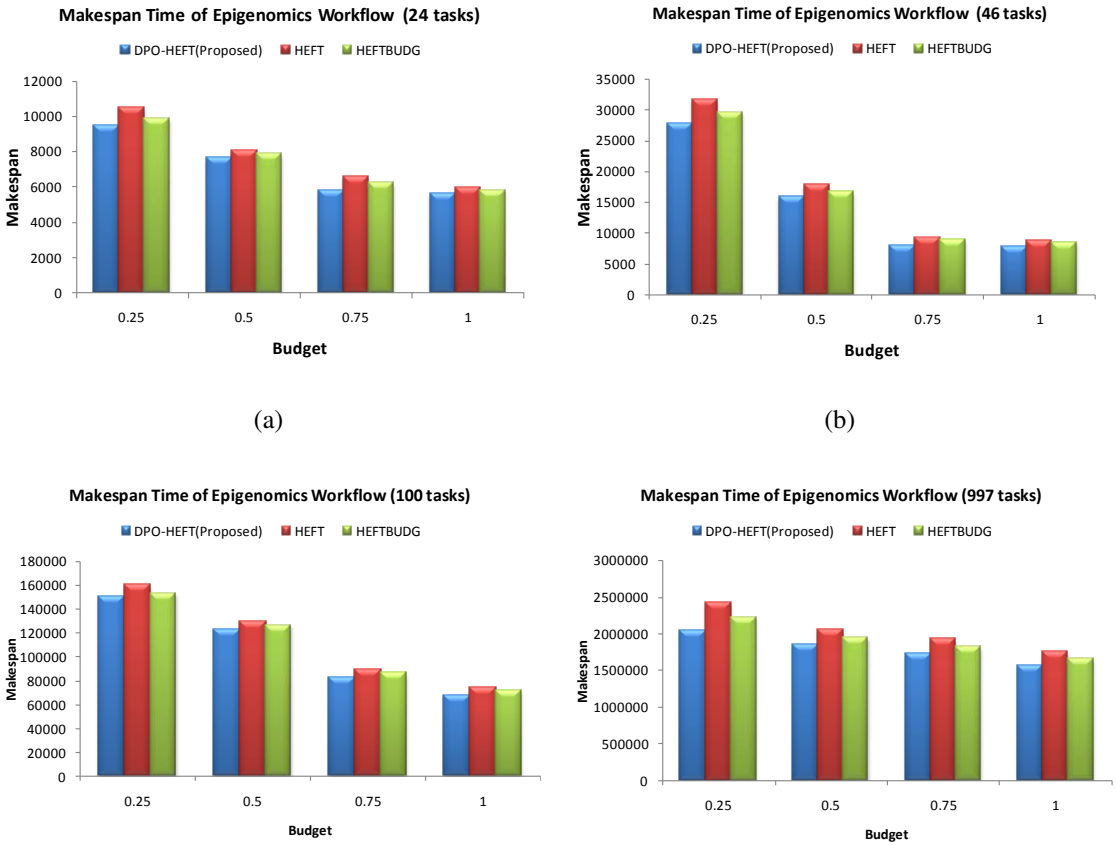


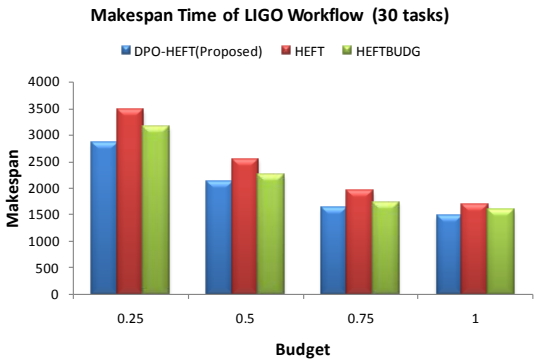
Figure 5.10: Makespan Time of Epigenomics workflow with 24, 46, 100 and 997 tasks

minimum budget, the execution time taken by HEFT and HEFTBUDG is 3458.96 seconds, 3144.51 seconds respectively while DPO-HEFT completes the execution in 2837.52 seconds. Similarly, considering the case with 1000 tasks and 0.75 budget, DPO-HEFT completes the execution in 98336.44 seconds while HEFT and HEFTBUDG complete the execution in 92567.74 seconds and 88055.87 seconds respectively. Hence, DPO-HEFT performs better than existing algorithms with minimum time for execution. The number of tasks involved in workflow does not impact the results but the difference in execution time is higher in a small number of tasks in comparison to a larger number of tasks. The execution time results for LIGO workflow can be analyzed from Figure 5.11.

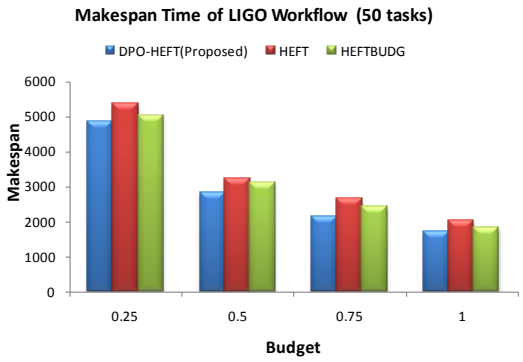
- **SIPHT:** The Table 5.9 represents the execution time for SIPHT workflow using HEFT, HEFTBUDG and DPO-HEFT with varying budget and number of tasks as 30, 60, 100 and 1000.

Table 5.8: Execution Time for LIGO Workflow

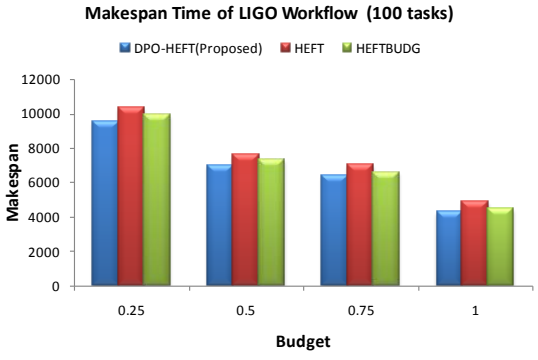
	No. of tasks	Budget: 0.25	Budget: 0.5	Budget: 0.75	Budget: 1
DPO-HEFT	30	2837.52	2100.58	1614.27	1461.22
HEFT	30	3458.96	2529.65	1933.74	1683.41
HEFTBUDG	30	3144.51	2218.74	1721.45	1572.54
DPO-HEFT	50	4821.6	2780.93	2121.65	1692.87
HEFT	50	5342.31	3205.54	2642.82	2013.87
HEFTBUDG	50	5033.57	3091.02	2394.81	1802.08
DPO-HEFT	100	9463.91	6962.74	6372.76	4236.44
HEFT	100	10282.5	7582.45	6994.52	4853.71
HEFTBUDG	100	9874.21	7270.95	6581.41	4445.77
DPO-HEFT	1000	142064.65	109661.45	98336.44	84041.83
HEFT	1000	164094.67	129383.41	113367.54	92567.74
HEFTBUDG	1000	148064.42	119673.43	101355.62	88055.87



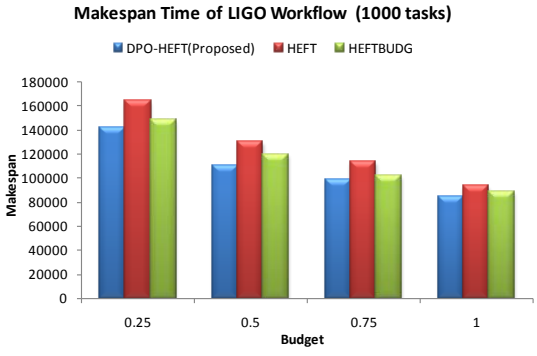
(a)



(b)



(c)



(d)

Figure 5.11: Makespan Time of LIGO workflow with 30, 50, 100 and 1000 tasks

It is depicted from the Table 5.9 that for 30 tasks with the minimum budget the execution time taken by HEFT and HEFTBUDG is 4514.57 seconds, 4505.21 seconds respectively while DPO-HEFT completes the execution in 4492.28 seconds. Similarly, considering the case with 1000 tasks and 0.75 budget, DPO-HEFT completes the execution in 28263.89 seconds while HEFT and HEFTBUDG complete the execution in 32324.96 seconds and 30364.71 seconds respectively. Hence DPO-HEFT performs much better than the other state-of-art algorithms with minimum time for execution. The number of tasks involved in workflow does not impact the results but the difference in execution time is higher in a small number of tasks in comparison to a larger number of tasks. The execution time results for SIPHT workflow can be analyzed from Figure 5.12.

Table 5.9: Execution Time for SIPHT Workflow

	No. of tasks	**Budget: 0.25**	**Budget: 0.5**	**Budget: 0.75**	**Budget: 1**
DPO-HEFT	30	4492.28	4504.61	4498.01	4421.09
HEFT	30	4514.57	4523.89	4517.91	4442.74
HEFTBUDG	30	4505.21	4515.84	4509.58	4431.69
DPO-HEFT	60	5325.77	5207.49	4967.06	4735.73
HEFT	60	5348.93	5228.47	4988.08	4756.38
HEFTBUDG	60	5332.8	5219.91	4976.31	4745.1
DPO-HEFT	100	8121.7	5197.97	5008.16	4511.81
HEFT	100	10112.66	5818.67	5827.44	5032.29
HEFTBUDG	100	9547.53	5714.75	5617.61	4821.77
DPO-HEFT	1000	48640.62	44715.91	28263.89	25744.43
HEFT	1000	51584.71	50586.45	32324.96	30325.79
HEFTBUDG	1000	49685.87	48554.53	30364.71	28664.29

The results depict that in case of any scientific workflow application while fixing the initial budget, DPO-HEFT is performing better than other algorithms in terms of makespan of the workflow. The schedules available for five workflows Montage, LIGO, Epigenomics, SIPHT and Cybershake obtained by DPO-HEFT results in minimum makespan, whether the number of tasks is minimum or maximum. The proposed algorithm DPO-HEFT manages to achieve smaller makespan using fewer VMs in comparison to HEFT and HEFTBUDG.

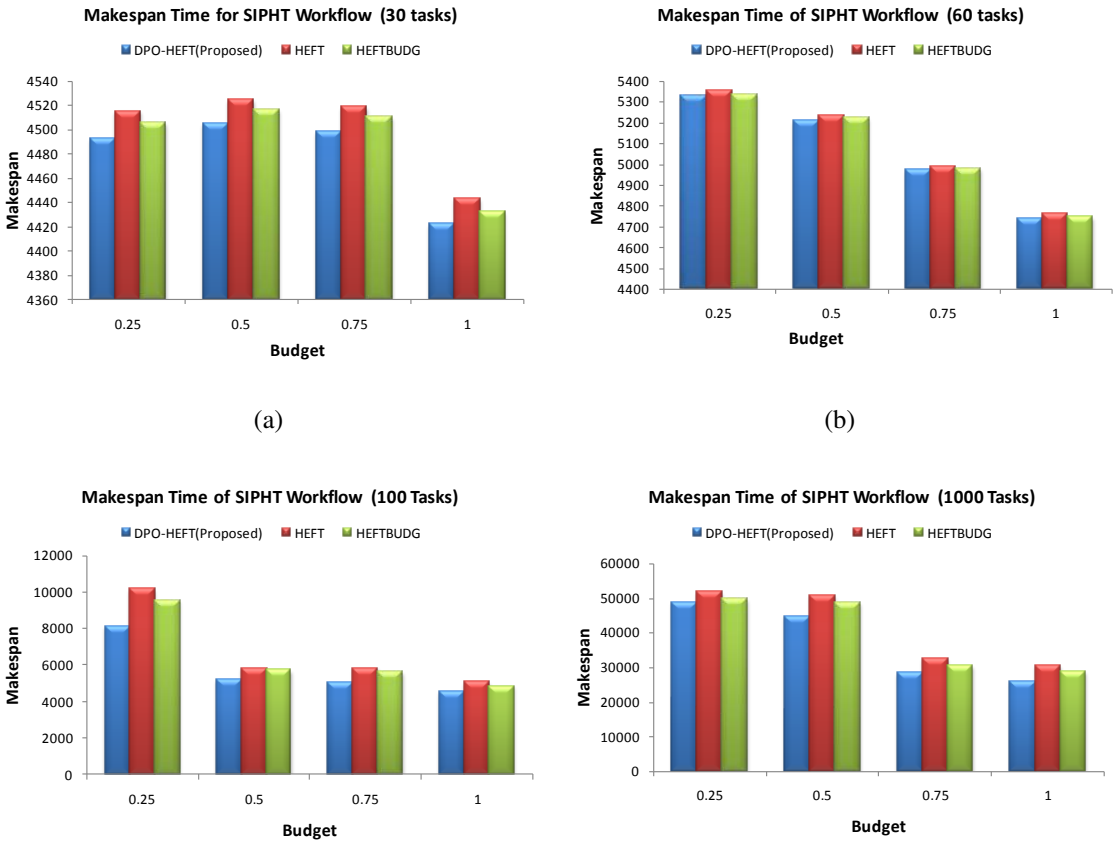


Figure 5.12: Makespan Time of SIPHT workflow with 30, 60, 100 and 1000 tasks

The structure of the tasks in workflow also affects the makespan of scheduling workflow with a budget constraint. As in MONTAGE and CYBERSHAKE workflows, a large number of initial tasks are involved. For each initial task, the amount of work involves the same magnitude. Due to the nearby structure of a task bag of LIGO, there is hardly any improvement. The Epigenomics workflow exhibit more makespan in comparison to other workflow application due to more parallelization required. Hence, the type of structure of workflow also impacts the performance.

5.6 Summary

In this chapter, a scheduling algorithm is presented with budget and deadline trade-off to schedule scientific workflows. This technique contemplates the task clustering and data placement strategy. These platforms enable multiple types of VMs with different cost and speed parameters to dynamically enrolled. DPO-HEFT is an extension of the

HEFT scheduling algorithm. The proposed algorithm manages to find a solution while enforcing the prescribed budget and smaller makespan in comparison to the baseline versions. DPO-HEFT outperforms over HEFTBUDG, HEFT especially for workflows with a non-trivial interdependence graph. The proposed algorithm achieved a better makespan and is able to find a better schedule.

CHAPTER 6

CONCLUSION AND FUTURE DIRECTIONS

In this chapter, the research work presented in the thesis is summarized. The contributions are outlined with future research directions.

6.1 Conclusion and Discussion

In this thesis, the schemes and algorithms for clustering and scheduling scientific workflows are explored to meet different functional and quality of service requirements. In this work, the set of algorithms are designed and implemented to allow scientific workflows to be distributed among different cloud data centers in order to meet functional requirements such as monetary cost, budget and deadline.

Chapter 2 presents an overview of deploying scientific workflow on the cloud. In the literature review, different task clustering, data placement, and scheduling algorithms are discussed. However, scarcity of research exists that considers planning and deploying the scientific workflow application in a cloud environment with different QoS parameters. To bridge this gap, these problems are addressed with different proposed algorithms.

Chapter 3 describes a proposed algorithm for task clustering in scientific workflows. The scientific workflows in cloud computing exhibit different kinds of structure. There is no efficient technique available to consider the structure of the workflow and perform the execution. In this chapter, the researcher proposed Hybrid Balanced (HYB) task clustering algorithm for scientific workflows that reduces the makespan of workflow execution and thus avoids wastage of resources. The proposed approach merges the tasks with similar impact factor in the pipeline. An experiment is conducted for evaluation of the proposed algorithm in comparison to four clustering methods: Horizontal Runtime Balancing (HRB), Horizontal Clustering (HC), Horizontal Distance Balancing (HDB), and Horizontal Impact Factor Balancing (HIFB). The experiment aimed to evaluate execution time improvement. The results depict that the proposed clustering method is able to perform better than the existing clustering algorithms for scientific workflow. Therefore, it provides the benefit to reduce the scheduling overhead for different workflow structures.

Chapter 4 proposed an algorithm for the placement of intermediate datasets generated during the execution of the workflow. Different characteristics of scientific cloud workflows are evaluated. The data placement technique based on Crow Search Algorithm (CSA) has been designed that automatically allocate the intermediate datasets between data centers based on the factor of dependency with data centers. Experiments performed to reveal that the proposed data placement technique reduces the movement of datasets during workflow execution. The data movements also decrease in spite of having a certain percentage of fixed location datasets in the workflow.

Chapter 5 proposed Data Placement Oriented Heterogeneous Earliest Finish Time (DPO-HEFT) algorithm to schedule scientific workflows on Cloud with data placement and task clustering. Scheduling of scientific workflows on cloud is a crucial activity. DPO-HEFT is an extension of HEFT and is able to find a solution of deadline and budget constraints while enforcing the prescribed QoS. The makespan for the scientific workflow is improved by DPO-HEFT in comparison to other state-of-art algorithms especially for workflows with a non-trivial interdependence graph. The proposed algorithm achieved a better makespan and able to find a better resource for the workflow task to be executed.

All in all, the proposed framework helps the scientists to deploy the scientific workflows on cloud in an optimal manner and provide budget and deadline parameters.

6.2 Future Scope

This thesis has investigated the approach for task clustering, data placement and scheduling of scientific workflows in a cloud environment. The developed approaches can be extended in different ways by other higher level optimization algorithms. The following sections describe important future directions in this field.

6.2.1 Energy-Efficient Placement and Scheduling

The carbon footprints and a large amount of energy consumption is the major issue in the cloud data centers. The regular increase in electricity charges also arises the need to design energy efficient techniques for cloud applications. Scientific workflow scheduling consumes a large number of resources. Thus a solution needs to provide which select energy efficient data center on priority for data placement and scheduling.

6.2.2 Replication Management

The replication technique placed the same data sets on more than one data center, this help to increase the data available on more than one location in time-critical applications. This technique has the potential to be a future direction for data placement, which further decreases the movement of data. In the future, the bandwidth of data centers can also be considered which can further increase the effectiveness of the strategy, the number of parameters can be added to make strategy cost-effective.

6.2.3 Fault-tolerance and Provenance Management

Scheduling approach can be extended with provenance technique. Cloud infrastructure allows tasks to be interrupted and re-scheduled. Regular monitoring the health of schedule task insist to re-schedule the task. This could be possible to terminate the task on current VM and restart on VM with good performance. Such dynamic decisions also involve budgetary risks. For example, deriving timeouts from execution is a challenging issue, but the online design heuristics with maximum likelihood can reduce the

final makespan while taking into consideration of initial budget constraint. Fault tolerance based scheduling heuristic can be developed to make the execution of scientific workflow seamless.

6.2.4 Pricing Models

Cloud offers different pricing models such as reserved, on-demand and spot instances. The reserved instances are taking the fixed resources for a long period e.g. 1 year. The cloud providers are giving high benefit on this policy. On-demand resources are scale-up or scale-down the resources at the run-time requirement of the applications. Spot-instances are the spare capacity form reserved resources. They are much cheaper as compare to on-demand and reserved resources. The highly cost-efficient planning algorithms could be designed by mixing the resources with different pricing models.

The cloud providers offer different pricing window such as per hour or per minute. This mode directly affects the total deployment and running cost of the workflow. The techniques are required to select the appropriate pricing policy form multi-cloud providers is another possible future direction.

REFERENCES

[1] C. Lin, S. Lu, Z. Lai, A. Chebotko, X. Fei, J. Hua, and F. Fotouhi, "Servic-Oriented Architecture for VIEW: a Visual Scientific Workflow Management System," *IEEE International Conference*, pp. 335–342, 2008.

[2] D. Hollingsworth and U. Hampshire, "Workflow management coalition: The workflow reference model," *Document Number TC00-1003*, vol. 19, 1995.

[3] G. Berriman, A. Laity, J. Good, J. Jacob, D. Katz, E. Deelman, G. Singh, M. Su, and T. Prince, "Montage: The architecture and scientific applications of a national virtual observatory service for computing astronomical image mosaics," in *Proceedings of Earth Sciences Technology Conference*, 2006.

[4] R. Graves, T. H. Jordan, S. Callaghan, E. Deelman, E. Field, G. Juve, C. Kesselman, P. Maechling, G. Mehta, K. Milner, *et al.*, "Cybershake: a physics-based seismic hazard model for southern california," *Pure and Applied Geophysics*, vol. 168, no. 3-4, pp. 367–381, 2011.

[5] G. Juve and E. Deelman, "Scientific workflows in the cloud," in *Grids, clouds and virtualization*, pp. 71–91, Springer, 2011.

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

[7] A. Abramovici, W. E. Althouse, R. W. Drever, Y. Gürsel, S. Kawamura, F. J. Raab, D. Shoemaker, L. Sievers, R. E. Spero, K. S. Thorne, *et al.*, "Ligo: The laser interferometer gravitational-wave observatory," *Science*, vol. 256, no. 5055, pp. 325–333, 1992.

[8] X. Liu and A. Datta, "Towards intelligent data placement for scientific workflows in collaborative cloud environment," in *Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 2011 IEEE International Symposium on*, pp. 1052–1061, IEEE, 2011.

[9] F. International and J. Conference, "2009 Fifth International Joint Conference on INC , IMS and IDC," pp. 44–51, 2009.

[10] I. Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud computing and grid computing 360-degree compared," *Grid Computing Environments Workshop 2008 (GCE '08)*, pp. 1–10, 2008.

[11] N. Sclater, "elearning in the cloud," *International Journal of Virtual and Personal Learning Environments (IJVPLE)*, vol. 1, no. 1, pp. 10–19, 2010.

[12] K. Keahey, P. Armstrong, J. Bresnahan, D. LaBissoniere, and P. Riteau, "Infrastructure outsourcing in multi-cloud environment," in *Proceedings of the 2012 workshop on Cloud services, federation, and the 8th open cirrus summit*, pp. 33–38, ACM, 2012.

[13] L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, "A break in the clouds," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 1, p. 50, 2008.

[14] D. Hollingsworth, "Workflow Management Coalition The Workflow Reference Model," no. 1, pp. 1–55, 1995.

[15] L. Ramakrishnan and B. Plale, "A multi-dimensional classification model for scientific workflow characteristics," *Proceedings of the 1st International Workshop on Workflow Approaches to New Data-centric Science - Wands '10*, pp. 1–12, 2010.

[16] WfMC, "Workflow Management Coalition Terminology {&} Glossary," *Management*, vol. 39, no. 3, pp. 1–65, 1999.

[17] E. Deelman, G. Singh, M.-h. Su, J. Blythe, Y. Gil, C. Kesselman, G. B. Berriman, J. Good, A. Laity, J. C. Jacob, D. S. Katz, a. Gil, G. Mehta, and K. Vahi, "Pegasus: a framework for mapping complex scientific workflows onto distributed systems," *Scientific Programming Journal*, vol. 13, no. January, pp. 219–237, 2005.

[18] D. Hull, K. Wolstencroft, R. Stevens, C. Goble, M. R. Pocock, P. Li, and T. Oinn, "Taverna: A tool for building and running workflows of services," *Nucleic Acids Research*, vol. 34, no. WEB. SERV. ISS., pp. 729–732, 2006.

[19] D. Warneke and O. Kao, "Nephele: Efficient Parallel Data Processing in the Cloud," *Proceedings of the 2nd Workshop on ManyTask Computing on Grids and Supercomputers MTAGS 09 November 2009 Portland OR USA*, pp. 1–10, 2009.

[20] P. Bientinesi, R. Iakymchuk, and J. Napper, "Handbook of Cloud Computing (2010)," *Handbook of Cloud Computing*, pp. 493–516, 2010.

[21] D. De Oliveira, E. Ogasawara, F. Baiao, and M. Mattoso, "SciCumulus: A lightweigh cloud middleware to explore many task computing paradigm in scientific workflows," *Proceedings - 2010 IEEE 3rd International Conference on Cloud Computing, CLOUD 2010*, pp. 378–385, 2010.

[22] D. Oliveira, "Similarity-based Workflow Clustering," no. July 2016, 2010.

[23] Y. Wei, K. Sukumar, C. Vecchiola, D. Karunamoorthy, and R. Buyya, "Chapter 27 Aneka Cloud Application Platform and Its Integration with Windows Azure," pp. 1–30.

[24] W. Chen, M. Rey, and M. Rey, "Fault Tolerant Clustering in Scientific Workflows," 2012.

[25] W. Chen, R. Ferreira, E. Deelman, and R. Sakellariou, "Balanced Task Clustering in Scientific Workflows," pp. 1–8, 2013.

[26] W. Liu, S. Peng, W. Du, and W. Wang, "Security-aware intermediate data placement strategy in scientific cloud workflows," 2014.

[27] X. Li, Y. Wu, F. Ma, E. Zhu, F. Wang, L. Wu, and Y. Yang, "A New Particle Swarm Optimization-Based Strategy for Cost-Effective Data Placement in Scientific," pp. 115–120, 2014.

[28] M. M. Zhu and C. Q. Wu, "Energy-Efficient Resource Management for Scientific Workflows in Clouds," pp. 402–409, 2014.

[29] E. Le Sueur and G. Heiser, "Dynamic voltage and frequency scaling: The laws of diminishing returns," in *Proceedings of the 2010 international conference on Power aware computing and systems*, pp. 1–8, 2010.

[30] D. Andresen, T. Yang, V. Holmedahl, and O. H. Ibarra, "Sweb: Towards a scalable world wide web server on multicomputers," in *Parallel Processing Symposium, 1996., Proceedings of IPPS'96, The 10th International*, pp. 850–856, IEEE, 1996.

[31] J. Wang, P. Korambath, I. Altintas, J. Davis, and D. Crawl, "Workflow as a Service in the cloud: Architecture and scheduling algorithms," *Procedia Computer Science*, vol. 29, pp. 546–556, 2014.

[32] W. Chen, R. Ferreira, E. Deelman, and R. Sakellariou, "Using imbalance metrics to optimize task clustering in scientific workflow executions," *Future Generation Computer Systems*, vol. 46, pp. 69–84, 2015.

[33] J. Sahni and D. P. Vidyarthi, "Workflow-and-Platform Aware task clustering for scientific workflow execution in Cloud environment," 2016.

[34] Z. Zhou, Z. Cheng, L.-j. Zhang, W. Gaaloul, and K. Ning, "Scientific Workflow Clustering and Recommendation Leveraging Layer Hierarchical Analysis," vol. X, no. X, pp. 1–14, 2016.

[35] Q. Zhao, C. Xiong, and P. Wang, "Heuristic data placement for data-intensive applications in heterogeneous cloud," *Journal of Electrical and Computer Engineering*, vol. 2016, 2016.

[36] T. Wang, S. Yao, Z. Xu, and S. Jia, "Dccp: an effective data placement strategy for data-intensive computations in distributed cloud computing systems," *The Journal of Supercomputing*, vol. 72, no. 7, pp. 2537–2564, 2016.

[37] T. Kosar and M. Livny, "A framework for reliable and efficient data placement in distributed computing systems," *Journal of Parallel and Distributed Computing*, vol. 65, no. 10, pp. 1146–1157, 2005.

[38] S. Doraimani and A. Iamnitchi, "File grouping for scientific data management: lessons from experimenting with real traces," in *Proceedings of the 17th international symposium on High performance distributed computing*, pp. 153–164, ACM, 2008.

[39] G. Fedak, H. He, and F. Cappello, "Bitdew: a programmable environment for large-scale data management and distribution," in *High Performance Computing, Networking, Storage and Analysis, 2008. SC 2008. International Conference for*, pp. 1–12, IEEE, 2008.

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

[41] Z. Er-Dun, Q. Yong-Qiang, X. Xing-Xing, and C. Yi, "A data placement strategy based on genetic algorithm for scientific workflows," in *Computational Intelligence and Security (CIS), 2012 Eighth International Conference on*, pp. 146–149, IEEE, 2012.

[42] W. Guo and X. Wang, "A data placement strategy based on genetic algorithm in cloud computing platform," in *Web Information System and Application Conference (WISA), 2013 10th*, pp. 369–372, IEEE, 2013.

[43] P. Zheng, L.-Z. Cui, H.-Y. Wang, and M. Xu, "A data placement strategy for data-intensive applications in cloud," *Jisuanji Xuebao(Chinese Journal of Computers)*, vol. 33, no. 8, pp. 1472–1480, 2010.

[44] T. Back, *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford university press, 1996.

[45] M. Mitchell, *An introduction to genetic algorithms*. MIT press, 1998.

[46] M. Poess and R. O. Nambiar, "Tuning servers, storage and database for energy efficient data warehouses," in *Data Engineering (ICDE), 2010 IEEE 26th International Conference on*, pp. 1006–1017, IEEE, 2010.

[47] A. Beckmann, U. Meyer, P. Sanders, and J. Singler, "Energy-efficient sorting using solid state disks," *Sustainable Computing: Informatics and Systems*, vol. 1, no. 2, pp. 151–163, 2011.

[48] L. Rao, X. Liu, L. Xie, and W. Liu, "Minimizing electricity cost: Optimization of distributed internet data centers in a multi-electricity-market environment," in *INFOCOM, 2010 Proceedings IEEE*, pp. 1–9, IEEE, 2010.

[49] W. Lang and J. Patel, "Towards eco-friendly database management systems," *arXiv preprint arXiv:0909.1767*, 2009.

[50] E. M. Elnozahy, M. Kistler, and R. Rajamony, "Energy-efficient server clusters," in *International Workshop on Power-Aware Computer Systems*, pp. 179–197, Springer, 2002.

[51] Y. Xiao, J. Wang, Y. Li, and H. Gao, "An energy-efficient data placement algorithm and node scheduling strategies in cloud computing systems," in *Proc. of the 2nd Int'l Conf. on Advances in Computer Science and Engineering (CSE 2013). Paris: Atlantis Press*, vol. 63, 2013.

[52] E. Pinheiro and R. Bianchini, "Energy conservation techniques for disk array-based servers," in *ACM International Conference on Supercomputing 25th Anniversary Volume*, pp. 369–379, ACM, 2014.

[53] N. Maheshwari, R. Nanduri, and V. Varma, "Dynamic energy efficient data placement and cluster reconfiguration algorithm for mapreduce framework," *Future Generation Computer Systems*, vol. 28, no. 1, pp. 119–127, 2012.

[54] B. Li, J. Li, J. Huai, T. Wo, Q. Li, and L. Zhong, "Enacloud: An energy-saving application live placement approach for cloud computing environments," in *Cloud Computing, 2009. CLOUD'09. IEEE International Conference on*, pp. 17–24, IEEE, 2009.

[55] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," *Future generation computer systems*, vol. 28, no. 5, pp. 755–768, 2012.

[56] P.-Y. Yin, S.-S. Yu, P.-P. Wang, and Y.-T. Wang, "A hybrid particle swarm optimization algorithm for optimal task assignment in distributed systems," *Computer Standards & Interfaces*, vol. 28, no. 4, pp. 441–450, 2006.

[57] L. Guo, Z. He, S. Zhao, N. Zhang, J. Wang, and C. Jiang, "Multi-objective optimization for data placement strategy in cloud computing," in *International Conference on Information Computing and Applications*, pp. 119–126, Springer, 2012.

[58] X. Li, Y. Wu, F. Ma, E. Zhu, F. Wang, L. Wu, and Y. Yang, "A new particle swarm optimization-based strategy for cost-effective data placement in scientific cloud workflows," in *Future Information Technology*, pp. 115–120, Springer, 2014.

[59] L. Guo, S. Zhao, S. Shen, and C. Jiang, "Task scheduling optimization in cloud computing based on heuristic algorithm.," *JNW*, vol. 7, no. 3, pp. 547–553, 2012.

[60] J. M. Cope, N. Trebon, H. M. Tufo, and P. Beckman, "Robust data placement in urgent computing environments," in *Parallel & Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on*, pp. 1–13, IEEE, 2009.

[61] S. Pandey, L. Wu, S. M. Guru, and R. Buyya, "A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments," in *Advanced information networking and applications (AINA), 2010 24th IEEE international conference on*, pp. 400–407, IEEE, 2010.

[62] W. Liu, S. Peng, W. Du, W. Wang, and G. S. Zeng, "Security-aware intermediate data placement strategy in scientific cloud workflows," *Knowledge and information systems*, vol. 41, no. 2, pp. 423–447, 2014.

[63] S. Agarwal, J. Dunagan, N. Jain, S. Saroiu, A. Wolman, and H. Bhogan, "Volley: Automated data placement for geo-distributed cloud services.," in *NSDI*, vol. 10, pp. 28–0, 2010.

[64] R. Chaiken, B. Jenkins, P.-Å. Larson, B. Ramsey, D. Shakib, S. Weaver, and J. Zhou, "Scope: easy and efficient parallel processing of massive data sets," *Proceedings of the VLDB Endowment*, vol. 1, no. 2, pp. 1265–1276, 2008.

[65] Ü. V. Çatalyürek, K. Kaya, and B. Uçar, "Integrated data placement and task assignment for scientific workflows in clouds," in *Proceedings of the fourth international workshop on Data-intensive distributed computing*, pp. 45–54, ACM, 2011.

[66] Ü. Çatalyürek and C. Aykanat, "Patoh (partitioning tool for hypergraphs)," in *Encyclopedia of Parallel Computing*, pp. 1479–1487, Springer, 2011.

[67] C. M. Fiduccia and R. M. Mattheyses, "A linear-time heuristic for improving network partitions," in *Papers on Twenty-five years of electronic design automation*, pp. 241–247, ACM, 1988.

[68] S. Pandey, K. K. Gupta, A. Barker, and R. Buyya, "Minimizing cost when using globally distributed cloud services: A case study in analysis of intrusion detection workflow application," *Cloud Computing and Distributed Systems Laboratory, The University of Melbourne, Australia, Melbourne, Australia, Tech. Rep*, 2009.

[69] A. Kaur and P. Gupta, "A data placement strategy based on crow search algorithm in cloud computing," *Recent Patents on Computer Science*, vol. 12, 2019.

[70] Q. Li, K. Wang, S. Wei, X. Han, L. Xu, and M. Gao, "A data placement strategy based on clustering and consistent hashing algorithm in cloud computing," in *Communications and Networking in China (CHINACOM), 2014 9th International Conference on*, pp. 478–483, IEEE, 2014.

[71] Z. Chedrawy and S. S. R. Abidi, "An intelligent knowledge sharing strategy featuring item-based collaborative filtering and case based reasoning," in *Intelligent Systems Design and Applications, 2005. ISDA'05. Proceedings. 5th International Conference on*, pp. 67–72, IEEE, 2005.

[72] S. Bharathi and A. Chervenak, "Scheduling data-intensive workflows on storage constrained resources," in *Proceedings of the 4th Workshop on Workflows in Support of Large-Scale Science*, p. 3, ACM, 2009.

[73] Q. Wei, B. Veeravalli, B. Gong, L. Zeng, and D. Feng, "Cdrm: A cost-effective dynamic replication management scheme for cloud storage cluster," in *Cluster Computing (CLUSTER), 2010 IEEE International Conference on*, pp. 188–196, IEEE, 2010.

[74] Y. Ye, L. Xiao, I.-L. Yen, and F. Bastani, "Cloud storage design based on hybrid of replication and data partitioning," in *Parallel and Distributed Systems (ICPADS), 2010 IEEE 16th International Conference on*, pp. 415–422, IEEE, 2010.

[75] J. Myint and T. T. Naing, "A data placement algorithm with binary weighted tree on pc cluster-based cloud storage system," in *Cloud and Service Computing (CSC), 2011 International Conference on*, pp. 315–320, IEEE, 2011.

[76] X. Huang, Y. X. Peng, and P. F. You, "Data placement and query for cloud computing based on myheawood network," in *Applied Mechanics and Materials*, vol. 543, pp. 3100–3104, Trans Tech Publ, 2014.

[77] S. Pandey and R. Buyya, "Scheduling workflow applications based on multi-source parallel data retrieval in distributed computing networks," *The Computer Journal*, vol. 55, no. 11, pp. 1288–1308, 2012.

[78] I. Casas, J. Taheri, R. Ranjan, L. Wang, and A. Y. Zomaya, "A balanced scheduler with data reuse and replication for scientific workflows in cloud computing systems," *Future Generation Computer Systems*, 2016.

[79] D.-K. Kang, S.-H. Kim, C.-H. Youn, and M. Chen, "Cost adaptive workflow scheduling in cloud computing," *Proceedings of the 8th International Conference on Ubiquitous Information Management and Communication*, pp. 1–8, 2014.

[80] M. A. Rodriguez and R. Buyya, "Deadline Based Resource Provisioning and Scheduling Algorithm for Scientific Workflows on Clouds,"

[81] Y. Shi and R. C. Eberhart, "Empirical study of particle swarm optimization," in *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, vol. 3, IEEE, 1999.

[82] S.-C. Tai, C. Lai, and Y.-C. Lin, "Two fast nearest neighbor searching algorithms for image vector quantization," *Communications, IEEE Transactions on*, vol. 44, no. 12, pp. 1623–1628, 1996.

[83] L. Singh and S. Singh, "A survey of workflow scheduling algorithms and research issues," *International Journal of Computer Applications*, vol. 74, no. 15, 2013.

[84] X. Li, J. Song, and B. Huang, "A scientific workflow management system architecture and its scheduling based on cloud service platform for manufacturing big data analytics," *The International Journal of Advanced Manufacturing Technology*, 2015.

[85] C. Q. Wu, X. Lin, D. Yu, W. Xu, and L. Li, "End-to-End Delay Minimization for Scientific Workflows in Clouds under Budget Constraint,"

[86] Q. Zhao, C. Xiong, and P. Wang, "Heuristic Data Placement for Data-Intensive Applications in Heterogeneous Cloud," vol. 2016, 2016.

[87] P. Bryk, M. Malawski, and G. Juve, "Storage-aware Algorithms for Scheduling of Workflow Ensembles in Clouds," *Journal of Grid Computing*, pp. 359–378, 2016.

[88] M. Ebrahimi, A. Mohan, S. Lu, and A. Kashlev, "BDAP : A Big Data Placement Strategy for Cloud- Based Scientific Workflows,"

[89] I. Casas, J. Taheri, R. Ranjan, L. Wang, and A. Y. Zomaya, "A balanced scheduler with data reuse and replication for scientific workflows in cloud computing systems," *Future Generation Computer Systems*, 2016.

[90] Z. Li, J. Ge, H. Yang, L. Huang, H. Hu, H. Hu, and B. Luo, "A security and cost aware scheduling algorithm for heterogeneous tasks of scientific workflow in clouds," *Future Generation Computer Systems*, 2016.

[91] B. Ludäscher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger, M. Jones, E. A. Lee, J. Tao, and Y. Zhao, "Scientific workflow management and the kepler system," *Concurrency and Computation: Practice and Experience*, vol. 18, no. 10, pp. 1039–1065, 2006.

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

[93] C. Hoffa, G. Mehta, T. Freeman, E. Deelman, K. Keahey, B. Berriman, and J. Good, "On the use of cloud computing for scientific workflows," in *2008 IEEE fourth international conference on eScience*, pp. 640–645, IEEE, 2008.

[94] S. Ostermann, K. Plankensteiner, D. Bodner, G. Kraler, and R. Prodan, "Integration of an event-based simulation framework into a scientific workflow execution environment for grids and clouds," in *European Conference on a Service-Based Internet*, pp. 1–13, Springer, 2011.

[95] S. Callaghan, P. Maechling, P. Small, K. Milner, G. Juve, T. Jordan, E. Deelman, G. Mehta, K. Vahi, D. Gunter, K. Beattie, and C. X. Brooks, "Metrics for heterogeneous scientific workflows: A case study of an earthquake science application," *International Journal of High Performance Computing Applications*, vol. 25, no. 3, pp. 274–285, 2011.

[96] R. Mats, G. Juve, K. Vahi, S. Callaghan, G. Mehta, and P. J. M. andEwa Deelman, "Enabling large-scale scientific workflows on petascale resources using mpi master/worker," in *Proceedings of the 1st conference of the Extreme Science and Engineering Discovery Environment*, July 2012.

[97] "The TeraGrid Project." `http://www.teragrid.org`.

[98] E. Deelman, C. Kesselman, G. Mehta, L. Meshkat, L. Pearlman, K. Blackburn, P. Ehrens, A. Lazzarini, R. Williams, and S. Koranda, "GriPhyN and LIGO: building a virtual data grid for gravitational wave scientists," in *11th IEEE International Symposium on High Performance Distributed Computing (HPDC '02)*, 2002.

[99] S. Kalayci, G. Dasgupta, L. Fong, O. Ezenwoye, , and S. Sadjadi, "Distributed and adaptive execution of condor dagman workflows," in *Proceedings of the 22nd International Conference on Software Engineering and Knowledge Engineering (SEKE'2010)*, July 2010.

[100] G. Singh, M. Su, K. Vahi, E. Deelman, B. Berriman, J. Good, D. S. Katz, and G. Mehta, "Workflow task clustering for best effort systems with pegasus," in *15th ACM Mardi Gras Conference*, 2008.

[101] M. Hussin, Y. C. Lee, and A. Y. Zomaya, "Dynamic job-clustering with different computing priorities for computational resource allocation," in *The 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, May 2010.

[102] H. Zhao and X. Li, "Efficient grid task-bundle allocation using bargaining based self-adaptive auction," in *The 9th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, May 2009.

[103] S. P. Singh, A. Nayyar, R. Kumar, and A. Sharma, "Fog computing: from architecture to edge computing and big data processing," *The Journal of Supercomputing*, pp. 1–36, 2018.

[104] L. Tomas, B. Caminero, and C. Carrion, "Improving grid resource usage: Metrics for measuring fragmentation," in *12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid '12)*, pp. 352–359, 2012.

[105] X. Zhang, Y. Qu, and L. Xiao, "Improving distributed workload performance by sharing both cpu and memory resources," in *Proceedings of 20th International Conference on Distributed Computing Systems, (ICDCS'2000*, pp. 233–241, 2000.

[106] Z. Guo, M. Pierce, G. Fox, and M. Zhou, "Automatic task re-organization in mapreduce," in *Cluster Computing (CLUSTER), 2011 IEEE International Conference on*, pp. 335–343, 2011.

[107] H. Ying, G. Mingqiang, L. Xiangang, and L. Yong, "A webgis load-balancing algorithm based on collaborative task clustering," *Environmental Science and In-*

formation Application Technology, International Conference on, vol. 3, pp. 736–739, 2009.

[108] R. Ferreira da Silva, T. Glatard, and F. Desprez, "On-line, non-clairvoyant optimization of workflow activity granularity on grids," in *Euro-Par 2013 Parallel Processing*, vol. 8097 of *Lecture Notes in Computer Science*, pp. 255–266, Springer Berlin Heidelberg, 2013.

[109] N. Muthuvelu, J. Liu, N. L. Soe, S. Venugopal, A. Sulistio, and R. Buyya, "A dynamic job grouping-based scheduling for deploying applications with fine-grained tasks on global grids," in *Proceedings of the 2005 Australasian workshop on Grid computing and e-research*, 2005.

[110] N. Muthuvelu, I. Chai, and C. Eswaran, "An adaptive and parameterized job grouping algorithm for scheduling grid jobs," in *10th International Conference on Advanced Communication Technology (ICACT 2008)*, vol. 2, pp. 975 –980, 2008.

[111] N. Muthuvelu, C. Vecchiola, I. Chai, E. Chikkannan, and R. Buyya, "Task granularity policies for deploying bag-of-task applications on global grids," *Future Generation Computer Systems*, vol. 29, no. 1, pp. 170 – 181, 2013. ¡ce:title¿Including Special section: AIRCC-NetCoM 2009 and Special section: Clouds and Service-Oriented Architectures¡/ce:title¿.

[112] N. W. Keat, A. T. Fong, L. T. Chaw, and L. C. Sun, "Scheduling framework for bandwidth-aware job grouping-based scheduling in grid computing," *Malaysian Journal of Computer Science*, vol. 19, no. 2, pp. 117–126, 2006.

[113] Q. Liu and Y. Liao, "Grouping-based fine-grained job scheduling in grid computing," in *First International Workshop on Education Technology and Computer Science*, Mar. 2009.

[114] Z. Zhou, Z. Cheng, L.-j. Zhang, W. Gaaloul, and K. Ning, "Scientific Workflow Clustering and Recommendation Leveraging Layer Hierarchical Analysis," *IEEE Transactions on Services Computing*, vol. 11, no. 1, pp. 169–183, 2018.

[115] W. Chen, R. F. Da Silva, E. Deelman, and T. Fahringer, "Dynamic and fault-tolerant clustering for scientific workflows," *IEEE Transactions on Cloud Computing*, vol. 4, no. 1, pp. 49–62, 2016.

[116] T. Koohi-var and M. Zahedi, "S CIENTIFIC W ORKFLOW C LUSTERING BASED O N," vol. 7, no. 4, pp. 1–13, 2017.

[117] V. Silva, F. Chirigati, K. Maia, E. Ogasawara, D. Oliveira, V. Braganholo, L. Murta, and M. Mattoso, "Similarity-based workflow clustering," in *JCIS*, vol. 2, pp. 23–35, 2011.

[118] T. Fahringer, R. Prodan, R. Duan, J. Hofer, F. Nadeem, F. Nerieri, S. Podlipnig, J. Qin, M. Siddiqui, H.-L. Truong, *et al.*, "Askalon: A development and grid computing environment for scientific workflows," in *Workflows for e-Science*, pp. 450–471, Springer, 2007.

[119] G. B. Berriman, G. Juve, E. Deelman, M. Regelson, and P. Plavchan, "The application of cloud computing to astronomy: A study of cost and performance," in *Workshop on e-Science challenges in Astronomy and Astrophysics*, 2010.

[120] H. Topcuoglu, S. Hariri, and W. Min-You, "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 13, no. 3, pp. 260–274, 2002.

[121] J. Blythe, S. Jain, E. Deelman, Y. Gil, K. Vahi, A. Mandal, and K. Kennedy, "Task scheduling strategies for workflow-based applications in grids," in *5th IEEE International Symposium on Cluster Computing and the Grid (CCGrid '05)*, 2005.

[122] W. Chen, R. F. Da Silva, E. Deelman, and R. Sakellariou, "Balanced task clustering in scientific workflows," *Proceedings - IEEE 9th International Conference on e-Science, e-Science 2013*, pp. 188–195, 2013.

[123] W. Chen and E. Deelman, "Workflowsim: A toolkit for simulating scientific workflows in distributed environments," in *2012 IEEE 8th International Conference on E-Science*, pp. 1–8, IEEE, 2012.

[124] E. Deelman and A. Chervenak, "Data management challenges of data-intensive scientific workflows," in *Cluster Computing and the Grid, 2008. CCGRID'08. 8th IEEE International Symposium on*, pp. 687–692, IEEE, 2008.

[125] A. Labrinidis and H. V. Jagadish, "Challenges and opportunities with big data," *Proceedings of the VLDB Endowment*, vol. 5, no. 12, pp. 2032–2033, 2012.

[126] J. J. Rehr, F. D. Vila, J. P. Gardner, L. Svec, and M. Prange, "Scientific computing in the cloud," *Computing in science & Engineering*, vol. 12, no. 3, pp. 34–43, 2010.

[127] A. Saxena, G. Shrivastava, and K. Sharma, "Forensic investigation in cloud computing environment," *The International Journal of forensic computer science*, vol. 2, pp. 64–74, 2012.

[128] H. N. Wang, W. X. Xu, and C. L. Jia, "A high-speed railway data placement strategy based on cloud computing," in *Applied Mechanics and Materials*, vol. 135, pp. 43–49, Trans Tech Publ, 2012.

[129] K. Sharma and B. Gupta, "Multi-layer defense against malware attacks on smartphone wi-fi access channel," *Procedia Computer Science*, vol. 78, pp. 19–25, 2016.

[130] S. K. Shrivastava, P. Kumar, and A. Pandey, "Impact of software licenses in cloud computing based e-governance initiatives," in *Communication Systems and Network Technologies (CSNT), 2014 Fourth International Conference on*, pp. 592–596, IEEE, 2014.

[131] G. Shrivastava, K. Sharma, and A. Bawankan, "A new framework semantic web technology based e-learning," in *Environment and Electrical Engineering (EEEIC), 2012 11th International Conference on*, pp. 1017–1021, IEEE, 2012.

[132] C. Blum and A. Roli, "Metaheuristics in combinatorial optimization: Overview and conceptual comparison," *ACM computing surveys (CSUR)*, vol. 35, no. 3, pp. 268–308, 2003.

[133] X.-S. Yang, "Metaheuristic optimization," *Scholarpedia*, vol. 6, no. 8, p. 11472, 2011.

[134] Z. W. Geem, J. H. Kim, and G. V. Loganathan, "A new heuristic optimization algorithm: harmony search," *simulation*, vol. 76, no. 2, pp. 60–68, 2001.

[135] J. Holland, "Adaptation in natural and artificial systems: an introductory analysis with application to biology," *Control and artificial intelligence*, 1975.

[136] R. Kennedy, "J. and eberhart, particle swarm optimization," in *Proceedings of IEEE International Conference on Neural Networks IV, pages*, vol. 1000, 1995.

[137] X.-S. Yang and S. Deb, "Cuckoo search via lévy flights," in *Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on*, pp. 210–214, IEEE, 2009.

[138] M. Khari and P. Kumar, "An effective meta-heuristic cuckoo search algorithm for test suite optimization," *Informatica*, vol. 41, no. 3, 2017.

[139] X.-S. Yang, "Firefly algorithm, stochastic test functions and design optimisation," *International Journal of Bio-Inspired Computation*, vol. 2, no. 2, pp. 78–84, 2010.

[140] X.-S. Yang, "A new metaheuristic bat-inspired algorithm," in *Nature inspired cooperative strategies for optimization (NICSO 2010)*, pp. 65–74, Springer, 2010.

[141] J. K. Wang and X. Jia, "Data security and authentication in hybrid cloud computing model," in *Global High Tech Congress on Electronics (GHTCE), 2012 IEEE*, pp. 117–120, IEEE, 2012.

[142] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE transactions on evolutionary computation*, vol. 6, no. 2, pp. 182–197, 2002.

[143] Y. Yang, K. Liu, J. Chen, J. Lignier, and H. Jin, "Peer-to-peer based grid workflow runtime environment of swindew-g," in *e-Science and Grid Computing, IEEE International Conference on*, pp. 51–58, IEEE, 2007.

[144] Wikipedia contributors, "Corvus — Wikipedia, the free encyclopedia," 2018. [Online; accessed 2-September-2018].

[145] A. D. Craig and A. Craig, "How do you feel–now? the anterior insula and human awareness.," *Nature reviews neuroscience*, vol. 10, no. 1, 2009.

[146] D. C. Penn and D. J. Povinelli, "On the lack of evidence that non-human animals possess anything remotely resembling a 'theory of mind'," *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, vol. 362, no. 1480, pp. 731–744, 2007.

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

[148] S. Venugopal and R. Buyya, "An scp-based heuristic approach for scheduling distributed data-intensive applications on global grids," *Journal of Parallel and Distributed Computing*, vol. 68, no. 4, pp. 471–487, 2008.

[149] Y. Yang, K. Liu, J. Chen, X. Liu, D. Yuan, and H. Jin, "An algorithm in swindew-c for scheduling transaction-intensive cost-constrained cloud workflows," in *eScience, 2008. eScience'08. IEEE Fourth International Conference on*, pp. 374–375, IEEE, 2008.

[150] A. Askarzadeh, "A novel metaheuristic method for solving constrained engineering optimization problems : Crow search algorithm," vol. 169, pp. 1–12, 2016.

[151] J. Yan, Y. Yang, and G. K. Raikundalia, "Swindew-a p2p-based decentralized workflow management system," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 36, no. 5, pp. 922–935, 2006.

[152] T. Wu, H. Gu, J. Zhou, T. Wei, X. Liu, and M. Chen, "Soft error-aware energy-efficient task scheduling for workflow applications in dvfs-enabled cloud," *Journal of Systems Architecture*, vol. 84, pp. 12–27, 2018.

[153] X. Zhang, T. Wu, M. Chen, T. Wei, J. Zhou, S. Hu, and R. Buyya, "Energy-aware virtual machine allocation for cloud with resource reservation," *Journal of Systems and Software*, vol. 147, pp. 147–161, 2019.

[154] Z. Zhu, G. Zhang, M. Li, and X. Liu, "Evolutionary multi-objective workflow scheduling in cloud," *IEEE Transactions on parallel and distributed Systems*, vol. 27, no. 5, pp. 1344–1357, 2016.

[155] E. N. Alkhanak, S. P. Lee, and S. U. R. Khan, "Cost-aware challenges for workflow scheduling approaches in cloud computing environments: Taxonomy and opportunities," *Future Generation Computer Systems*, vol. 50, pp. 3–21, 2015.

[156] H. Topcuoglu, S. Hariri, and M.-y. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE transactions on parallel and distributed systems*, vol. 13, no. 3, pp. 260–274, 2002.

[157] Z. Tang, X. Zhang, K. Li, and K. Li, "An intermediate data placement algorithm for load balancing in spark computing environment," *Future Generation Computer Systems*, vol. 78, pp. 287–301, 2018.

[158] L. Xu, K. Wang, Z. Ouyang, and X. Qi, "An improved binary pso-based task scheduling algorithm in green cloud computing," in *9th International Conference on Communications and Networking in China*, pp. 126–131, IEEE, 2014.

[159] S. Mullainathan and J. Spiess, "Machine learning: an applied econometric approach," *Journal of Economic Perspectives*, vol. 31, no. 2, pp. 87–106, 2017.

[160] K. Li, "Energy and time constrained task scheduling on multiprocessor computers with discrete speed levels," *Journal of Parallel and Distributed Computing*, vol. 95, pp. 15–28, 2016.

[161] V. Kumar, C. Katti, and P. Saxena, "A novel task scheduling algorithm for heterogeneous computing," *International Journal of Computer Applications*, vol. 85, no. 18, 2014.

[162] H. Arabnejad and J. G. Barbosa, "List scheduling algorithm for heterogeneous systems by an optimistic cost table," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 3, pp. 682–694, 2014.

[163] M. F. Akbar, E. U. Munir, M. M. Rafique, Z. Malik, S. U. Khan, and L. T. Yang, "List-based task scheduling for cloud computing," in *2016 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing*

and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), pp. 652–659, IEEE, 2016.

[164] L. Zeng, B. Veeravalli, and X. Li, "Saba: A security-aware and budget-aware workflow scheduling strategy in clouds," *Journal of parallel and Distributed computing*, vol. 75, pp. 141–151, 2015.

[165] H. M. Fard, R. Prodan, and T. Fahringer, "A truthful dynamic workflow scheduling mechanism for commercial multicloud environments," *IEEE Transactions on Parallel and Distributed systems*, vol. 24, no. 6, pp. 1203–1212, 2013.

[166] R. N. Calheiros and R. Buyya, "Meeting deadlines of scientific workflows in public clouds with tasks replication," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 7, pp. 1787–1796, 2014.

[167] 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*, vol. 48, pp. 1–18, 2015.

[168] Amazon, "Amazon ec2 instance types." `https://aws.amazon.com/ec2/instance-types/`, 2008. [Online; accessed 19-March-2019].

[169] C. Q. Wu, X. Lin, D. Yu, W. Xu, and L. Li, "End-to-end delay minimization for scientific workflows in clouds under budget constraint," *IEEE Transactions on Cloud Computing*, vol. 3, no. 2, pp. 169–181, 2015.

[170] D. M. Abdelkader and F. Omara, "Dynamic task scheduling algorithm with load balancing for heterogeneous computing system," *Egyptian Informatics Journal*, vol. 13, no. 2, pp. 135–145, 2012.

[171] A. Kaur, P. Gupta, and M. Singh, "Hybrid balanced task clustering algorithm for scientific workflows in cloud computing," *Scalable Computing: Practice and Experience*, vol. 20, no. 2, pp. 237–258, 2019.

[172] V. Arabnejad, K. Bubendorfer, and B. Ng, "Budget and deadline aware e-science workflow scheduling in clouds," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 1, pp. 29–44, 2019.

[173] P. G. Avinash Kaur, "A data placement strategy based on crow search algorithm in cloud computing," *Recent Patents on Computer Science*, vol. 12, no. 1, 2019.

[174] "Source-Code of Workflow Generator,." `https://github.com/pegasus-isi/WorkflowGenerator`. Accessed: 2018-09-30.

[175] M. Wang, K. Ramamohanarao, and J. Chen, "Trust-based robust scheduling and runtime adaptation of scientific workflow," *Concurrency and Computation: Practice and Experience*, vol. 21, no. 16, pp. 1982–1998, 2009.

[176] Y. Caniou, E. Caron, A. K. W. Chang, and Y. Robert, "Budget-aware scheduling algorithms for scientific workflows with stochastic task weights on heterogeneous iaas cloud platforms," in *2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pp. 15–26, May 2018.

LIST OF PUBLICATIONS

- **Avinash Kaur,** Pooja Gupta and Manpreet Singh, "Hybrid Balanced Task Clustering Algorithm for Scientific Workflows in cloud computing", *Scalable Computing: Practice and Experience, 20(2), pp. 237-258*, DOI: https://doi.org/10.12694/scpe.v20i2.1515, 2019. (Scopus, ESCI)

- **Avinash Kaur,** Pooja Gupta , "A Data Placement Strategy Based on Crow Search Algorithm in Cloud Computing", *Recent Patents on Computer Science 12(1)*, DOI : https://doi.org/10.2174/ 2213275912666181127123431, 2019. (Scopus)

- **Avinash Kaur,** Pooja Gupta and Manpreet Singh, Anand N. "Data Placement in Era Of Cloud Computing: A Survey, Taxonomy And Open Research Issues", *Scalable Computing: Practice and Experience, 20(2), pp. 377-398*, DOI : https://doi.org/10.12694/scpe.v20i2.1530, 2019. (Scopus, ESCI)

- **Avinash Kaur,** Pooja Gupta and Manpreet Singh, "DPO-HEFT(Data Placement Oriented HEFT) for Scheduling Scientific Workflows in Cloud Computing", *Recent Patents on Computer Science* (Under Review), 2019. (Scopus)

- **Avinash Kaur,** Vaishali and Pooja Gupta,"A survey on the services provided by various cloud providers", *Jn. of Control Theory and Applications*, 2016. (Scopus)

- **Avinash Kaur,** Pooja Gupta and Manpreet Singh, "Task Clustering and Data Placement Aligned Scheduling of Scientific Workflows", *International Conference on Intelligent Computing and Control Systems, 2019. IEEE (Scopus).*