A Hybrid Approach for Time Efficient and Reliable Workflow Scheduling in Cloud Computing

Thesis submitted in partial fulfillment of the requirements for the award of degree of

Master of Engineering
in
Computer Science and Engineering

Submitted By
Jatin Jain
(Roll No. 801232009)

Under the supervision of:
Dr. Seema Bawa
Professor

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
THAPAR UNIVERSITY
PATIALA – 147004

June 2014

Certificate

I hereby certify that the work which is being presented in the thesis entitled, "*A Hybrid Approach for Time Efficient and Reliable Workflow Scheduling in Cloud Computing*", in partial fulfillment of the requirements for the award of degree of Master of Engineering in *Computer Science and Engineering* submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of *Dr. SeemaBawa*and refers other researcher's work which are duly listed in the reference section.

The matter presented in the thesis has not been submitted for award of any other degree of this or any other University.

Jatin Jain

(Jatin Jain)

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

Seema Bawa
14/7/14

(Dr. SeemaBawa)

Professor,
Computer Science and Engineering Department

Countersigned by

(Dr. Deepak Garg)
Head
Computer Science and Engineering Department
Thapar University
Patiala

(Dr. S. K. Mohapatra)
Dean (Academic Affairs)
Thapar University
Patiala

Acknowledgement

I would like to express my deep sense of gratitude to my supervisor Dr. Seema Bawa, Professor, Computer Science & Engineering Department for her invaluable guidance, support and encouragement during this thesis work. She provided me all the resources and guidance throughout thesis work.

I express my gratitude to Dr. Deepak Garg, Head, Computer Science & Engineering Department for providing us adequate environment, facility for carrying thesis work.

I would also like to thank all staff members and my colleagues who were always there at the need of hour and provided with all help and facilities, which I required, for the completion of my thesis work.

I would also like to thanks to all my friends for helping me in the hour of need and providing me all the help and support for completion of my thesis.

Last but not the least I am highly grateful to all my family members for their inspiration and ever encouraging moral support, which enables me to purse my studies.

Jatin Jain

Abstract

Nowadays, scientific and business applications are becoming quite complex all around the world. These large scale problems require computing power that goes beyond the capabilities of a single machine. The data and computation requirements of these problems demand a high performance computing environment such as a cluster, a grid or a cloud platform in order to be solved in a reasonable amount of time. Workflow provides an easy and efficient way to implement these applications. Many companies and researchers are opting to implement the workflow in cloud computing. In order to efficiently execute workflows and utilize the cloud resources in an appropriate way, scheduling policies should be implemented.

Most of the workflow applications hosted by cloud are real time applications which require high reliability and fast execution. A number of fault can occurs in cloud due to hardware faults, software faults, network faults and so on. So, there is need of methods for providing the reliable execution of the applications which are efficient in terms of time.

In this thesis, we have proposed a hybrid approach for time efficient and reliable workflow scheduling in cloud as a combination of two existing state-of-the-art heuristics which considers the parameters of reliability and time while scheduling the tasks. Using simulation, we have compared the performance of our proposed MCTR algorithm with the existing reliable scheduling algorithm. Experiment results show that proposed algorithm gives better results which optimize makespan of workflow while providing reliable execution.

Table of Content

List of Figures

List of Tables

Chapter 1
Introduction

This chapter introduces cloud computing, its evolution and various related technologies like grid computing, utility computing, distributed computing. A brief introduction about various cloud service and deployment models, issues in cloud computing is given followed by research motivation and organization of the thesis.

1.1. Cloud Computing

Throughout its 60 year of history, computer systems have been evolved in a spiral way of integration and distribution. They develop gradually from centralized, huge, sharable mainframes in 1970 to decentralized, small, private PC's in 1999. In 2010s, they are again moving into compact, shareable machines invisible to users: so called Cloud Computing [1]. Cloud computing is a type of internet computing that relies on sharable computing resources in place of private or local resources easily accessible through internet. With the emergence of cloud computing, the long-held dream of computing as a service come to true [2] [3].

1.1.1. Definitions of Cloud Computing

According to Gartner, "*Cloud computing is a style of computing where massively scalable IT-related capabilities are provided "as a service" using Internet technologies to connect multiple external customers*" [4].

According to NIST:"*Cloud is a model for enabling convenient, on demand, network access to shared pool of configurable computing resources(e.g. networks, servers, storage, applications and services) that can be rapidly provisioned and leased with minimal effort or service provider interaction*"[5].

According to NIST definition, every cloud has following five main characteristics [5]:-

i) **On-demand self service**: A user can independently provision computing resources like processing capacity, memory, applications etc. as per requirements automatically without any human interaction or help from any service provider.

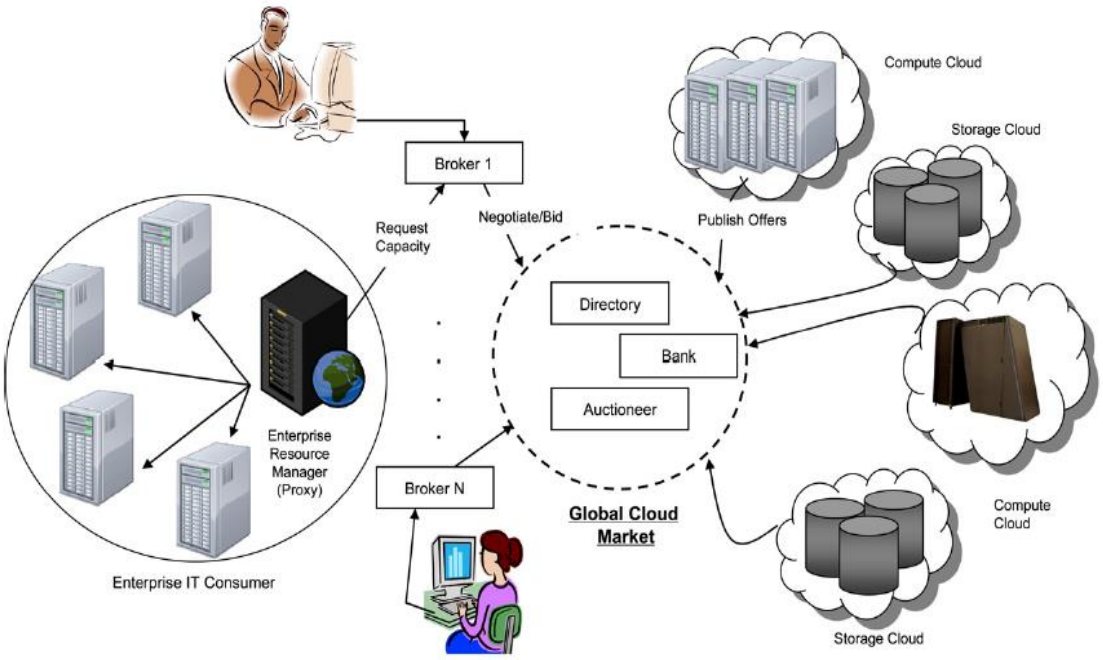


Figure 1.1: A View of Cloud Computing [3]

ii) **Broad Network Access**: Cloud computing is a type of internet computing where services are available anywhere over the network through internet access. A user can easily access cloud services through heterogeneous thin and thick client platforms.

iii) **Resource Pooling***:* In cloud, we have a huge collection of heterogeneous resources such as storage, processing power, memory, network bandwidth that are pooled together which serves multiple user requirements using a multi-tenant model, where different resources are dynamically provisioned and re-provisioned on basis of user demand. There is a sense of location dependency as user does not know the actual location of resources used.

iv) **Rapid Elasticity**: In cloud resources can be dynamically scaled up and down according to user requirement easily and quickly. From user point of view, the capability appears to be unlimited and can be used any time.

v) **Measured Service**: Cloud is a measured service based upon pay-per-use model where user have to pay for what they have used. Resource usage can be easily measured, monitored, reported providing transparency to both user and service provider.

1.2. Evolution of Cloud Computing

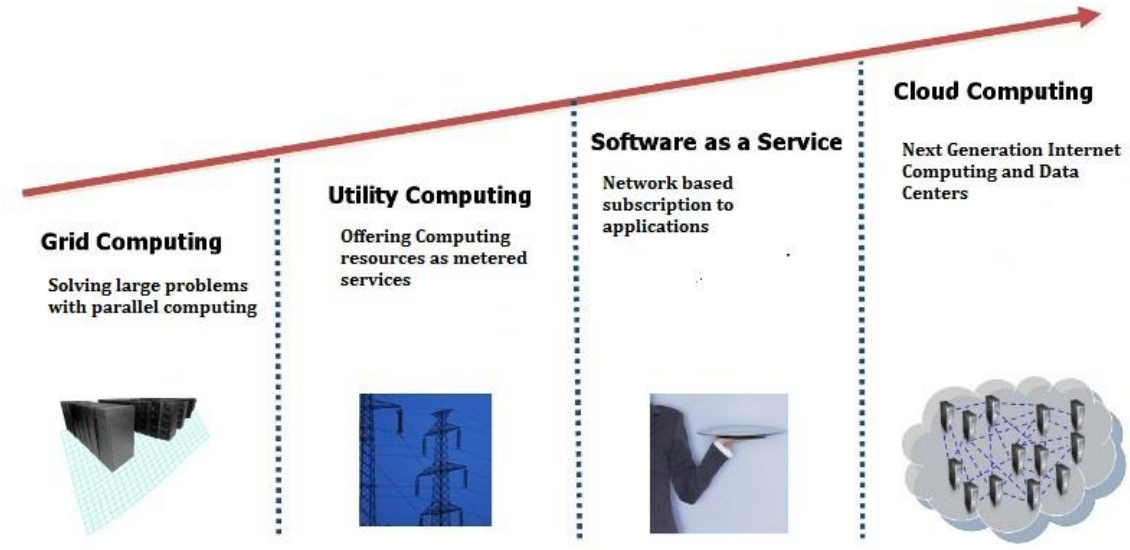


Figure 1.2: Evolution of Cloud Computing [6]

In 1969 Leonard Kleinrock, one of the chief scientist of the original Advanced Research Projects Agency Network (ARPANET) which seeded the internet said:

"As of now, computer networks are still in their infancy, but as they grow up and become sophisticated, we will probably see the spread of computer utilities which, like present telephone and electric utilities, will service individual home and offices across the country"[7].

This vision of computing utilities based on a service provisioning model anticipated the massive transformation of entire computing industry in the 21st century. The era of cloud computing started in late 1980s with the concept of grid computing when, for the first time, a number of computing resources where pooled together for solving a scientific problem which require high level of parallel computing. Grid Computing [8] enables sharing of wide variety of geographically distributed computing resources owned by different organizations for solving large scale resource and data intensive applications in field of science, commerce and engineering. Long distance optical networks are employed to connect different universities to form a computing grid of massive computing power in order to share this massive power for complex and large scientific problems. But grids are available for limited number of people, mostly used for scientific and research work.

In 1990, telecommunication companies began offering virtual private network service having high quality and low cost [8]. As computer became more widespread, researchers start exploring way to make large scale computing power available to more users. Also, the concept of virtualization was elaborated beyond virtual servers to higher levels of abstraction, first the virtual platform, including storage and network resources, and subsequently the virtual application, which has no specific underlying infrastructure. Utility computing offered clusters as virtual platforms for computing with a metered business model [9].

With increase in network bandwidth and new technologies like Java, PHP, Ajax, it is now possible to make interactive websites like online shopping, social networking, online ticket reservation and so on. This new concept is referred as Software-as-a-Service which gains popularity around in year 2000[10].

In 1999, saleforce.com put the idea of cloud computing as a service and launches a site which delivers its enterprises application to its user through network access. In 2002 Amazon launch the cloud based web services since then Amazon is the market leader in cloud service provider providing a large range of services. [3][11].

More recently Software as a Service (SaaS) has raised the level of virtualization to the application, with a business model of charging not by the resources consumed but by the value of the application to subscribers. The concept of cloud computing has evolved from the concepts of grid, utility and SaaS. It is an emerging model through which users can gain access to their applications from anywhere, at any time, through their connected devices. These applications remain in huge scalable data centers where resources can be dynamically provisioned and shared to achieve significant economies of scale.

A number of companies, including Google, Microsoft, Amazon, and IBM, have built huge datacenter-based computing power all over the world to support their Web service offerings. With this computing infrastructure in place these companies are already poised to offer new cloud-based software applications.

Cloud Computing make it possible for small organizations to use heavily priced software's like ERP (Enterprise Resource Planning) through SaaS. This is just a small example of importance of cloud computing [3].

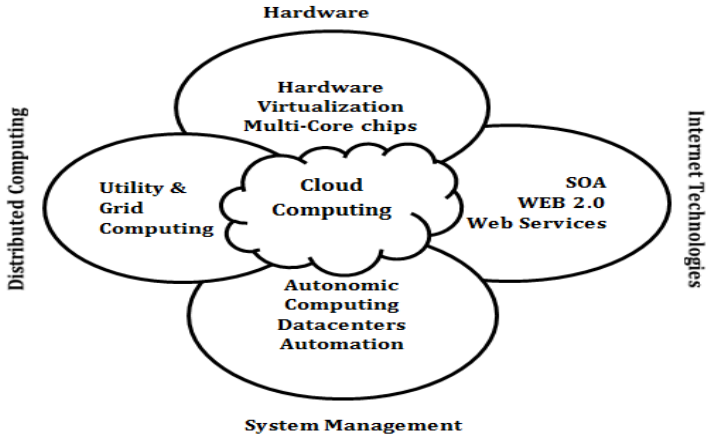


Figure 1.3 Cloud Computing and its Related Technologies [12]

1.3 Cloud Computing Framework

Cloud contains large pool of heterogeneous virtual resources that can be dynamically provisioned easily and without any overhead. Figure 1.4 gives the framework of the cloud computing [5].

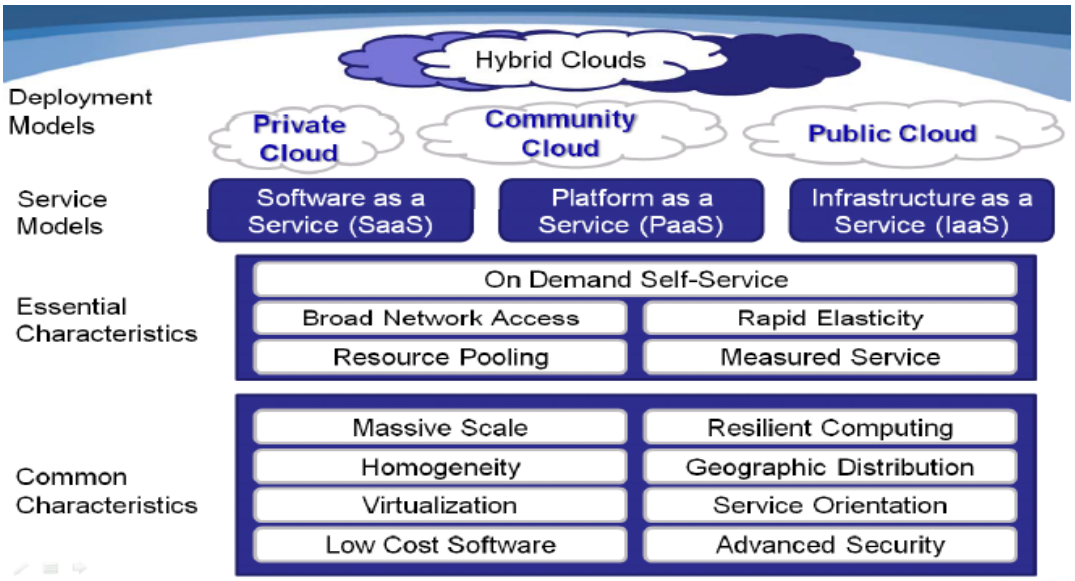


Figure 1.4: Cloud Computing Framework [5]

1.3.1. Cloud Computing Deployment Models

Cloud services can be deployed in different ways, depending on the organizational structure and provisioning location. In cloud, we have basically four different deployment models [3] [5] [6].

i) Private Cloud

Private clouds are built exclusively for a particular organization with in its premises to provide service to its internal users. Private clouds are mostly used by large organization and government firms due to security and data sovereignty concerns. Private cloud offers control over security and resilience because of restricted access to a single organization. One of main disadvantage of private cloud is inability to scale dynamically as compare public cloud. Popular examples of private cloud include Amazon Virtual Private Cloud (Amazon VPC), Eucalyptus Cloud Platform, IBM SmartCloud.

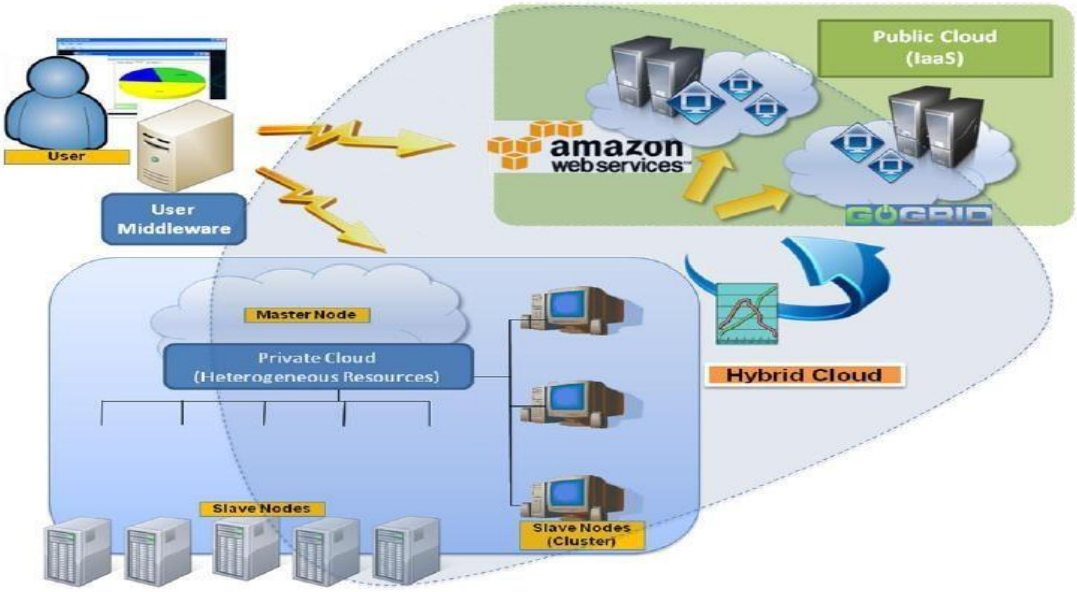


Figure 1.5: Cloud Computing Deployment Models [11]

ii) Public Cloud

A public cloud is a publicly accessible cloud environment owned by third-party cloud providers. Public cloud is the most common deployment model where services are available to anyone through internet on the basis of pay-per-use manner. Cloud provider is a profit making firm which will provide you services and charge for it accordingly. A public cloud can fulfill the requirement of all type of user, from small to large businesses and home users. Popular examples of public clouds include Amazon Elastic Cloud Compute, Google App Engine, Blue Cloud by IBM and Azure services Platform by Windows.

iii) Hybrid Cloud

Hybrid cloud is a combination of both public and private cloud which offers greater control, flexibility and security. Cloud bursting is a state where private cloud is unable to fulfill the current resources requirement so, it make use of public cloud to fulfill the current needs. The main advantage of using hybrid deployment is its cost advantage, as organization has to pay for extra resources only when they are needed. Hybrid cloud architecture requires both on-premises resources and off-site (remote) server-based cloud infrastructure.

iv) Community Cloud

Community cloud is a special type of cloud which is open between the organizations with similar interests and requirement. These clouds are owned and managed internally by one or more organization or by a third-party group of organization. The cost of cloud is shared among the organizations and only the user from that group of organization can use the cloud. Some areas where community cloud is used are government agencies, universities, NGOs.

1.3.2. Cloud Computing Service Models

Cloud computing is typically divided into three levels of service offerings as shown in figure given below [3] [5] [6]:-

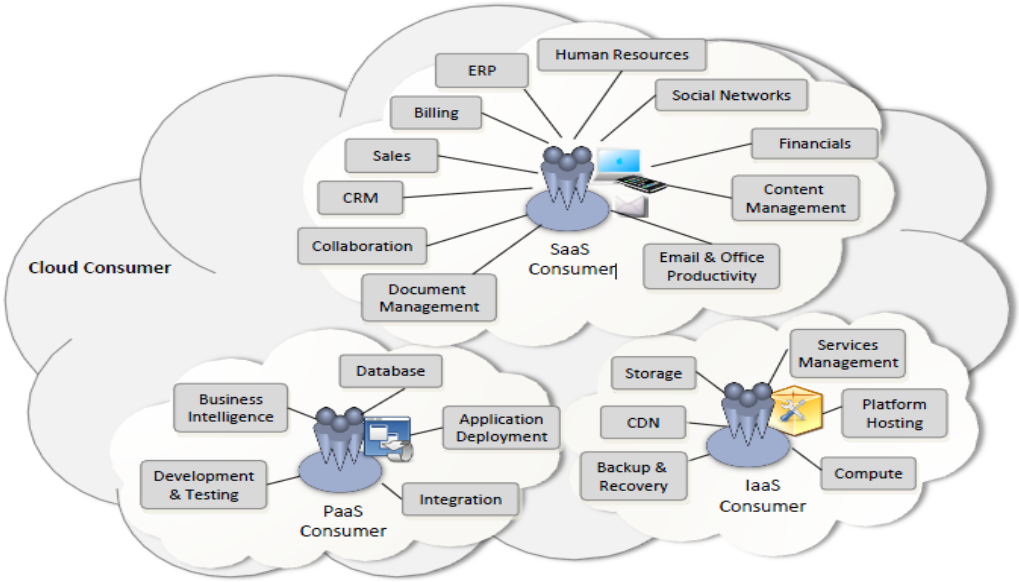


Figure 1.6: Cloud Computing Service Models [11]

i) **Software as a Service (SaaS)**

In SaaS, complete application access is provided to user which runs on cloud infrastructure, as service. A single instance of application is running on cloud servicing multiple end users. The applications are accessible through thin client or thick client. The users have only to pay for the service while providers have to manage and pay for underlying infrastructure and software. Some example of SaaS is Gmail by Google, Facebook, Salesforce.

ii) **Platform as a Service (PaaS)**

PaaS provides application development and deployment platform to the developers as a service. It allows development and deployment of application without the cost and complexity of purchasing and managing the underlying infrastructure and provide suitable and configurable runtime environment for the efficient development and deployment of the application. PaaS consist of computing resources, memory, storage space, network bandwidth, software like database, OS etc. Google Apps Engine is a PaaS platform provided by Google where user can easily develop and deploy there apps.

iii) **Infrastructure as a Service (IaaS)**

The capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications. The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, and deployed applications; and possibly limited control of select networking components. Amazon Web Services Elastic Compute Cloud (EC2) and Secure Storage Service (S3), GoGrid, 3 Tera are the examples of IaaS offerings.

1.4. Research Issues in Cloud Computing

The emergence of cloud computing has made a tremendous impact on IT industry over the past few years. Cloud computing provides an economical, flexible and reliable ways of computing to IT industry. Cloud computing is in initial stage, with many issues still need to be addressed [13] [14] [15] [16].

i) Energy Efficiency

Improving energy efficiency is one of the major issues in cloud computing. According to reports, cost of powering and cooling accounts for 53% of total operational cost of data center. Efficient hardware architectures such as slowing down processors and switching off unused resources are some commonly used methods. Server consolidation and energy aware job scheduling can also help in reducing power consumption. Finding a trade-off solution between energy efficiency and achieving desirable performance is a one of the key challenge.

ii) Fault Tolerance

Fault tolerance is another major issue in cloud computing. A number of failures can occur in cloud due hardware faults, software faults etc. According to Survey, by Gibson and Schroeder [17], most of the failure occurs in cloud due to hardware failure mainly in processor, hard-disk, IC-sockets and memory. For achieving required performance, these faults should be handled. Achieving highly reliable cloud environment is one of the key issues.

iii) Virtualization

Virtualization is one of key technology behind cloud computing which provide functionalities like VM migration. VM migration is a process of transferring a VM from one server to other server. Live machine migration is used to transfer the entire OS and all application from one server to another without stopping task execution.VM migration can be helpful in solving many other issues like load balancing, handling faults etc. The main issues in this are how to detect hotspots and start migration process.

iv) Server Consolidation

Server Consolidation approach is use to maximize resource utilization and minimize energy consumption. Live machine migration is used to transfer VM from underutilized servers onto a single server. But we have to consider a number of factors while performing server consolidation such as performance requirement, resource requirement of applications. The major issue arises are which machine to migrate, where to migrate and when to migrate.

v) Interoperability

Interoperability means to allow applications to be shifted from one cloud to another or to use multiple clouds together without any difficulty. Cloud providers do not provide enough transparency to users. Sometimes, user doesn't get required performance and transparency from providers for e.g. access issue. In such cases users get frustrated and want to change cloud provides, for these situation we, require some autonomous agent that allow user to choose and shift from different cloud service providers for this we require a high level of interoperability.

vi) Data Security

Data security is another important research topic in cloud. Since users don't have direct access to physical security of their data, they have to rely on the service provider for data security. The service provider must provide.

- Confidentiality- for secure data access and transfer
- Auditability- for attesting weather data security has been tempered or not.

Cryptography protocols are used to maintain confidentiality while auditability is achieved through remote attestation techniques. These attestation techniques required Trusted Platform Module (TPM) to generate system security reports. Recent work has been done to designing efficient protocols for trust establishment and management.

vii) Scheduling

For the efficient execution of tasks and utilize the cloud resources in an efficient way, scheduling policies need to be implemented. Proper scheduling can have a major impact on the system performance. Scheduling is a process of mapping task to resources. While scheduling tasks to resources we have to consider a number of factors like cost, time, reliability, energy consumption, load balancing etc. There are a number of algorithms available for scheduling workflow in cloud but still there is a need of more efficient algorithms.

1.5. Workflow Scheduling in Cloud Computing

A workflow consists of a number of interdependent tasks of a business or scientific activity which provide services or process information. Workflows are used to automate these activities and execute them efficiently. There are various example of workflow

application in bioinformatics, business enterprises, astronomy where a big task or objective is divided into to a number of interdependent subtasks to execute and manage them efficiently.

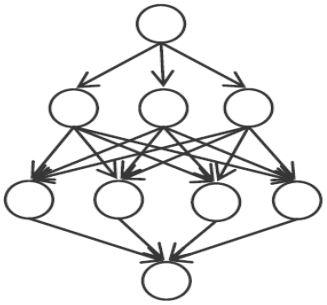


Figure 1.7: A Simple Workflow Example [18].

A workflow application is generally represented as a Directed Acyclic Graph (DAG) which is a directed graph with no cycles such as G(N,E) where N represent a set of nodes, each node represent a task and E represent set of edges, each edge represent a dependency among a task. In workflow execution we have to maintain parent child relationship such that a child task cannot be executed before its parent task. A task with no parent task is called as *entry task* and a task without a child task is called as *exit task.*

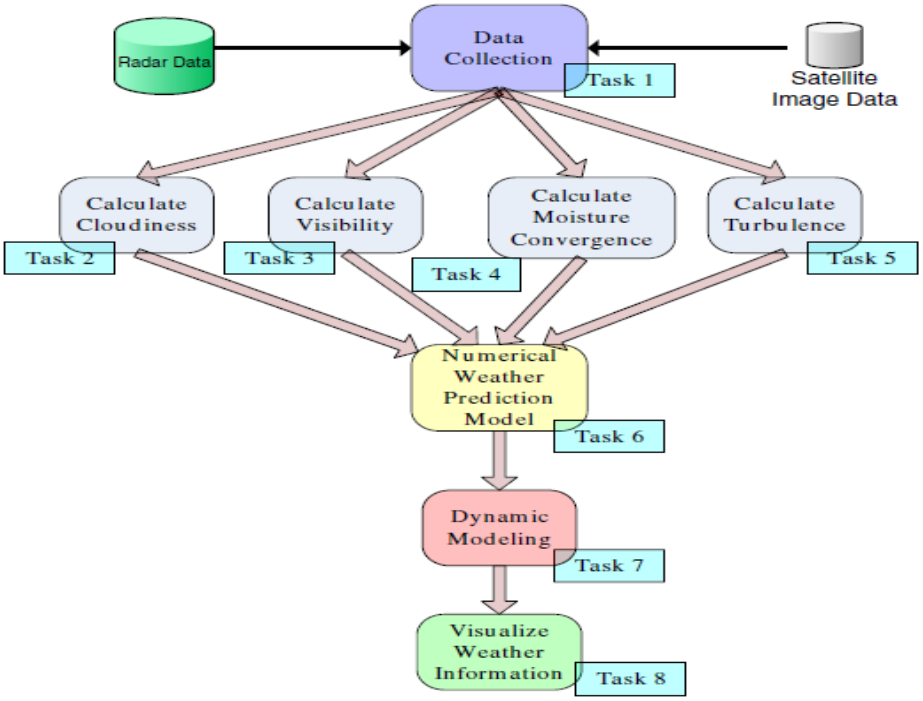


Figure 1.8: A Real Life Example of Workflow for Weather Monitoring System [19].

In order to efficiently execute the workflow and utilize the cloud resources in an appropriate way, scheduling policies need to be implemented. Scheduling is the process of mapping and managing the execution of workflow tasks on the allocated resources while satisfying user QoS requirements.

1.5.1. Need of Implementing Workflow in Cloud Computing

Nowadays, scientific and business applications are becoming quite complex all around the world. These large scale applications require high performance computing which are beyond the capability of single machine. Workflow provides an easy and efficient way to implement these applications. Simultaneously, the need of resources for executing these applications that are available all time and at low cost has increased. There are number of issues related to workflow execution like workflow contains a number of interdependent tasks each have different processing and data requirements. Some of tasks require parallel processing while other require specific runtime environment for execution. Some workflow applications require advance reservation of resources while sometimes it is difficult to predict the resource requirement in advance. Cloud Computing provide an economical and flexible solution to execute these workflow in reasonable amount of time. Due to these reasons, researchers and companies are opting to implement their workflows on cloud.

1.5.2. QoS Parameters for Workflow Scheduling

While performing workflow scheduling, we have to consider a number of QoS parameters to achieve desired performance. These parameters are [18] [20] [21]:-

i) **Makespan**

This is one of the major factors which should be minimized while workflow scheduling. Makespan refers to the total execution time of the workflow i.e. time from starting execution of entry task to finishing execution of exit task.

ii) **Cost**

This is another major factor for workflow scheduling as users have to pay for each resource acquired by him. Cost refers to overall cost of workflow execution on resources including storage cost, data transfer cost and other costs.

iii) **Reliability**

A number of faults occur in cloud and whenever fault occurs it may lead to failure of application. Therefore, reliable execution of the workflow should be provided.

iv) **Energy**

Energy is another important factor considered during workflow execution to reduce energy consumption cost and carbon footprint.

v) **Load Balancing**

Load balancing refers to distributing load to all the resources in equally and efficient manner. As good load balancing help in improving system performance.

vi) **Resource Utilization**

Resource utilization refers to utilizing the acquired resources in an efficient manner so as to reduce cost and energy consumption of workflow execution.

vii) **Scalability**

Scalability refers to the scaling up and down resources dynamically while scheduling so as to maintain required system performance and overall cost of execution.

1.6 Research Motivation

Cloud computing is a current advancement where applications and IT infrastructure are provided as a services on a usage based payment model. A number of companies and scientist are opting cloud computing for processing their workflow applications in the field of astronomy, gravitational-physics, biology, climate modeling, and life-sciences. Workflow scheduling is one of the key issues in the effective execution of the workflow as proper scheduling can have a major impact on the overall system performance and user QoS requirements. While performing workflow scheduling there are a number of parameter needs to be considered. Most of the applications hosted by cloud are real time applications which require high reliability and fast execution. Cloud contains a large pool of resources like processors, storage, communication network which are prone to failure. Whenever fault occurs, it causes loss of time and money as we have to re-execute the task and it will take time and money both.

There are number of algorithms exist which provide reliable execution of workflow but they are not efficient in terms of execution time, as time is one of the major factors in

workflow scheduling. In this thesis work we will propose a hybrid approach for workflow scheduling which will provide the reliable execution of the workflow while minimizing makespan of the workflow.

1.7 Organization of Thesis

The rest of the thesis is organized as follows:

Chapter 2 – This chapter describes in detail the literature survey done to study the concept of workflow scheduling, Workflow Management System (WFMS), existing algorithms for workflow scheduling.

Chapter 3 – This chapter describes the problem analysis of the thesis work. It gives the gap analysis and problem statement.

Chapter 4 – This chapter gives the solution of the problem and design of the proposed solution.

Chapter 5 – This chapter focus on the implementation details and experimental results – description of CloudSim, Netbeans and snapshots of the Cloud Portal designed to study the technique.

Chapter 6 – This chapter gives the conclusion, contribution of the work done and future research work possible.

Chapter 2

Literature Survey

This chapter discuss about state of the art and research issues in workflow scheduling, architecture of workflow management system, workflow scheduling algorithms.

2.1 Workflow Management System (WFMS)

WFMS is a framework that supports definition, execution and management of workflow on distributed environment. While executing workflow job, different types of information like dependency constraints, data and resources required and so on need to be provided. WFMS provide user with a user-friendly platform where user can create, submit and execute their workflow applications without any difficulty and overhead [22].

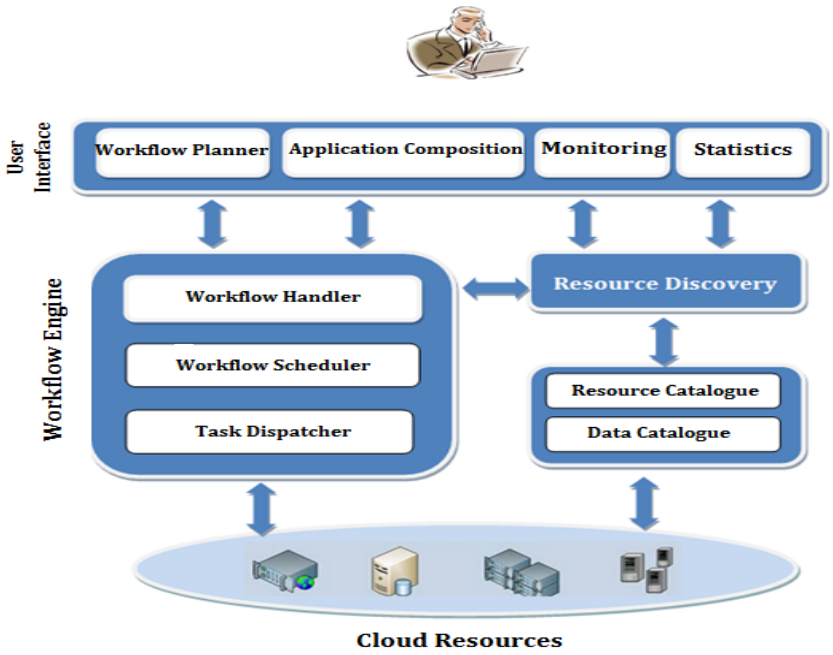


Figure 2.1: Workflow Management System [22]

The main components of WFMS are:-

i) User Interface

User Interface acts as a middleware between the user and the workflow engine. It is the entry point where user can create the workflow by providing dependency constraints and other details like data and resources required. Here user can submit

the job and get results. User can also monitor his job using user interface and can access the information regarding system like resources availability, bandwidth availability, storage availability etc.

ii) Resource Discovery

This component act as interface between workflow engine and information services component. It is used by workflow engine to find required desirable resource from available resources.

iii) Information Services

It is the system catalogue which contains information about all the resources available and data storage location.

iv) Workflow Engine

It is the heart of WFMS where actual execution and management of workflow take place. In workflow engine there are three sub-components: workflow handler, scheduler and task dispatcher. Workflow handler accepts the submitted workflow and transforms it into the objects which act as input to scheduler. Scheduler selects the optimal resource on the basis of scheduling algorithm and resource availability. Task dispatcher dispatches the task to allocated resources to get executed.

2.2 Workflow Scheduling Techniques

Workflow scheduling is used for the efficient execution of the workflow and to utilize the cloud resources in efficient way. It defines the placement policies, according to which task are located to resources. While mapping task with resources, we have to consider various performance metrics like time, cost, resource utilization, load balancing. A scheduling algorithm consists of three phases: (1) deciding the task mapping order, (2) scheduling task to resources, (3) task submission.

2.2.1 Workflow Scheduling Approaches

Workflow scheduling algorithms can be broadly classified into following categories [18] [19] [20] [21]:-

i) List Based Heuristics

A list based scheduling prioritizes the task of workflow and schedules the task on basis of their priorities. In this approach, there are two phases: Task Prioritizing and Resource Selection phase. In task prioritization phase, priority is assigned to each task using a rank function and tasks are sorted according to priority. In resource selection phase, tasks are mapped to optimal resource on the basis of priority.

ii) Genetic Based Scheduling (GA)

Genetic algorithms provide robust search technique that allows high quality of solution to be derived from large search space in polynomial time by using principle of evolution. A number of workflow scheduling algorithms use genetic algorithm to provide robust and good solution. There are four phases in genetic based algorithm: selection, crossover, mutation and fitness evaluation. In this we have to define a fitness function based upon user optimization requirement. Genetic algorithm works on the principle of survival of fittest thus giving result having highest fitness value i.e. good solutions according criteria.

iii) Ant Colony Optimization (ACO)

ACO is a meta-heuristic approach introduced by Dorigo [23] which uses foraging behavior of ants. It works by simulating the pheromone-depositing and following nature of ants and has been used in various optimization problems. The main advantage of ACO for workflow scheduling is that ACO make full use of instance based heuristic information.

iv) Particle Swarm Optimization (PSO)

Particle Swarm Optimization (PSO) is a self-adaptive global search based optimization technique developed by Kennedy and Eberhart [24]. The algorithm is similar to other population-based algorithms like genetic algorithms but, there is no direct re-combination of individuals of the population. PSO uses social behavior of the particles. In every generation, every particle adjusts its trajectory on the basis of its best position (pbest) and position of the best particle (gbest) of entire population. This approach increases the stochastic nature of the particle and converges quickly to global minima with a reasonable good solution. Experimental result shows the PSO

based workflow scheduling algorithms give better results as compare to Genetic and ACO based solutions.

2.2.2 Approaches for Reliable Workflow Scheduling

In cloud, a number of fault tolerance approaches are used for the reliable execution of the workflow. These approaches can be classified into two main categories: time redundancy and resource redundancy [25].

i) **Time Redundancy**: In this approach, weather a backup resource is assigned in advance for each running task in case failure occurs at primary, task is executed on the backup resource or task is rescheduled to a new resource in case fault occurs and overall execution time is extended.

ii) **Resource Redundancy**: In this approach, tasks are replicated to multiple resources such that all replicated copy of tasks run in parallel to tolerate a particular number of faults.

The above two categories can be further classified into following six categories:-

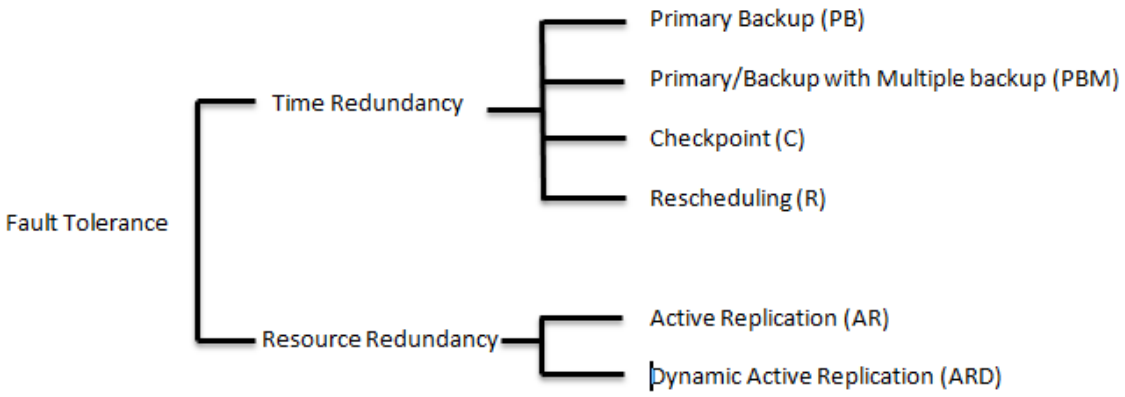


Figure 2.2: Various Fault tolerance Approaches in cloud [25]

i) **Checkpoint**

Checkpoint is a type of time redundancy technique which minimizes the wasted execution by periodically saving the resource state. In case failure occur task need to be executed from last checkpoint thus reducing extra computation.

ii) **Primary Backup with Overloading Backups**

In this approach, tasks are mapped to two resources: primary resource and backup resource. Initially task is executed on primary resource but in case of failure at primary, backup resource is activated to execute the task. This approach uses free time of resources to set backup to provide cost efficiency. This approach is able to handle only one fault.

iii) Primary/Backup with Multiple Backups

It is the extended version of original Primary/Backup approach in which we use more than one backup resource along with one primary resource. This approach is able to handle fault equal to number of backup replicas.

iv) Dynamic Rescheduling

In this approach, no advance reservation for resource is done. Whenever a failure occurs task is rescheduled to another machine for execution. This approach can handle any number or fault but may lead to long execution time.

v) Active Replication

In this approach, each task is replicated to n+1 resources to tolerate n number of failures and all replicated copies of task run in parallel.

vi) Active Replication with Dynamic Replicas

In this, tasks are replicated to different number of most reliable resources to achieve specified reliability requirement to execute in parallel.

2.3 Workflow Scheduling Algorithms for Makespan Optimization

There are various scheduling algorithm exist which try to optimize the makespan of workflow execution in cloud environment. These algorithms are discussed below:-

i) Topcuoglu et al. [26] proposed a new list based scheduling algorithm which targets to minimize makespan of the workflow execution. There are two phases in the algorithm: Task Prioritization and Resource Selection. In task prioritization phase, priority is assigned to tasks on the basis of a rank function. After that tasks are sorted on basis of priority. In resource allocation phase, tasks are allocated to a resource which can execute it in minimum possible time on basis of priority. This algorithm provides better results as compare to Myopic, min-min, max-min scheduling algorithms.

ii) Lin et al. [27] proposed a new scheduling algorithm known as Scalable Heterogeneous Earliest Finish Time First (SHEFT) which is an extension to Heterogeneous Earliest Finish Time (HEFT). In this algorithm, initial scheduling in done using HEFT algorithm, in case during execution if task is ready but allocated resource is busy then instead of waiting for resource to get free, task is allocated to available processor. Experimental results show that SHEFT not only outperforms HEFT in execution time but also help in resource to scale up elastically.

iii) Delavar et al. [28] gives a genetic based hybrid heuristic algorithm, HSGA for makespan optimization and load balancing which uses two stage initial population computation strategies. Firstly, it makes task prioritization in complex graph considering their impact on other task on basis of graph topology. Secondly, it combines the best-fit and round robin to make initial population which provides fast and good result.

iv) Liu et al. [29] proposed a novel Cost Time Compromised (CTC) scheduling algorithm which considers instance intensive cost constrained workflows based upon sub deadline distribution approach. The algorithm focus upon on minimizing the overall cost of execution including data transfer and computation cost while fulfilling user provided deadline. It also provide a just in time graph thus providing ability to user to change the constraint dynamically on the basis of requirement.

v) Wu et al. [30] presented a market oriented hierarchal scheduling strategy in which scheduling process is divided into two level of hierarchy: Service Level and Task Level. In case of service level, higher level scheduling takes place in which tasks are allocated to cloud services. In case of task level, lower level scheduling takes place in which tasks are allocated to virtual machines in local datacenters. At service level a package based random scheduling algorithm is used and at task level three different approaches (GA, PSO, ACO) are analyzed out of which ACO gives the best performance in terms of makespan and cost.

vi) Rahman et al. [19] presented a dynamic critical path based adaptive scheduling algorithm named DCP-G for makespan optimization. In DCP-G, critical path is calculated at every step to find critical tasks. A critical task is a task which lies on the critical path. After finding critical tasks, these tasks are mapped to optimal resources

first as compare to other tasks. The result shows that the algorithm works better as compare to static algorithms and avoid performance degradation in dynamically changing grid and cloud environment. The algorithm gives better performance as compare to HEFT, MinMin, MaxMin, Myopic algorithms.

vii) Barbosa et al. [31] gives a Heterogeneous Budget Constrained Scheduling (HBCS) algorithm which guarantees an execution cost within in the budget while minimizing makespan of the workflow. HBCS uses combination of HEFT and Loss-Gain algorithm. In this algorithm, firstly, tasks are sorted on the basis of priority assigned to them using a rank function. The algorithm makes use of Loss-Gain algorithm approach to find a suitable resource for task which efficient in terms of both cost and execution time.

viii) Pandey et al. [32] gives a scheduling algorithm based upon PSO technique. The main motive of this algorithm is to reduce overall cost of workflow execution in the cloud. Both execution cost and communication cost are considered while calculating overall cost of workflow. Experimental results show three times cost saving as compare to BRS scheduling algorithm.

ix) Liu et al. [33] proposed a scheduling algorithm based on ACO technique. In the proposed algorithm, various QoS parameter are considered including time, cost, security and reliability. In this user specify the required QoS threshold in SLA and use SLA monitoring module to monitor the running condition of cloud services.

x) Abrishami et al. [34] developed a deadline constrained workflow scheduling algorithm based upon Partial Critical Path (PCP) and proposed two new algorithms: IC-PCPD2 and IC-PCPC. IC-PCPD2 has two main phases: Deadline distribution and Planning. In the Distribution phase critical path is identified and deadline is distributed to all tasks on critical path. In the planning phase it allocate resources to task in such a way to utilize existing instance of computation services, in case it fails to fulfill deadline it will request for new instance. IC-PCP is a one phase algorithm which first calculates the critical path at every step and after that schedule the entire path on a computation service which can execute it within deadline with lowest execution cost.

2.4. Workflow Scheduling Algorithms for Reliable Workflow Execution

There are various scheduling algorithm exist which try to provide reliable workflow execution in cloud and grid environment. These algorithms are discussed below:-

i) Benoit et al. [35] proposes a fault tolerant scheduling algorithm based on the active replication approach called as Fault Tolerance Scheduling Algorithm (FTSA). In this algorithm combination of top level and bottom level approach is used for task prioritization. After that task are mapped to first n+1 processor which can execute task in minimum execution time to handle up to n faults.

ii) Benoit et al. [36] propose another reliable scheduling algorithm based on active replication known as Contention Aware Fault Tolerance (CAFT). The algorithm in an extension of FTSA with a difference that in this case, replicas are scheduled on first n+1 processors which hold maximum number of predecessor tasks replicas to reduce communication time.

iii) Zheng et al. [37] proposed two new fault tolerant scheduling algorithm: Minimum Replication Cost with Early Computation Time (MRC-ECT) and Minimum Completion Time with Least Replication Cost (MCT-LRC). Both algorithms are based upon Primary-Backup with Overloading approach. In this approach first primary copies of tasks are scheduled and then backup copies of task are schedule according to required goal. MRC-ECT as the name suggest will find an optimal schedule which will schedule the entire tasks in such a way that overall replication cost is minimum with earliest completion time. While MCT-LRC schedules tasks with least completion time and less cost of replication.

iv) Zhang et al. [38] proposed a reliable workflow scheduling algorithms which use combination of HEFT scheduling algorithm and active replication approach for the reliable workflow execution. In this paper, two algorithms are given: Scheduling with Over-Provisioning and Scheduling with Checkpoint Recovery. In Scheduling with over provisioning, first of all primary copies are mapped to processors using HEFT scheduling algorithm in which tasks are ranked and allocated to fastest available resource. Once all primary copies are assigned, replica copies are then mapped to processor in similar way. Scheduling with checkpoint recovery uses light weight checkpoint strategy is used to store the state or resource at different location. In case,

all replicas fail to complete task, task is rescheduled to new location and started from the last checkpoint.

v) Mills et al. [39] proposed an energy aware fault tolerant scheduling algorithm which uses the approach of active replication. In this algorithm, every task is replicated on two resources, first one is primary copy and second one is shadow copy. Both primary and shadow copy run in parallel but in case of shadow copy, Dynamic Voltage and Frequency Scaling (DVFS) technique is used to find lowest speed to run shadow copy to reduce energy consumption. In case, fault occurs at primary copy, shadow copy is resumed to execute at full speed to fulfill deadline of task. Experimental results show about 15-20% reduction in power consumption as compare to static replication algorithms.

vi) Lee et al. [40] proposed a new reliable scheduling algorithm based on rescheduling approach. In this algorithm, the entire tasks are mapped to resources on basis of static scheduling algorithm so as to reduce the makespan of application. Finishing time of each task is calculated on basis of resource allocated to them. Now, tasks are monitored to check whether any task is taking more time to execute than calculated one. If such task is found, it is rescheduled to new resource in order to get executed as fast as possible. This algorithm provides efficient use of cloud resources and minimizes increase in makespan.

vii) Lee et al. [41] proposed cost efficient and reliable workflow scheduling algorithm based upon the greedy approach. In Reliability with Profit Assurance Algorithm (RPA), an initial schedule is founded using greedy approach i.e. bottom level approach is used for task prioritization and task are allocated to resource which can execute it in minimum time. The initial schedule then go under replication process in which loss is estimated on the basis of two factors i.e. penalty due to increase in makespan and cost incurred due to usage of resource during extra time. If this loss is greater than cost of replication, then task is replicated otherwise it is not replicated. Thus, algorithm provides reliable execution in a cost efficient manner.

viii) Wang et al. [42] proposed a reliable scheduling algorithm with makespan optimization based upon Look Ahead Genetic Algorithm (LAGA). In this algorithm Reliability-Driven (RD) reputation is used to find out the timed dependent reliability

of resources. In LAGA two novel features are used: (1) it optimizes GA by new mutation operator based on resource priority and (2) It uses a novel evolution and evaluation approach for task resource mapping while task execution order is determined using max-min approach. In crossover step, algorithm will randomly select some pair of chromosomes with a probability, for each pair a random cutoff point is calculated and both upper and lower parts are exchanged. In mutation step, algorithm will randomly select a particular task and allocate it to a more reliable machine than the previous one. Experiment result shows that LAGA performs better as compare to Binary Genetic Algorithm (BGA), DLS and RDLS.

ix) Oliveira et al. [43] presented an adaptive scheduling heuristic for workflow scheduling which uses multi-objective optimization to find a trade-off solution between time, cost and reliability. The algorithm is divided into four parts: Greedy Scheduling, Cloud Activities Grouping, Cloud Activity Perform Algorithm and Load Balancing Algorithm. Dynamic allocation of resources is used to fulfill QoS constraints as per specified by the user.

x) Sampaio et al. [44] proposed dynamic scheduling algorithms based on proactive fault tolerance approach. In this algorithm, VM's are allocated to the resources on basis of the reliability of resources. In the paper, author proposes two new scheduling algorithms: Minimum Time Task Execution (MTTE) and Relaxed Time Task Execution (RTTE). In MTTE, as job arrived task are sorted on the basis of execution time and resources are also sorted on basis of reliability. From the sorted list first task is taken, a physical node that can execute the task with in minimum time and energy efficient way is allocated. In case of RTTE, same process is followed but a resource that can execute that particular task within deadline with least energy consumption is allocated first.

xi) Zhao et al. [45] proposed a new reliable workflow scheduling algorithm based on the dynamic active replication approach. The algorithm uses failure prediction of next time interval to calculate the number of replicas required for each task. Three scheduling strategies are proposed: (1)RR algorithm which uses least resource to meet reliability assigned task to group of high reliability processor whose aggregate reliability is greater than the required one , (2)DRR which is an extension of RR

which take both reliability and deadline into consideration while mapping task to resources, (3)Dynamic Reliable algorithm uses a messaging protocol to get the reliability of each processor during mapping phase to get error free reliability parameter and schedule task to group of resources whose aggregate reliability is greater than required one.

xii) Gao et al. [46] proposed an energy efficient fault tolerant scheduling algorithm. The algorithm consists of two phases: Static Scheduling and Dynamic Scheduling. In static scheduling phase, tasks are allocated to resources on bases of energy efficiency in such a way that all tasks get executed within required deadline. In dynamic scheduling phase, all VM's are monitored using sensors to predict failure. In case of fault, dynamic scheduler tries to reschedule the task to another VM so as to complete the task within required deadline. This algorithm tries to optimize energy consumption cost within required deadline constraint

xiii) Egwutuoha et al. [47] proposed a reliable scheduling algorithm based on the proactive fault tolerant approach. In this live VM migration technique is used for transferring tasks from unhealthy machine to new healthy machine. Sensors are used to predict the failure of the machine by monitoring CPU temperature, fan speed, memories and other parameter. A message is send to controller in case any parameter value exceeds its threshold. In case failure occur, a new machine is taken on lease and all the tasks from unhealthy machine is transferred to this newly machine. This approach is highly cost effective as no resource wastage occurs. Experimental results show a 30% reduction in execution cost as compare to other algorithms.

The various techniques mentioned above are summarized in the Table 2.1 and 2.2 given below. Existing reliable workflow scheduling algorithms are not efficient in terms of execution time.

Table 2.1: Comparison of various Workflow Scheduling Algorithms for Makespan Optimization

Algorithm Name	Method	Parameter	Finding	Environment
1. HEFT	List Based	Makespan	1. Optimizes makespan 2. Use bottom level approach for task prioritization	Grid
2. SHEFT	List Based	Makespan	1.Try to minimize makespan	Cloud

		Scalability	2. Give better result as compare to HEFT	
3. HSGA	Genetic	Makespan Load balancing	1. Effectively distribute load among resources 2. Best fit and round robin based population for fast results	Cloud
4. CTC algorithm	List Based	Cost, Time	1. Just in Time graph for user so that user can make changes to compromise between cost & time 2. Reduce conflicting and competition of services due to multiple concurrent instances	Cloud
5. Market oriented Hierarchal Scheduling	Hybrid	Makespan Cost, CPU time	1. ACO perform better as compare to GA and PSO at task level. 2. Both makespan and cost are optimized simultaneously.	Cloud
6. DCP-G	Critical Path Based	Time	1. Adaptive to exploit dynamic nature of cloud 2. Better result as compare to HEFT, GA, min-min, GRASP	Cloud, Grid
7. HBCS	List Based	Cost , Time	1.30% more cost saving as compare to HEFT 2. Best resources is selected for allocation in terms of time and cost	Cloud
8. PSO based Workflow Scheduling Algorithm	Particle Swarm Optimization	Cost	1. Consider both execution and communication cost 2. Good distribution of Workload	Grid
9. ACO based scheduling Algorithm	Ant colony Optimization	Reliability Response Time, Cost, Security	1. Multi-objective optimization 2. Satisfactory results	Cloud
10 IC-PCP And IC-PCPD2	List Based	Cost, Time	1. IC-PCPD2 try to utilize remaining time of existing computing resource 2. IC-PCP outperforms IC-PCPD2	Cloud

Table 2.2: Comparison of various Reliable Workflow Scheduling Algorithms

Algorithm Name	Approach Used	Parameter	Finding	Environment
1. FTSA	Active Replication	Reliability	1.Try to provide reliable execution	Grid
2. CAFT	Active Replication	Reliability, Makespan	1. Reduce time by allocation task on resources which have their predecessor task	Grid
3. MRC-ECT	List based, Primary backup with overloading	Reliability, Cost, Time	1. Try to minimize cost while optimizing time and provide reliable execution 2. Satisfactory results	Grid
4. MCT-LRC	List based, Primary backup with	Reliability, Time, Cost	1. Try to minimize makespan while reducing cost and provide reliable execution.	Grid

	overloading		2. Satisfactory results	
5. HEFT with Overprovisioning	List Based, Active Replication	Reliability, Cost, Time	1. Provide reliable Workflow execution 2. Optimizes makespan	Grid
6. Shadow Computing	List Based, Active Replication	Reliability, Energy	1. DVFS is used for energy efficiency. 2. Reduce 15% energy consumption	Cloud
7. RC^2 Algorithm	List Based	Reliability, Makespan	1. Use rescheduling approach for reliability 2. A task is reschedule only when it results in increase in makespan	Cloud
8. RPA	List Based, Active Replication	Reliability, Cost	1. Assure profit to service provider 2. Task replicated if replication cost is less than penalty cost	Cloud,Grid
9.LAGA	Genetic Based	Reliability	1. Use RD reputation to calculate reliability of resource 2. Use novel mutation and crossover operator 3. Better result as compare to BGA algorithm	Cloud
10. Provence Based Adaptive Scheduling	Greedy	Time, Cost, Reliability	1. Try to find a trade off solution between time, cost, reliability 2. Provide optimal schedule	Cloud
11. MTTE	Dynamic VM allocation	Reliability, Makespan, Energy	1. MTTE try to optimize makespan while maximize reliability	Cloud
12. RTTE	Dynamic VM allocation	Reliability, Makespan, Energy	1. RTTE try to optimize energy consumption while meeting deadline and reliability	
13. RR Algorithm	List Based, Dynamic Active Replication	Reliability	1. Reliable scheduling with least resource usage	Cloud
14. DRR Algorithm	List Based, Dynamic Active Replication	Reliability, Time	1. DRR also consider deadline with reliability	Cloud
15. Energy-Aware fault tolerant scheduling	List based, Rescheduling	Fault tolerant, Energy	1. Reschedule task in case of failure 2. Optimizes energy consumption while fulfilling deadline	Cloud
16. Cost oriented proactive fault tolerant scheduling	List based, Live VM migration	Fault tolerant, Cost	1. Proactive Fault Tolerance 2. No extra cost for spare resource	Cloud

Chapter 3

Problem Statement

In previous chapter we have analyzed various works done in the field of workflow scheduling in cloud and grids. In this chapter, we will focus upon the gaps in the literature review that has been reviewed

3.1. Gap Analysis

Based on literature survey done in previous chapter, there are a number of methods which provide reliable execution of workflow but none of them are efficient in terms of time and algorithms which consider time does not provide or ensure highly reliable execution.

Table 3.1 Analysis of various Reliable Workflow Scheduling Algorithms

S.No	Algorithm Name	Fault Tolerance Technique	Parameter	Advantage	Disadvantage
1.	FTSA	Active Replication	Reliability	• Provide reliable execution.	• Not efficient in terms of time • Require N+1 replication for handling N faults
2.	CAFT	Active Replication	Reliability, Makespan	• Reliable execution with makespan optimization.	• Not efficient in terms of time • Require N+1 replication for handling N faults
3.	MRC-ECT	Primary backup with overloading	Reliability, Time, Cost	• Cost efficient as no resource wastage due to replication.	• Delay in execution in case fault occur
4.	MCT-LRC	Primary backup with overloading	Reliability, Time, Cost	• Cost efficient as no resource wastage due to replication.	• Delay in execution in case fault occur
5.	HEFT with Overprovis-ioning	Active Replication	Reliability,	• Optimizes makespan	• Require N+1 replica for handling N faults
6.	Shadow Computing	Active Replication	Reliability, Energy	• Reduce 15% energy consumption.	• Inefficient in terms of time and cost
7.	RC^2 Algorithm	Dynamic Rescheduling	Reliability, Makespan	• Cost efficient	• In case of fault, delay in execution
8.	RPA	Active Replication	Reliability, Cost	• Cost efficient • Only necessary tasks	• Not efficient in terms of time.

				are replicated.	• Job failure in case faults occurs at task which is not replicated.
9.	LAGA	Genetic Based	Reliability	• Cost efficient • No resource wastage	• Job failure in case fault occurs.
10.	Provence Based Adaptive Scheduling	Greedy Based	Time, Cost , Reliability	• Try to find a trade off solution between time, cost and reliability.	• Job failure in case fault occurs.
11.	MTTE	Dynamic VM allocation	Reliability, Makespan, Energy	• Optimize makespan and energy consumption • Cost efficient due to no replication	• Job failure in case VM failure occurs during task execution
12.	RTTE	Dynamic VM allocation	Reliability, Makespan, Energy	• Optimize energy consumption while meeting deadline • Cost Efficient due to no replication	• Job failure in case VM failure occurs during task execution
13.	RR	List Based, Dynamic Active Replication	Reliability	• Less resource usage for replication.	• Not efficient in terms of time.
14	DRR	List Based, Dynamic Active Replication	Reliability	• Less resource usage for replication. • Fulfill deadline.	• Not efficient in terms of time.
15.	Energy-Aware fault tolerant scheduling	Rescheduling	Fault tolerant, Energy	• No resource wastage due to replication. • Optimizes energy consumption.	• Delay in execution in case of fault.
16.	Cost oriented proactive fault tolerant scheduling	List based, Live VM migration	Fault tolerant, Cost	• No extra cost for spare resource.	• Application fails if fault is not predicted in advance.

3.2. Problem Statement

Most of the workflow applications hosted by cloud are real time applications which require high reliability and fast execution. In cloud, a number of failures occur due to various faults like hardware fault, software faults and so on. To efficiently execute these workflows and utilize the cloud resources in an appropriate way, scheduling policies need to be implemented. Currently available reliable workflow scheduling algorithms try to provide reliable execution but these algorithms are not efficient in terms of time.

To find the best schedule for mapping workflow tasks on the cloud resources that provide reliable execution while minimizing execution time is a well known NP-complete problem. The aim of this thesis work is to propose a new efficient workflow scheduling algorithm which will provide reliable execution of the workflow in such a way that execution time is minimized.

3.3 Thesis Objectives

The objectives of thesis are as follows:-

i) To study the existing workflow scheduling algorithms for makespan optimization and reliable workflow execution.

ii) To propose and design an efficient workflow scheduling approach for time efficient and reliable workflow execution in Cloud

iii) To implement and validate the proposed approach in Cloud Computing.

Chapter 4

Proposed Work

In this chapter we, will propose a hybrid approach for workflow scheduling which will provide the reliable execution of the workflow while minimizing the makespan of the workflow.

4.1. Assumptions and System Models

i) Resource Reliability Model

We assume that our platform consist of hardware and software with heterogeneous configurations. Thus, each processor will take different time to execute a particular task and it will cost accordingly. The mean time between failures (MTBF) of every processor is also different from one another. As per in [48], we consider that no fault will occur while processor is idle. Reliability of resource is the probability that no failure occur on resource p_j during time period T, this can be calculated using the memory less property of Poisson distribution. The formulae to calculate the reliability is given as [48]:-

$$Rel(t, p_j) = e^{-\lambda_j ET(t,p_j)} \times e^{-\lambda_j \sum_{t_k \in on(p_j)} ET(t_k,p_j)} \quad (1)$$

$$Rel(t, p_j) = R(t, p_j) \times \prod_{T_k \in on(p_j)} R(t_k, p_j) \quad (2)$$

Where λ_j is the expected number of fault occurrences in unit time for resource p_j.

t_i is the current task.

t_k is the set of previously allocated tasks on resource p_j.

ii) Workflow Model

A workflow application is generally represented as a Directed Acyclic Graph (DAG) which is a directed graph with no cycles such as W (N,E) where N represent a set of nodes or task and E represent set of edges which represents dependency among tasks. In workflow execution we have to maintain parent child relationship

such that a child task cannot be executed before its parent task. A task with no parent task is called as *entry task* and a task without a child task is called as *exit task.*

iii) Time Model

Makespan (T_{total}) of workflow is defined as the total time required for the complete execution of the workflow. It is the actual finish time (AFT) of exit task. It consists of total execution and data transfer time for each task in the workflow. Execution time depends upon the workload of task and resource allocated while data transfer time depends upon the network latency and data size to be transferred

$$T_{total} = \max AFT (T_{exit}) \quad (3)$$

iv) Reliability of Workflow

Suppose, reliability required by user is R (for example, .98). We will calculate the sub-reliability of all the sub-tasks in the workflow using the below given formulae [48]. We assume that reliability of all the tasks in the workflow is equal.

$$r = \sqrt[n]{R} \quad (4)$$

4.2. Proposed MCTR Algorithm

A workflow scheduling algorithm consists of two phases: - Task Prioritization and Resource Allocation. In task prioritization phase, tasks are sorted according to some rank value or function and in resource allocation phase, sorted task are mapped to resources one by one.

In existing algorithms for reliable workflow scheduling, researchers have used bottom level approach for task prioritization. But there are more efficient approaches are available like critical path technique. Rahman et al. [19] proposed a critical path based approach for workflow scheduling. In this algorithm, critical path is calculated at every step to identify critical tasks. Experiment results show that this approach gives better results in term of time as compare to other approaches.

For reliable execution of workflow, dynamic active replication is effective approach which provides reliable execution of the workflow without violation the deadline of

workflow execution. Zhao et al. [46] provide a solution on how to use least number of replication to achieve desired reliability in workflow but it does not consider execution time taken by the resource while mapping tasks to resources.

In the proposed Minimum Completion Time and Reliable (MCTR) algorithm, we will use the above two approaches for makespan optimization and reliable workflow execution along with considering execution time taken by resource to execute a particular task to provide time efficient and reliable workflow scheduling algorithm for cloud computing.

MCTR Algorithm

1: Input: Workflow W (T, E), Task Dependency list, resource set P, Reliability R.
2: begin
3: Calculate sub-reliability of tasks by using formulae r= $\sqrt[n]{R}$
4: for all t ∈ T of workflow W
5: Calculate average execution and communication time acc to formulae (5) &(6)
6: end for
7: for all t ∈ T of workflow W
8: Calculate EST for t according to formulae (7)
9: end for
10: Calculate CPL using formulae (8)
11: for all t ∈ T of workflow W
12: Calculate LST for t using formulae (9)
13: end for
14: while all task in T are not scheduled do
15: Task list ← get unscheduled ready tasks for workflow W
16: Schedule_task (Task list, P, r)
17: Update Task Dependency list
18: end while
19: end
20: PROCEDURE: Schedule_task
21: Input: Task list, resource set P, sub reliability of tasks r
22: begin
23: while Task list not empty do
24: t_i ← Get critical Task from all task of Task list
25: for each resource p_j ∈ P do
26: P = rel (p_j, t_i) // calculate the reliability of each resource for execution task t_i using formulae (2)
27: end for
28: sort (P, r, t) // sort resources on the basis of reliability and time as per section 4.2 (ii)
29: fp ← 1.0, counter ← 0, j ← 0 // fp is probability of resource failure

30: while (1 - fp) < r && j <= m do // where m is number of available resources
31: j++;
32: counter = counter + 1
33: fp = fp x [1 – rel(P_j, T_i)]
34: end while
35: n ← counter, counter ← 0 // n is the number of replication factor
36: Schedule t_i on first n resources from sorted list
37: Remove t_i on task list
38: for all t ∈ T of workflow W
39: Calculate EST for t
40: end for
41: Calculate CPL
42: for all unscheduled t ∈ T of workflow W
43: Calculate LST for t
44: end for
45: end while
46: end

In the above proposed algorithm line 1-20 give working of task prioritization phase and line 21-48 gives working of resource allocation phase

i) Task Prioritization Phase

In critical path approach, lower and upper bound of starting time of tasks is defined as Earliest Start Time (EST) and Latest Start Time (LST). A task is said to be in critical path whose EST is equal to LST. Delay in execution of a critical task effect the overall execution time of the workflow. Therefore, a critical task should be mapped first on best resource for it. This process is repeated till we have all the tasks scheduled.

In task prioritization phase, initially sub-reliability of each task is calculated using formulae (2) and after that minimum execution and communication time for each task is calculated (line 5-7) using formulae (5) & (6).

$$ET(t, p_i) = \frac{Task\_Size(t)}{max_{i \in Resources} \{Capacity(p_i)\}} \quad (5)$$

$$CT(t, p_i) = \frac{Data\_Size(t)}{max_{i \in Resources} \{bandwidth(p_i)\}} \quad (6)$$

Where, capacity refers to processing capacity of p_i

Whenever a task t is allocated to a particular resource, the values of ET(t,p_i) and CT(t, p_i) are updated accordingly. On the basis of above two values, Estimated Start

Time(EST) for each task on a resource p_k in the workflow is calculated using below given formulae.

$$EST(t, p_k) = \max_{1 \le i \le p} \{ \max_{1 \le j \le m} \{ EST(t_i, p_j) + ET(t_i, p_j) + CT(p_j, p_k) \} \} \quad (7)$$

where t has p parents, t_i is the i^{th} parent task. As we are using active replication technique for reliable execution, we have to consider all replica of task while calculating the EST. In above m is the number of replicas on which task t_i is replicated. In case of unscheduled tasks, value of m is equal to one.

CT (p_j, p_k) = 0 if $p_j = p_k$

CT (p_j, p_k) = CT (t, p_k) : if t_j and t are not scheduled.

Once EST of all task are computed, the Critical Path Length (CPL) is calculated which the length of partially mapped workflow using below given formulae.

$$CPL = \max_{1 \le i \le n} \{ \max_{1 \le j \le m} \{ EST(t_i, p_j) + ET(t_i, p_j) \} \} \quad (8)$$

Where n is total no of task in the workflow and a task t_i is replicated on m resources. For unscheduled task we take number of replication equal to one.

After calculating CPL, Latest Start Time (LST) for all the unscheduled tasks is calculated by traversing graph in reverse direction using below given formulae [19].

$$LST(t, p_k) = \min_{1 \le i \le c} \{ LST(t_i, p_j) - ET(t_i, p_k) - CT(p_k, p_j) \} \quad (9)$$

Where t has c child tasks, t_i is the i^{th} child of the task t.

As LST is calculated for the unscheduled tasks, we need not to consider replication factor in the above formulae for LST calculation.

LST (t,p_i) = CPL-ET (t) ; if t is exit task.

CT (p_k, p_i) = 0; if $p_k = p_i$

CT (p_k, p_i) = CT(t, , p_i); if t_j and t are not scheduled

A task is considered to be in critical path if it's EST and LST values are equal. To reduce execution time a task is selected for mapping which is on the critical path and has no unmapped parents.

ii) Resource Allocation Phase

In this, reliability of each resource for executing the critical task t_i is calculated using formulae (2).

After that, resources are sorted on basis of their reliability and time so as to get the resource with least execution time and highest reliability at first position in sorted list. For example, below given Table 4.1 and 4.2 shows the list of resources before sorting and after sorting.

Table 4.1 List of Resources before Sorting

Resources	Reliability	Execution time for task T_i
P_1	.97	5
P_2	.96	4
P_3	.97	4
P_4	.98	5
P_5	.98	4

Table 4.2 List of Resources after Sorting

Resources	Reliability	Execution time for task T_i
P_5	.98	4
P_4	.98	5
P_3	.97	4
P_1	.97	5
P_2	.96	4

In the above Table 4.1, we have five resources, each have different reliability value for executing task t_i and each will take different time to execute task t_i , both execution time and data transfer time is considered for calculating execution time of the task. We will sort the resource list given in Table 4.1 firstly, according to the reliability value and secondly, according to the execution time so as to get the sorted list of resources so as to get a resource with highest reliability and least execution time at first place in sorted list as shown in Table 4.2.

In line 29-34, replication factor (n) is calculated in such a way to get least number of resources whose aggregate reliability is greater that required reliability level. Once value of n is calculated, the task T_i is mapped on the first n resources from the sorted list of resources.

In case value of n is greater than available resources, all available resources are used to achieve maximum reliability level possible.

4.3. Flow Chart of Proposed MCTR Algorithm

We have explained the working of the proposed MCTR scheduling algorithm with the help of flow chart given below.

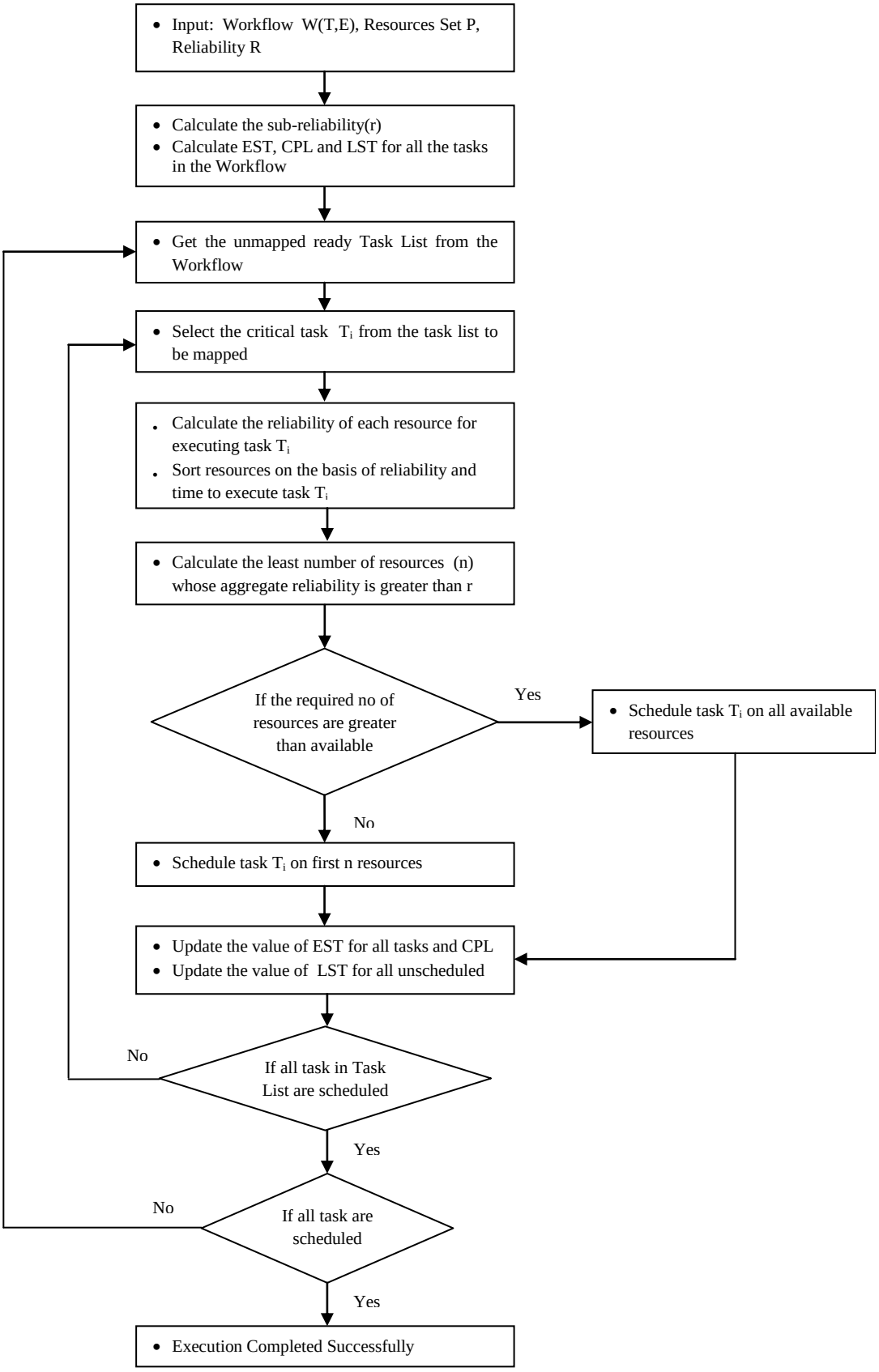


Figure 4.1: Flow Chart of Proposed MCTR Algorithm

Chapter 5

Implementation Details and Experiment Results

This chapter gives details about tools used for setting cloud environment, implementation of the proposed workflow scheduling algorithm and the experiment results gathered.

5.1 Tools for setting Cloud Environment

We have used CloudSim simulator for setting cloud environment to analyze our proposed algorithm.

i) CloudSim

CloudSim is a simulation framework that allows modeling and simulation of cloud computing infrastructure and services. CloudSim provides an easy-to-use and control cloud environment where researcher and developers can analyze the performance of their new applications. CloudSim supports modeling of infrastructure and behavior of cloud system components like VM's, data centers etc. without much effort. The two main advantages of using CloudSim are [49]:

- Time Effectiveness: It require very less time to setup cloud environment to test application performance.
- Flexibility and Applicability: Researchers can test their application for different cloud deployment models like public, private or hybrid with little extra overhead.

CloudSim provide the facility of implementing our own new scheduling policies by just extending some classes of CloudSim. The main features of CloudSim are:-

- Provide support for modeling and simulation of cloud computing environment on a single computing node like a laptop.
- Provide support for modeling service brokers, allocation and provisioning policies.
- Provide support for modeling network connection between different simulated cloud components like VM's, Data centers.
- Provide support for simulating Hybrid Cloud.
- Provide a virtualization engine that support management of virtualized services on a data centers.

- Provide support for two type of processing allocation policies: Time Shared and Space Shared.

Below given figure 5.1 gives the layered architecture of CloudSim.

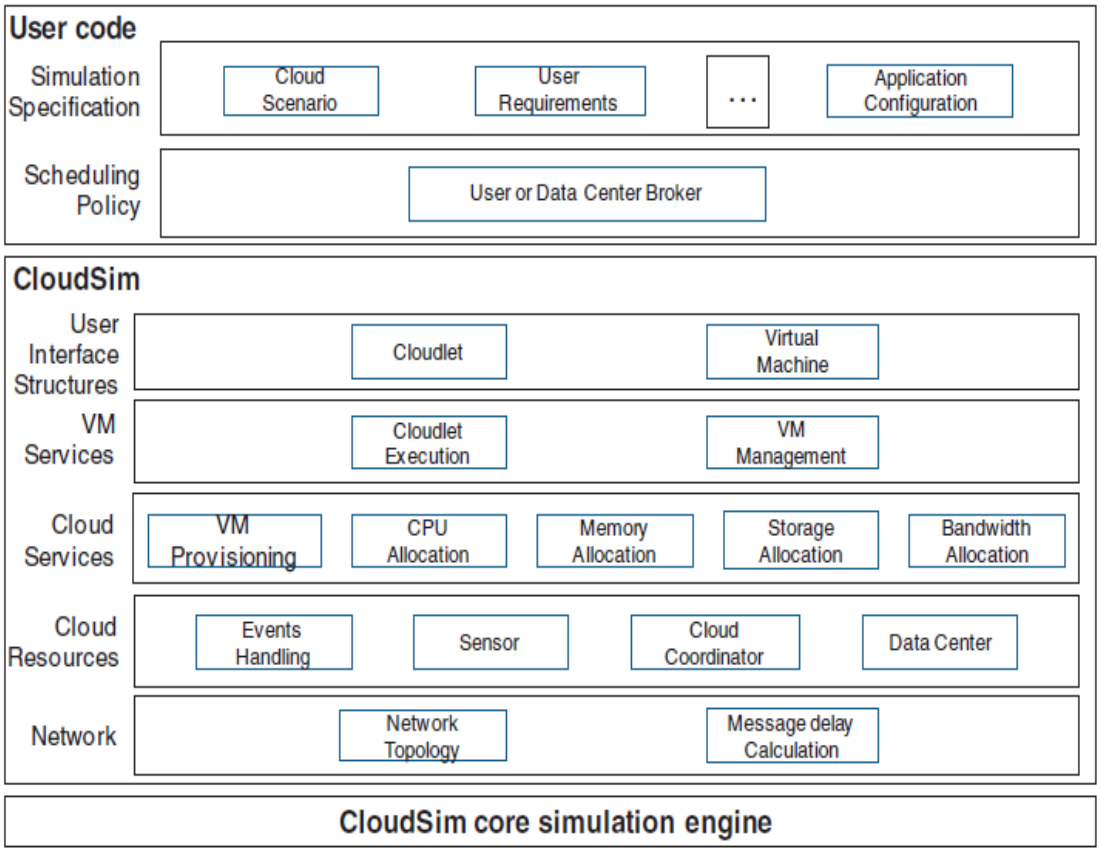


Figure 5.1: Architecture of CloudSim[49]

ii) NetBeans IDE

NetBeans is an open source freely available integrated development environment (IDE) which provides multi language software development support. Netbeans provide a user-friendly interface for developing applications in java and other programming languages. Netbeans is a powerful IDE developed specially for JAVA programmers which supports technologies like JDK, JavaEE, JavaFX. It provide user with a source code editor, compiler and build in automation tools. It is written in JAVA and compatible with many operating systems like Windows, Linux, and Solaris. CloudSim toolkit can easily be integrated with Netbeans easily without any difficulty [50].

iii) Oracle Database

We have used Oracle 10g for storing and managing the database of our portal. Oracle database commonly referred to as oracle RDBMS is an object-relational database management system produced and marketed by Oracle Corporation. It stores data in form of objects and relational tables. Structured Query Language (SQL) is used to access the database [51].

5.2 Implementation Details

A cloud portal has been developed to implement our proposed time efficient reliable workflow scheduling algorithm using CloudSim Toolkit and Netbeans IDE.

i) Cloud Portal Snapshots

Figure 5.2 gives the view of the Login Page, where user has to enter the username and password to access the Cloud Portal. In case, user enters wrong username or password it will give an error message as shown in Figure 5.3.

Figure 5.2: Login Page

Figure 5.3: Login Failed Page

In the portal, new user can register by providing his details as per shown in Figure 5.4. Validations are applied on each data field to check data consistency. In case, user enters wrong data in any of the field, an error message is displayed as per shown in Figure 5.5.

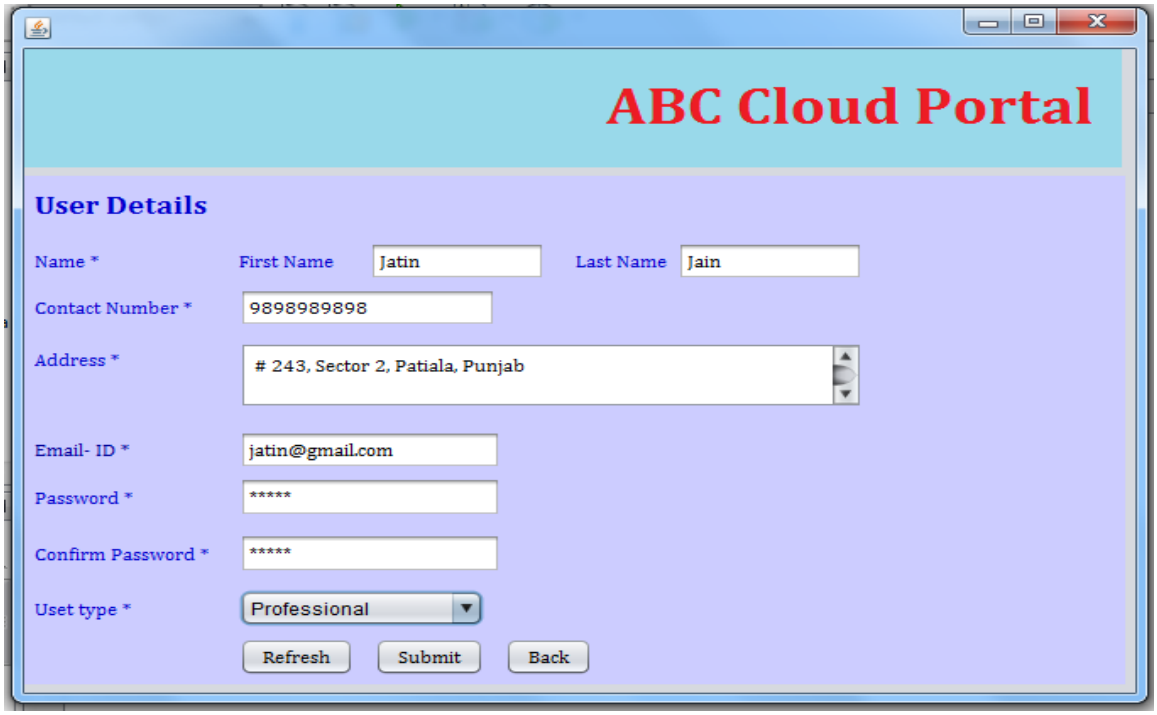


Figure 5.4: New User Registration Page

Figure 5.5: New User Registration Error Page

User will get a registeration successfully message in case user have filled all neccesary detials as shown in figure 5.6.

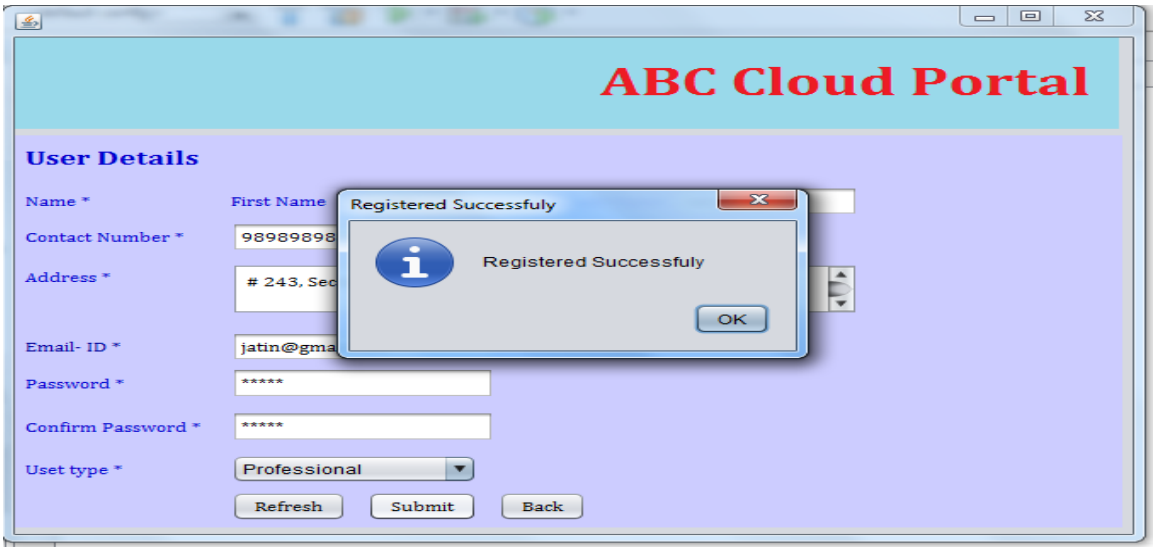


Figure 5.6: User Registered Successfully

Once successful login, user can use the services provided by the portal like manage his account, use portal to generate reliable schedule. Figure 5.7 give the view of service page

of our portal where user enter details about the workflow and required SLA (reliability level) and submit the workflow job.

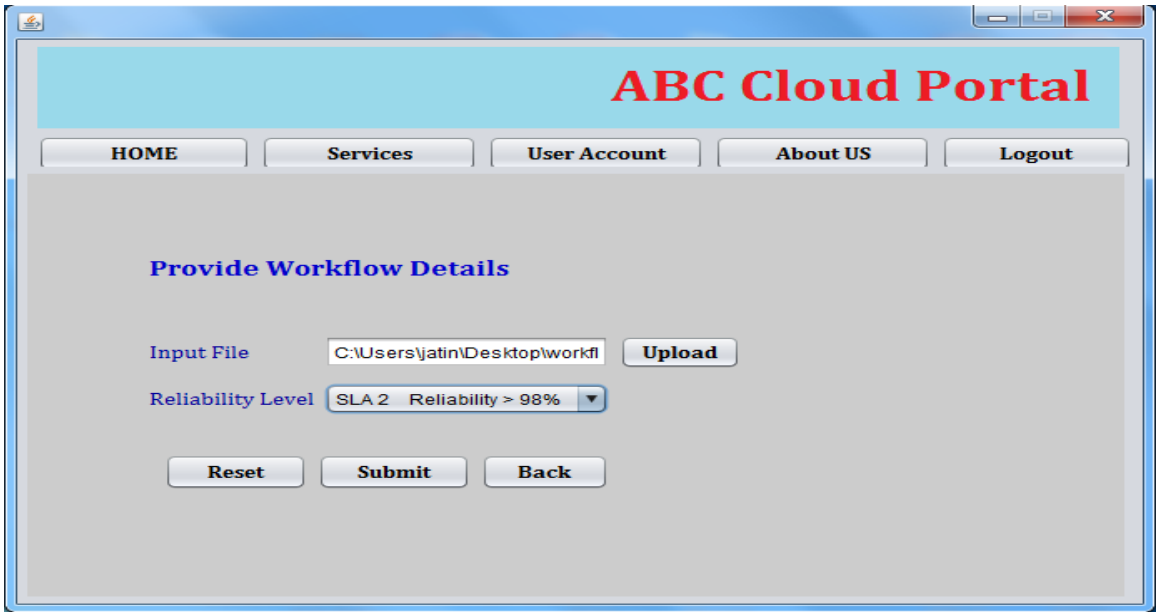


Figure 5.7: Workflow Details Submission Page

On successful submission of job, the proposed MCTR algorithm runs at the back end in CloudSim to generate the time efficient reliable schedule. The processing of CloudSim is shown in Figure 5.8, 5.9 and 5.10.

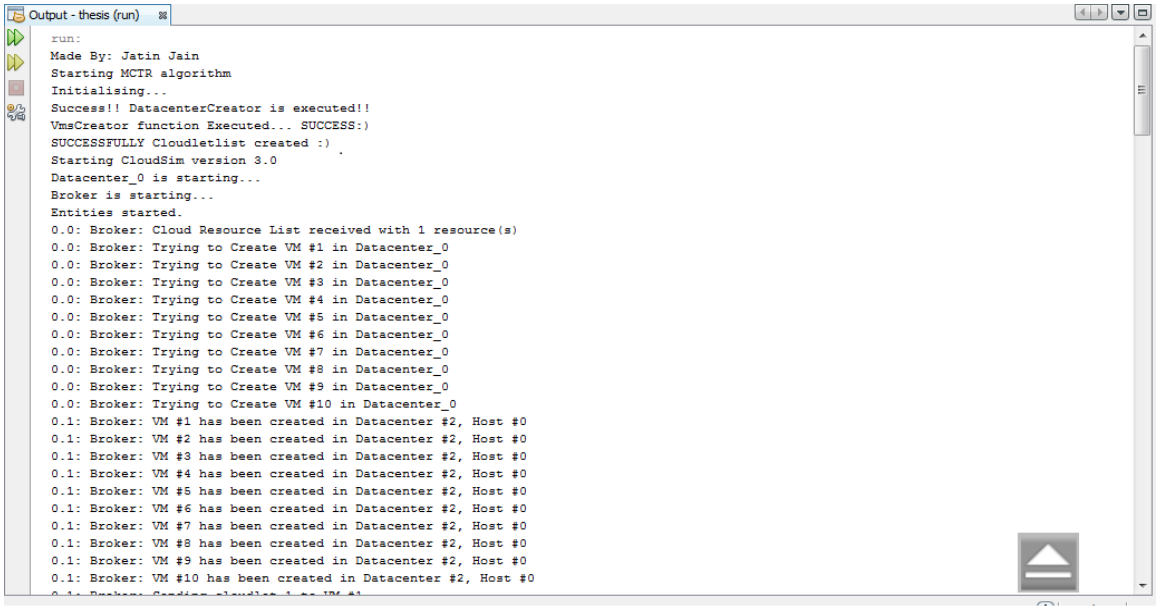


Figure 5.8: Processing in CloudSim 1

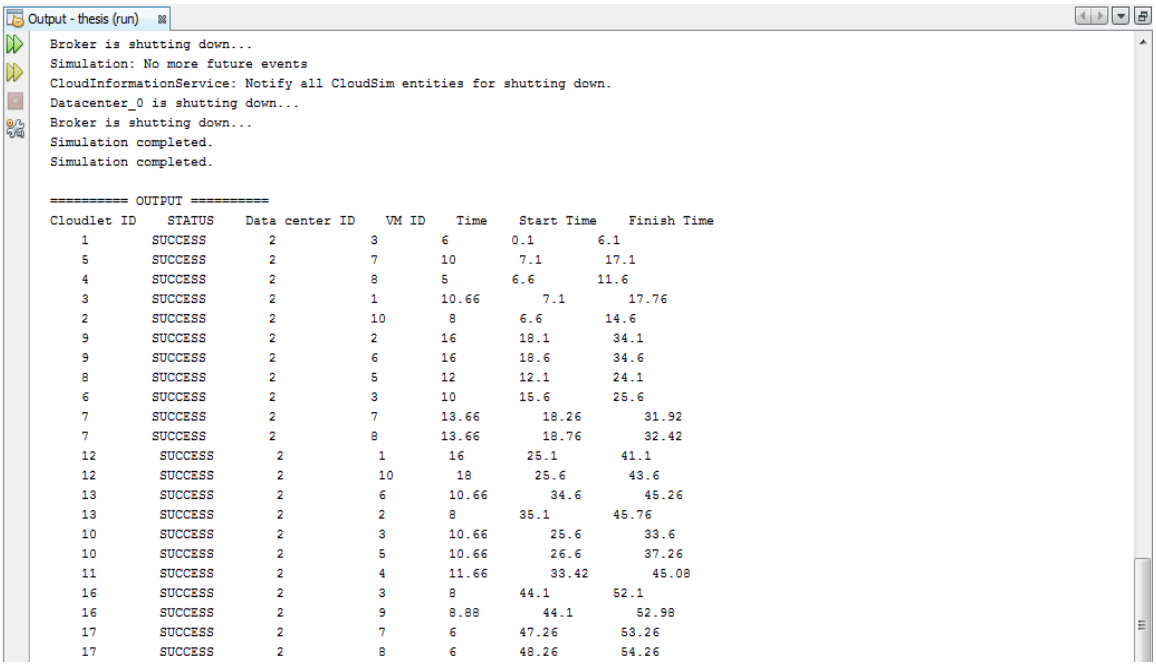


Figure 5.9: Processing in CloudSim 2

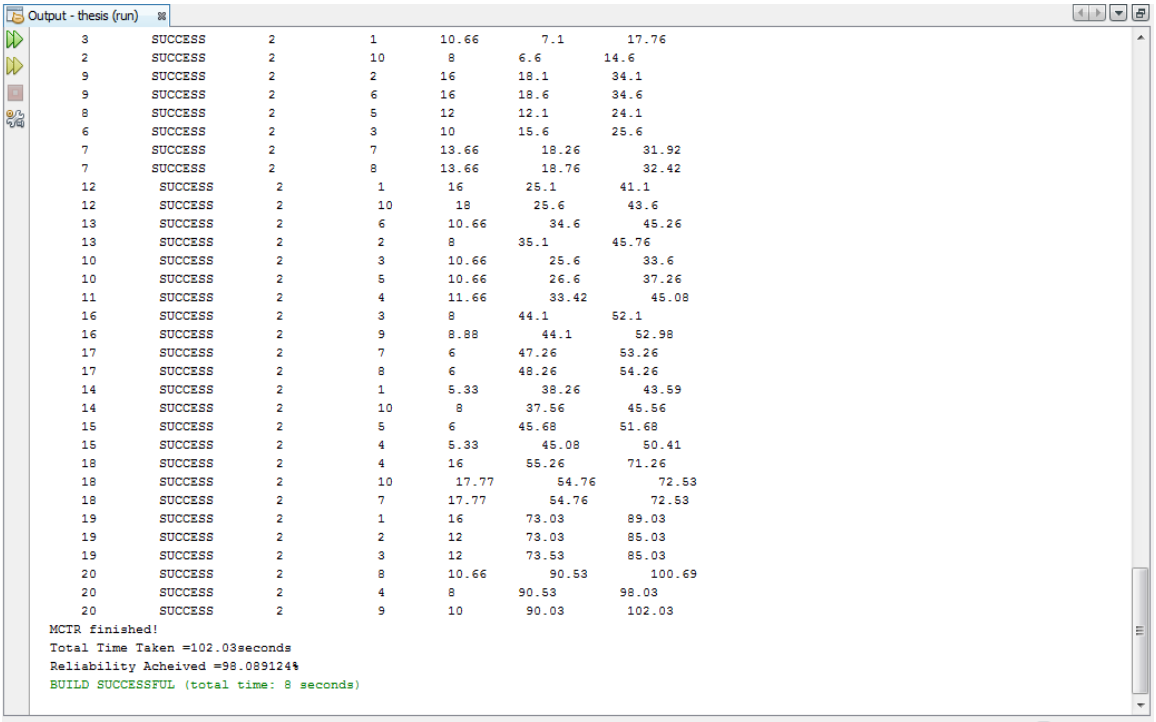


Figure 5.10: Processing in CloudSim 3

Once CloudSim generate the required schedule, user will get the output as shown in the given below figure 5.11.

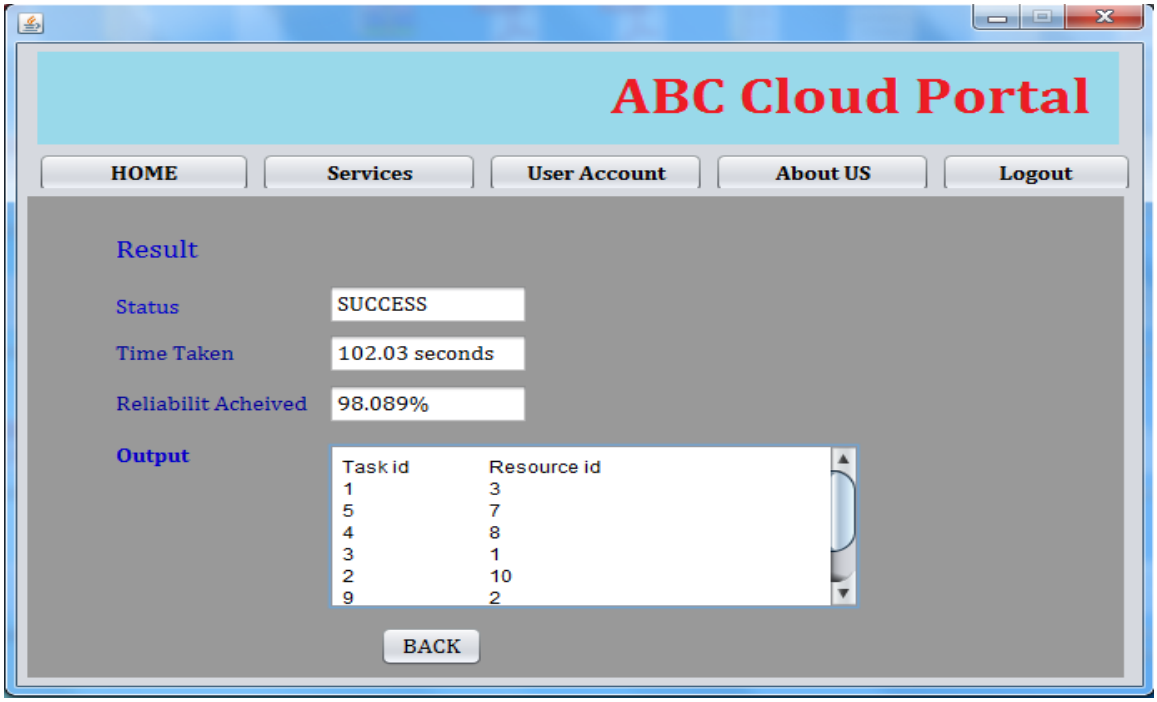


Figure 5.11: Result Page

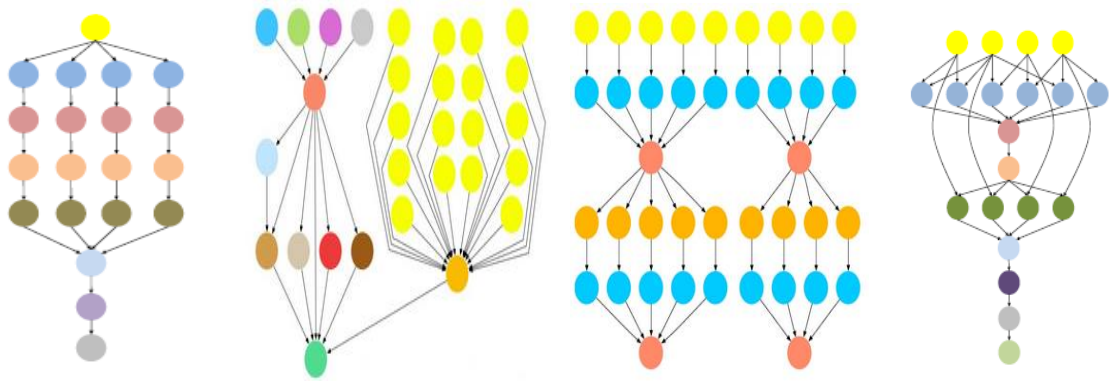
5.3 Experimental Results

The experiments are conducted considering cloud characteristics with respect to heterogeneity in resource properties and various properties for applications. We have used CloudSim simulator to implement cloud environment to test the proposed MCTR algorithm. The simulation parameters are listed in Table 5.1. We have compared our proposed algorithm with the RR algorithm.

Table 5.1: Simulation Parameters

Parameter	Value
No of Tasks	20-50
Tasks Length	10-40 X 10^3 MIPS
Data Size	20-200 MB
No of Resources	10-30
Failure Rate	1-3 X10^{-4}
Resource Speed	1000-2000 MIPS
Bandwidth	10-100 MBPS

We have considered four different types of scientific workflows Epigenomics, SIPHT, LIGO and Montage to test the proposed algorithm. The average results are taken for comparison. We have compared our proposed algorithm on the basis of makespan and reliability achieved.



(a) Type 1: Epigenomics (b) Type 2: SIPHT (c) Type 3: LIGO (d) Type 4: Montage

Figure 5.12: Structure of Realistic Scientific Workflows [24].

i) Test Case 1

In this case, we have considered four different type of workflow as shown in above Figure 5.12. The total number of tasks in each workflow is 20 and total numbers of resources taken are 10. The required reliability level for each workflow is 98%.

Table 5.2: Simulation Results for Test Case 1

	Makespan (in seconds)		Reliability Achieved (in %)	
	RR Algorithm	MCTR Algorithm	RR Algorithm	MCTR Algorithm
Workflow 1	110	102	98.056	98.042
Workflow 2	112	107	98.029	98.017
Workflow 3	107	101	98.035	98.026
Workflow 4	119	115	98.019	98.012

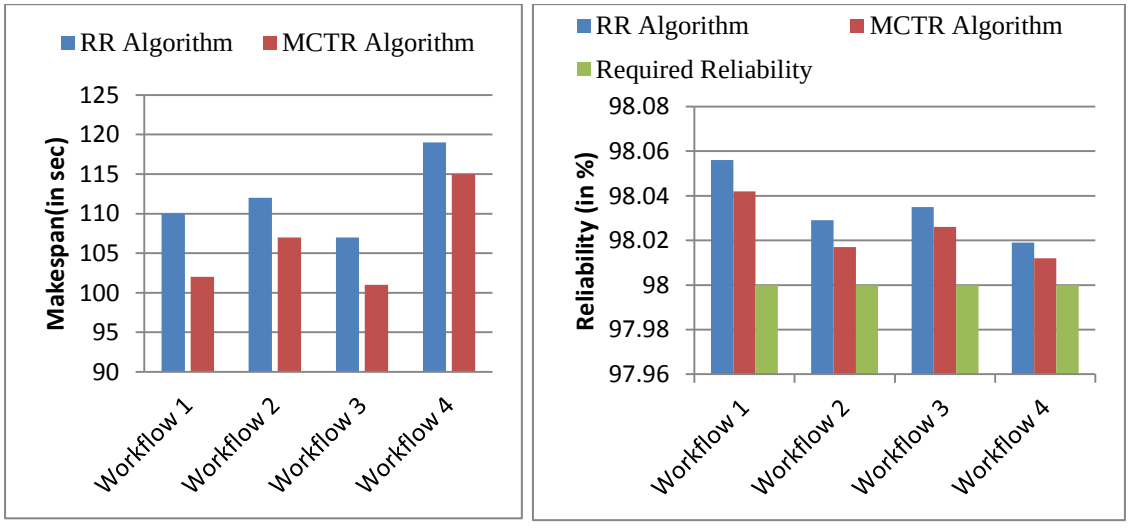


Figure 5.13: Graph of Simulation Results for Test Case 1

The above graph in Figure 5.13 shows that the proposed MCTR algorithm reduces the total execution time of the workflow as compare to the RR algorithm. The reliability achieved by the MCTR algorithm is less than RR algorithm but greater than the required reliability level.

ii) Test Case 2

In this case, we have taken four different types of workflow as in Figure 5.12. The total number of tasks in each workflow is 30 and total numbers of resources taken are 20. The required reliability level for each workflow is 96%.

Table 5.3: Simulation Results for Test Case 2

	Makespan (in seconds)		Reliability Achieved (in %)	
	RR Algorithm	MCTR Algorithm	RR Algorithm	MCTR Algorithm
Workflow 1	198	189	96.076	96.052
Workflow 2	211	203	96.051	96.036
Workflow 3	205	198	96.054	96.049
Workflow 4	219	216	96.025	96.011

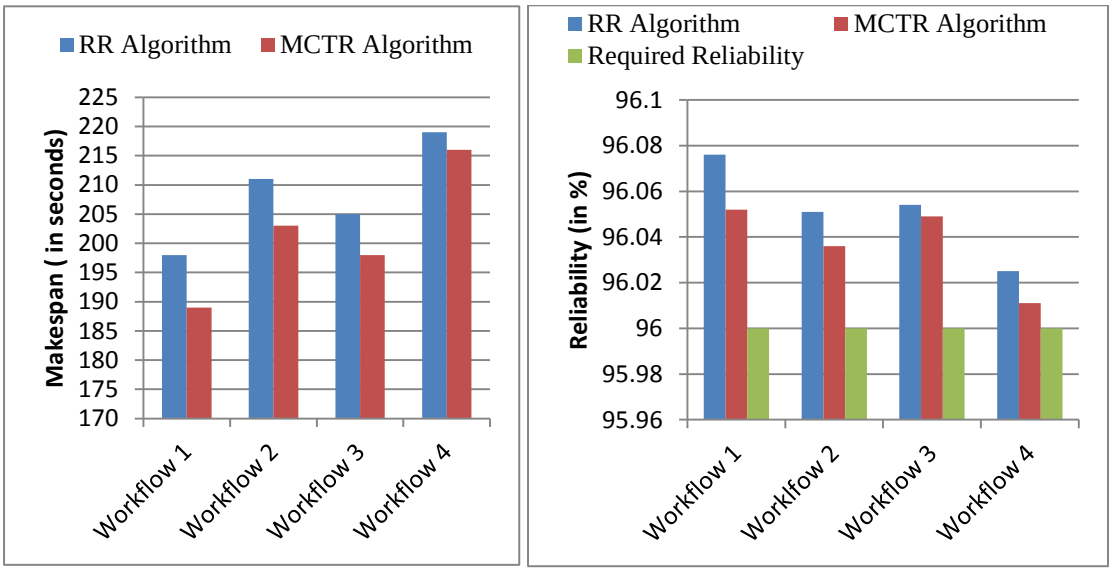


Figure 5.14: Graph of Simulation Results for Case 2

The above graph in Figure 5.14 shows the similar result as in Test Case 1. The proposed MCTR algorithm reduces makespan of the workflow as compare to RR algorithm while achieving the reliability greater than user required reliability level.

iii) Test Case 3

In this case, we have again considered four different types of workflow shown in Figure 5.12. The total number of tasks in each workflow is 40 and total numbers of resources taken are 20. The required reliability level for each workflow is 97%.

Table 5.4: Simulation Results for Test Case 3

	Makespan (in seconds)		Reliability Achieved (in %)	
	RR Algorithm	MCTR Algorithm	RR Algorithm	MCTR Algorithm
Workflow 1	253	245	97.089	97.071
Workflow 2	267	261	97.041	97.024
Workflow 3	260	252	97.059	97.044
Workflow 4	281	280	97.029	97.021

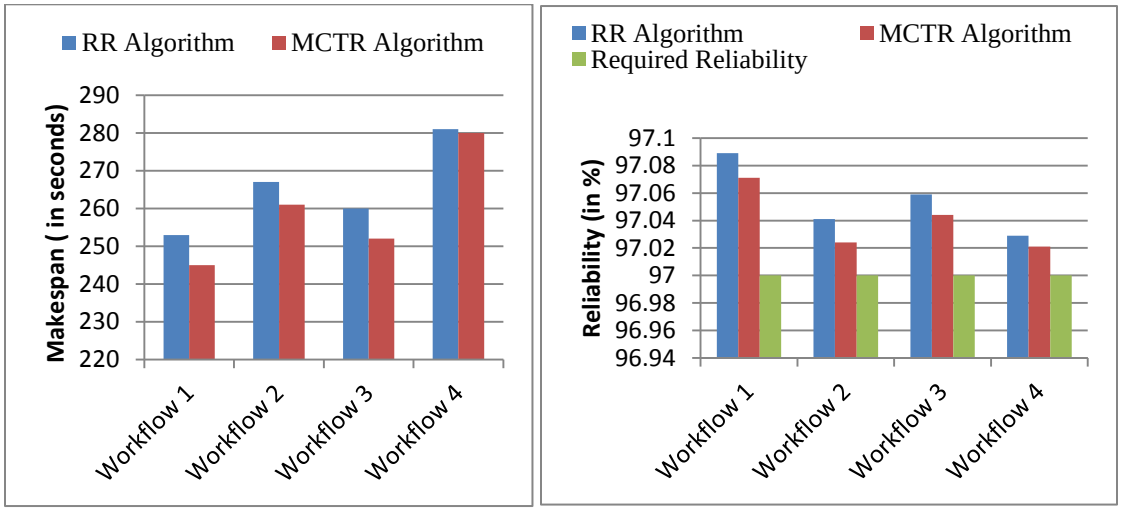


Figure 5.15: Graph of Simulation Results for Test Case 3

The simulation results showed in above graph in Figure 5.15 shows the similar result as in previous test cases. The proposed MCTR algorithm reduces the makespan of the workflow while providing reliable workflow execution as compare to RR algorithm. Reliability provided by proposed MCTR algorithm is less than RR algorithm but greater than the required reliability level.

iv) Test Case 4:

In this case, we have again considered four different types of workflow. The total number of tasks in each workflow is 50 and total numbers of resources taken are 30. The required reliability level for each workflow is 98%.

Table 5.5: Simulation Results for Test Case 4

	Makespan (in seconds)		Reliability Achieved (in %)	
	RR Algorithm	MCTR Algorithm	RR Algorithm	MCTR Algorithm
Workflow 1	322	310	98.073	98.042
Workflow 2	344	336	98.052	98.038
Workflow 3	338	329	98.066	98.041
Workflow 4	364	358	98.014	98.008

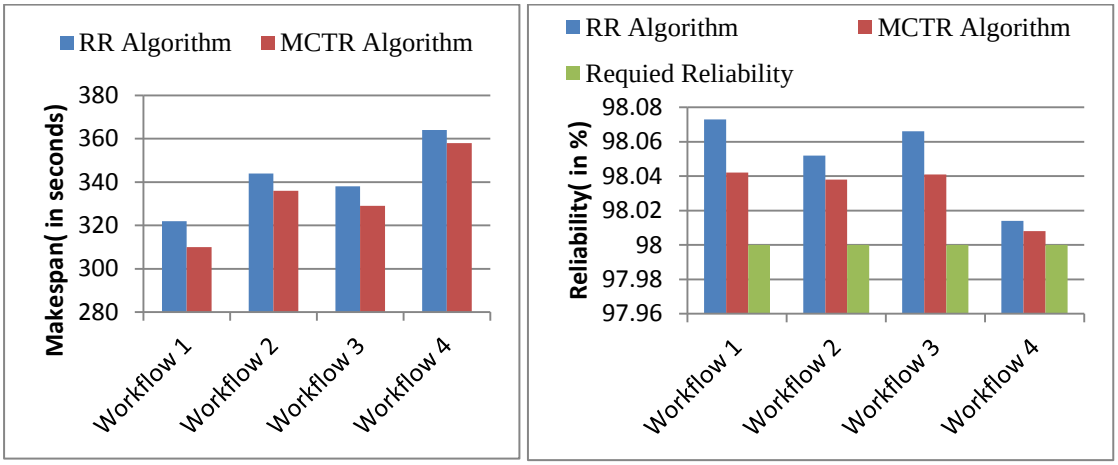


Figure 5.16: Graphs of Simulation Results for Test Case 4

The simulation results shown in above graph in Figure 5.16 again proves that the proposed MCTR algorithm take less time for workflow execution as compare to the RR algorithm while providing the reliability greater than the user required reliability.

5.4 Results Analysis

From the above experimental results, it can be concluded that the proposed MCTR algorithm reduces the makespan of the workflow while providing reliable execution as compare to the RR algorithm. When both algorithms are compared on the basis of reliability, the MCTR algorithm provide reliability level greater than user required reliability level, but less than the RR algorithm. The above experimental results clearly demonstrate that the proposed MCTR algorithm performs better than RR algorithm.

Chapter 6

Conclusion and Future Work

This chapter gives the conclusion of this thesis work and discuss about future direction which can be taken further.

6.1 Conclusion

This thesis gives introduction of various existing workflow scheduling algorithms for makespan optimization and reliable execution of workflows. In this thesis work, we have proposed MCTR scheduling algorithm, a hybrid approach for time efficient and reliable workflow execution which takes time and reliability as major concerns. Our scheduling algorithm tries to minimize the overall execution time of the workflow while meeting the required reliability as per user requirement. Simulations results have demonstrated that our algorithm achieves lower execution time as compare to RR algorithm while meeting user-designated reliability constraint.

6.2 Thesis Contribution

i) In this thesis, existing workflow scheduling algorithms for makespan optimization and reliable workflow execution have been analyzed and compared according to their parameters.
ii) MCTR algorithm, a hybrid approach for time efficient and reliable workflow scheduling has been proposed.
iii) The proposed algorithm has been implemented on CloudSim toolkit by using Netbeans IDE and JAVA runtime environment.
iv) Experimental results have been gathered and compared with previously existing RR algorithm.

6.3 Future Scope

In future, hybrid approach using multiple fault tolerance approaches can be used to further improve the efficiency of algorithm. Other workflow scheduling techniques like

PSO, ACO can also be used provide reliable workflow execution while satisfying certain QoS parameters. Also, performance of proposed MCTR algorithm can be further improved for cost and power efficiency.

References

[1] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica and M. Zaharia, "A view of cloud computing", Communications of the ACM, vol.53, no.4, pp. 50-58, 2010.

[2] M. Creeger, "Cloud Computing: An Overview," ACM Queue, vol.7, no.5, pp. 2.

[3] R. Buyya, C.S. Yeo, S. Venugopal, J. Broberg and I. Brandic, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility," Future Generation computer systems, vol. 25, no. 6, pp. 599-616, 2009.

[4] D. C. Plummer, T. J. Bittman, T. Austin, D. W. Cearley and D.M. Smith, "Cloud computing: Defining and describing an emerging phenomenon," Gartner, June, 17, 2008.

[5] P. Mell and T. Grance, "The NIST definition of cloud computing," National Institute of Standards and Technology, vol. 53, no. 6, pp. 50, 2009.

[6] D. C. Wyld, "Moving to the cloud: An introduction to cloud computing in government," IBM Center for the Business of Government, 2009.

[7] L. Kleinrock, "A vision for the Internet," ST Journal of Research, vol. 2, no. 1, pp. 4-5, 2005.

[8] I. Foster and C. Kesselman, "The Grid 2: Blueprint for a future computing infrastructure," Waltham: Morgan Kaufmann Publishers, 2004.

[9] M. A. Rappa, "The utility business model and the future of computing services," IBM Systems Journal, vol. 43, no. 1, pp. 32-42, 2004.

[10] M. Turner, D. Budgen and P. Brereton, "Turning software into a service," Computer. vol.36, no.10, pp. 38-44, 2003.

[11] "Cloud Watch Hub [Online]," Available at: http://www.cloudwatchhub.eu/glossary, Accessed on 4th October, 2013.

[12] S. Garg and R. Buyya, "Green Cloud Computing and Environmental Sustainability, Harnessing Green IT: Principles and Practices," S. Murugesan and G. Gangadharan, Wiley Press, UK, in press, accepted on April 2, 2011.

[13] Q. Zhang, L. Cheng and R. Boutaba, "Cloud computing: state-of-the-art and research challenges," Journal of internet services and applications, vol. 1, no. 1, pp. 7-18, 2010.

[14] M. A. Vouk, "Cloud computing–issues, research and implementations," CIT. Journal of Computing and Information Technology, vol. 16, no. 4, pp. 235-246, 2008.

[15] V. K. Reddy, B. T. Rao and L. S. S. Reddy, "Research Issues in Cloud Computing," Global Journal of Computer Science and Technology, vol. 11, no. 11, 2011.

[16] T. Dillon, C. Wu and E. Chang, "Cloud computing: issues and challenges. In Advanced Information Networking and Applications (AINA)," 2010 24th IEEE International Conference, pp. 27-33, April, 2010.

[17] B. Schroeder, and G. A. Gibson, "A large-scale study of failures in high performance computing systems," Dependable and Secure Computing, IEEE Transactions on vol. 7, no.4, pp. 337-351, October, 2010.

[18] L. Singh and S. Singh, "A Survey of Workflow Scheduling Algorithms and Research Issues," International Journal of Computer Applications (0975 – 8887), vol. 74, no.15, July, 2013.

[19] M. Rahman, R. Hassan, R. Ranjan and R. Buyya, "Adaptive workflow scheduling for dynamic grid and cloud computing environment," Concurrency and Computation: Practice and Experience, vol. 25, no.13, pp. 1816-1842, 2013.

[20] A. Bala and I. Chana, "A survey of various workflow scheduling algorithms in cloud environment," In 2nd National Conference on Information and Communication Technology (NCICT), 2011.

[21] Vijindra and S. Shenai, "Survey on scheduling issues in cloud computing," Procedia Engineering, vol. 38, pp. 2881-2888, 2012.

[22] S. Pandey, D. Karunamoorthy and R. Buyya, "Workflow engine for clouds". Cloud Computing: Principles and Paradigms, pp. 321-344, 2011.

[23] M. Dorigo and M. Birattari, "Ant colony optimization. In Encyclopedia of Machine Learning," Springer US, pp. 36-39, 2010.

[24] J. Kennedy and R. Eberhart, "Particle swarm optimization," In Proceedings of IEEE international conference on neural networks, vol. 4, no. 2, pp. 1942-1948, 1995.

[25] A. Ganesh, M. Sandhya and S. Shankar, "A study on fault tolerance methods in Cloud Computing," IEEE International Advance Computing Conference (IACC), 2014, pp. 844-849, 2014.

[26] 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.

[27] C. Lin and S. Lu, "Scheduling scientific workflows elastically for cloud computing," IEEE International Conference in Cloud Computing (CLOUD), pp. 746-747, February, 2011.

[28] A. G. Delavar and Y. Aryan, "HSGA: a hybrid heuristic algorithm for workflow scheduling in cloud systems," Cluster Computing, pp. 1-9, 2013.

[29] K. Liu, H. Jin, J. Chen, X. Liu, D. Yuan and Y. Yang, "A compromised-time-cost scheduling algorithm in swindew-c for instance-intensive cost-constrained workflows on a cloud computing platform," International Journal of High Performance Computing Applications, vol. 24, no. 4, pp. 445-456, 2010.

[30] Z. Wu, X. Liu, Z. Ni, D. Yuan and Y. Yang, "A market-oriented hierarchical scheduling strategy in cloud workflow systems," The Journal of Supercomputing, vol. 63, no. 1, pp. 256-293, 2013.

[31] H. Arabnejad and J. G. Barbosa, A Budget Constrained Scheduling Algorithm for Workflow Applications," Journal of Grid Computing, pp. 1-15, 2014.

[32] S. Pandey, L. Wu, S. M. Guru and R. Buyya, "A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments," 24th IEEE International Conference on Advanced Information Networking and Applications(AINA), pp. 400-407, April, 2010.

[33] W. N. Chen and J. Zhang, "An ant colony optimization approach to a grid workflow scheduling problem with various QoS requirements," IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, vol. 39, no. 1, pp. 29-43, 2009.

[34] S. Abrishami, M. Naghibzadeh and D. H. Epema, "Deadline-constrained workflow scheduling algorithms for Infrastructure as a Service Clouds," Future Generation Computer Systems, vol. 29, no. 1, pp. 158-169, 2013.

[35] A. Benoit, M. Hakem and Y. Robert, "Fault tolerant scheduling of precedence task graphs on heterogeneous platforms," IEEE International Symposium on Parallel and Distributed Processing (IPDPS), pp. 1-8, April,2008.

[36] A. Benoit, M. Hakem and Y. Robert, "Contention awareness and fault-tolerant scheduling for precedence constrained tasks in heterogeneous systems," Parallel Computing, vol. 35, no. 2, pp. 83-108, 2009

[37] Q. Zheng, B. Veeravalli and C. K. Tham, "On the design of fault-tolerant scheduling strategies using primary-backup approach for computational grids with low replication costs," IEEE Transactions on Computers, vol. 58, no. 3, pp. 380-393, 2009.

[38] Y. Zhang, A. Mandal, C. Koelbel and K. Cooper, "Combined fault tolerance and scheduling techniques for workflow applications on computational grids," 9th IEEE/ACM International Symposium on Computing and the Grid(CCGRID'09), pp. 244-251, May, 2009

[39] B. Mills, T. Znati and R. Melhem, "Shadow computing: An energy-aware fault tolerant computing model," In Proceedings of the International Conference on Computing, Networking and Communications (ICNC), 2014.

[40] Y. C. Lee and A. Y. Zomaya, "Rescheduling for reliable job completion with the support of clouds," Future Generation Computer Systems, vol. 26, no. 8, pp. 1192-1199, 2010.

[41] Y. C. Lee, A. Y. Zomaya and M. Yousif, "Reliable workflow execution in distributed systems for cost efficiency," 11th IEEE/ACM International Conference on Grid Computing (GRID), pp. 89-96, October, 2010.

[42] X. Wang, C. S. Yeo, R. Buyya and J. Su, "Optimizing the makespan and reliability for workflow applications with reputation and a look-ahead genetic algorithm," Future Generation Computer Systems, vol. 27,no. 8, pp. 1124-1134, 2011.

[43] D. de Oliveira, K. A. Ocana, F. Baiao and M. Mattoso, "A provenance-based adaptive scheduling heuristic for parallel scientific workflows in clouds, " Journal of Grid Computing, vol. 10, no. 3, pp. 521-552, 2012.

[44] A. M. Sampaio and J. G. Barbosa, "Dynamic Power-and Failure-Aware Cloud Resources Allocation for Sets of Independent Tasks," IEEE International Conference on Cloud Engineering (IC2E), pp. 1-10. March, 2013.

[45] Y. Gao, S. K. Gupta, Y. Wang and M. Pedram. "An energy-aware fault tolerant scheduling framework for soft error resilient cloud computing systems," In Design, Automation and Test in Europe Conference and Exhibition (DATE), pp. 1-6, March, 2014.

[46] L. Zhao, Y. Ren and K. Sakurai, "Reliable workflow scheduling with less resource redundancy. Parallel Computing," vol. 39, no. 10, pp. 567-585, 2013.

[47] I. P. Egwutuoha, S. Chen, D. Levy, B. Selic and R. Calvo, "Cost-oriented proactive fault tolerance approach to high performance computing (HPC) in the cloud," International Journal of Parallel, Emergent and Distributed Systems, pp. 1-16, 2014.

[48] L. Zhao, Y. Ren, Y. Xiang and K. Sakurai, "Fault-tolerant scheduling with dynamic number of replicas in heterogeneous systems," 12th IEEE International Conference on High Performance Computing and Communications (HPCC), pp. 434-441, 2010.

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

[50] "NetBeans IDE [Online]," Available at: https://netbeans.org/features/index.html, Accessed on 11th March 2014.

[51] "Oracle Database [Online]," Available at: http://www.oracle.com/pls/db102/ home-page, Accessed on 11th March, 2014.

List of Papers

Communicated

[1] J. Jain and S. Bawa, "Workflow Scheduling Algorithms in Cloud Computing: A Review," In 5th International Conference Confluence 2014 at Amity University, 25-26th September, 2014.

[2] J. Jain and S. Bawa, "Time Efficient and Reliable Workflow Scheduling in Cloud Computing," In 5th International Conference Confluence 2014 at Amity University, 25-26th September, 2014.