

Priority Scheduling Algorithm with Fault Tolerance in Cloud Environment

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

**Master of Engineering
in
Computer Science and Engineering**

Submitted By
Nimisha Singla
(Roll No. 801132017)

Under the supervision of:
Dr. Seema Bawa
Professor and Dean of Student Affairs



COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
THAPAR UNIVERSITY
PATIALA – 147004

July 2013

CERTIFICATE

I hereby certify that the work which is being presented in the thesis entitled, "**Priority Scheduling Algorithm with Fault Tolerance in Cloud Environment**" 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. Seema Bawa** 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.


15/7/13
(Nimisha Singla)

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


15/07/2013
(Dr. Seema Bawa)

Professor
Department of Computer
Science and Engineering

Countersigned by


(Dr. Maninder Singh)
Head
Computer Science and Engineering Department
Thapar University
Patiala


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

Acknowledgment

First of all I would like to thank the Almighty, who has always guided me to work on the right path of the life.

This work would not have been possible without the encouragement and able guidance of my supervisor **Dr. Seema Bawa**. I thank my supervisor for their time, patience, discussions and valuable comments. Their enthusiasm and optimism made this experience both rewarding and enjoyable.

I am equally grateful to **Dr. Maninder Singh**, Associate Professor and Head, Computer Science & Engineering Department, for motivation and inspiration that triggered me for the thesis work.

I will be failing in my duty if I don't express my gratitude to **Dr. S. K. Mohapatra**, Senior Professor and Dean of Academic Affairs of the University, for making provisions of infrastructure such as library facilities, computer labs equipped with net facilities, immensely useful for the learners to equip themselves with the latest in the field.

I am also thankful to the entire faculty and staff members of Computer Science and Engineering Department for their direct-indirect help, cooperation, love and affection, which made my stay at Thapar University memorable.

Last but not least, I would like to thank my parents for their wonderful love and encouragement, without their blessings none of this would have been possible. I would also like to thank my brother as he deserves this much more than anybody else. I would also like to thank my friends for their constant support.

Nimisha Singla
(801132017)

Abstract

Cloud computing is an upcoming model for business computing, which makes computing tasks maintain resource pool which constitutes of a large number of computer resources, so that all application programs can get computing power, storage space and all kinds of software services. The computability of the various resource scheduling algorithms is different in different environments. Algorithm according to topology, aspect and many others ways are present. Energy efficient, dynamic priority based, virtual resources model and many other algorithms are present for scheduling tasks, resources and workflows.

In this work the Fault tolerance aspect of the server is taken into consideration. An attempt to embed fault tolerance functionality in priority scheduling algorithm is made. This algorithm focuses on the allocating the functionality of the faulty server to another server. Further this algorithm concentrates on the type of cloud architecture too. The performance is measured by providing different inputs. The results provided focuses on better execution of the tasks provided to the servers and depicted through graph.

Table of Contents

Certificate	i
Acknowledgement	ii
Abstract	iii
Table of Contents	iv
Table of Figures	vii
List of Tables	viii
Abbreviations	ix
Chapter 1: Introduction	1
1.1 Cloud Computing	1
1.1.1. Characteristics of Cloud Computing	3
1.1.2. Key Technologies of Cloud Computing	3
1.2 Scheduling Types in Cloud Computing	5
1.2.1. Cloud Service Scheduling	5
1.2.2. User Level Scheduling	5
1.2.3. Static and Dynamic Scheduling	6
1.2.4. Heuristic Scheduling	8
1.2.5. Real Time Scheduling	10
1.2.6. Workflow Scheduling	10
1.2.7. Resource Scheduling	10
1.3 Need for Resource Scheduling in Cloud Computing	11
1.4 Thesis Organization	13
Chapter 2: Literature Review	14
2.1. Algorithm based on Energy Efficient Optimization Methods	14
2.2. Dynamic priority scheduling algorithm	15
2.3. Non-Dominated Sorting Genetic Algorithm II	15
2.4. Optimizing Virtual Machine for High Performance Computing	16

2.5. Scheduling with Parallel Genetic Algorithm	17
2.6. Balance Reduce Algorithm (Fault Tolerant)	18
2.7. Heavy Traffic Optimal Algorithm	18
2.8. Optimized Resource Scheduling	19
2.9. Cost Based RS paradigm	19
2.10. Double Auction-based Scheduling of scientific applications	19
2.11. Algorithm based on Trust Degree	20
2.12. PSO based hierarchical strategy	20
2.13. Dynamic resource scheduling and workflow management	20
2.14. SLA Restriction in Pareto Optimality M×N production	20
2.15. Loyalty based resource allocation	21
2.16. Dynamic and integrated load balancing algorithm	21
2.17. Adaptive optimal global resource scheduling	21
2.18. Load balance based algorithm	21
2.19. Green power management for virtual machines	22
2.20. Component based resource allocation	22
2.21. Pareto based optimal scheduling	22
2.22. Smart Dynamic resource scheduling algorithm	22
2.23. Genetic algorithm with MultipleFitness	23
2.24. Hybrid multidimensional algorithm for network aware scheduling	23
2.25. Pricing algorithm	23
2.26. Load adaptive model based on ant colony algorithm	23
Chapter 3: Proposed Priority Scheduling Algorithm	26
3.1. Research Gaps	26
3.2. Problem Statement	26
3.3. Objectives	26
3.4. Methodology	27
3.5. Proposed Algorithm of Priority Scheduling	27

Chapter 4: Design and Implementation Details	31
4.1 CloudSim	31
4.2 Flow Chart of Proposed Algorithm	33
4.3 Implementation Details	34
Chapter 5: Testing and Results	37
5.1 Types of Testing Performed	37
5.1.1. Unit Testing	37
5.1.2. Integration Testing	37
5.1.3. System Testing	37
5.1.4. Alpha Testing	37
5.2 Test Cases	38
5.2.1. Test Case 1	38
5.2.2. Test Case 2	41
5.2.3. Test Case 3	44
5.3. Analysis of Results	47
Chapter 6: Conclusion and Future Work	49
6.1 Conclusion	49
6.2 Future Work	49
References	50
List of Publications	55

List of Figures

Fig. 1.1 Cloud Computing Architecture	2
Fig. 1.2 Cloud computing computation	4
Fig. 1.3 Resource Scheduling in Cloud Environment	11
Fig. 4.1 Layered Architecture of CloudSim	32
Fig. 4.2 Flowchart for priority scheduling of initial arriving tasks	33
Fig. 4.3 Flowchart for detecting fault in any of the server and reallocating the tasks to another server	34
Fig 4.4 Screenshots describing input and assigning values	35
Fig 4.5 Screenshots describing assignment of different constants	36
Fig 5.1 (a) Screenshots depicting results for one case in first test case scenario	39
Fig 5.1 (b) Screenshots depicting results for one case in first test case scenario	40
Fig 5.1 (c) Tradeoff between Safety Parameter and Mean for first test case scenario	41
Fig 5.1 (d) Tradeoff between Safety Parameter and Standard Deviation for first case scenario	41
Fig 5.2 (a) Screenshots depicting results for one case in second test case scenario	42
Fig 5.2 (b) Screenshots depicting results for one case in second test case scenario	43
Fig 5.2 (c) Tradeoff between Safety Parameter and Mean for second test case scenario	44
Fig 5.2 (d) Tradeoff between Safety Parameter and Standard Deviation for second case test scenario	44
Fig 5.3 (a) Screenshots depicting results for one case in third test case scenario	45
Fig 5.3 (b) Screenshots depicting results for one case in third test case scenario	46
Fig 5.3 (c) Tradeoff between Safety Parameter and Mean for third test case scenario	47
Fig 5.3 (d) Tradeoff between Safety Parameter and Standard Deviation for third test case scenario	47

List of Tables

Table 2.1 Comparison between some of the existing algorithms	24
Table 5.1 Experimental Results for first test case scenario	38
Table 5.2 Experimental Results for second test case scenario	42
Table 5.3 Experimental Results for third test case scenario	45

Abbreviations

ABC	Activity Based Costing
BAR	Balance Reduce Algorithm
BoT	Bag of Tasks
BS	Balanced Spiral
COA	Course of Action
CPU	Central Processing Unit
DPSA	Dynamic Priority Scheduling Algorithm
DRR	Deadline, Reliability, Resource-Aware
FCFS	First Come First Serve
FIFO	First In First Out
GA	Genetic Algorithm
HDFS	Hadoop Distributed File System
HPC	High Performance Computing
IaaS	Infrastructure as a Service
JGAP	Java Genetic Algorithm Package
JSQ	Join-the-shortest Queue
MAGA	Multi-Agent Genetic Algorithm
MO-GA	Multi Objective Genetic Algorithm
MMT	Minimum Migration Time
MQMW	Multiple QoS Constrained Scheduling Strategy of Multi Workflow
NDSA	Non-Dominated Sorting Genetic Algorithm
NP	Non-Deterministic Polynomial Time
PGA	Parallel Genetic Algorithm
PSO	Particle Swarm Optimization
QoS	Quality of Service
RASA	Resource Aware Scheduling Algorithm
RL	Reinforcement Learning
RPC	Remote Procedure Call

SAQA	Self-Adaptive QoS Aware
SES	Single Exponential Smoothing
SHEFT	Scalable Heterogeneous Earliest-Finish-Time
SLA	Service Level Agreement
SPSA	Static Priority Scheduling Algorithm
TUF	Time Utility Function
VMs	Virtual Machines
VP	Vector Projection

Chapter 1

Introduction

This chapter includes the introduction of the cloud computing environment, its characteristics and types of the scheduling practiced in cloud, resource scheduling introduction and need for the resource scheduling in the cloud environment.

1.1. Cloud Computing

Cloud computing is an emerging model for business computing, which makes computing tasks lay resource pool which constitutes of a large number of computer resources, so that all application programs can get computing power, storage space and all kinds of software services. The basic principles of cloud computing are that data entered by the user is not stored locally, but is stored in data center of internet. The companies that can provide cloud computing services manage and maintain the normal operation of these data centers, ensure strong computing power and large storage space for users, then users only at any time, and any place use any terminal equipment that is connected to the internet to access these services without having to think of the position of cloud that are stored.

Computing based on the internet sharing resources is called cloud computing. Cloud computing is the fastest growing technology in today's era offering ubiquitous services to users. The need of services to the lowest level is in demand. Nowadays everybody is not ready to purchase the devices that provide the services. The users rather purchase the services provided by the devices at the big servers. The infrastructure of pay-per-use is highly in demand. The users from different locations just like to have the services and pay for the time being they are availing the services. Cloud computing enables convenient and on-demand network access to shared pool of computing resources that needs to be managed. It delivers applications which are accessible from web browsers, desktop and mobile apps. Optimization of energy efficiency in cloud computing is necessary. It is a large scale computing using virtual resources. Its popularity is increasing as a cost effective alternative and also High

Performance Computing for supercomputers. There have been different clouds releases until now Eucalyptus, Hadoop, and Nimbus etc.

Cloud computing is integration of distributed processing, parallel processing and grid computing, which make data stored on the cloud, make software and services placed in the cloud and build on top of various standards and protocols, you can get it through all kinds of equipments. In the cloud computing model, the application and data storage of users do not do and are stored in personal computers, mobile phones and other devices, but can run and are stored in large scale internet server clusters. The cloud computing model can be used to integrate existing hardware and software resources, reducing software purchase cost, speeding up construction and application of network information resource with limited funds and time.

The basic principles of cloud computing are that data is done by the user is not stored locally, but is stored in data center of internet. The companies that can provide cloud computing services manage and maintain the normal operation of these data centers, ensure strong computing power and large storage space for users, then users only at any time, and any place use any terminal equipment that is connected to the internet to access these services without having to think of the position of cloud that are stored.

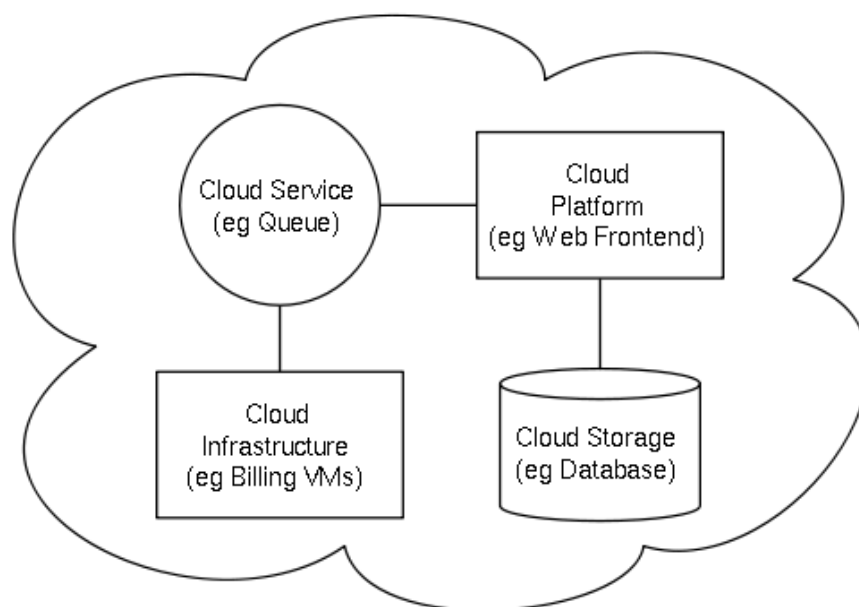


Fig. 1.1 Cloud Computing Architecture

1.1.1. Characteristics of Cloud Computing:

- Cloud computing provides the most reliable, and most secure data storage center. In the other side of cloud, the professional team is used to manage information, the most advanced data center is used to save data and users do not worry about data loss, viruses and other problems.
- While the users can conveniently use computer to access applications and services of "cloud" side through the browsers side.
- Cloud computing can easily implement data application and sharing of different equipments. In the cloud computing model of network applications, data is only preserved in other side of the "cloud", the terminal only is connected to the internet and one can access and use same data, and can share data with others.
- Cloud computing provides computing power. In the other side of cloud computing, a large cluster that consists of tens of thousands or more servers provides infinite number of space for the storing and management of data, but also provides infinitely powerful computing capacity for the completion of various applications.

1.1.2. Key Technologies of Cloud Computing

Cloud is data intensive supercomputing which regards data as center, show priority in data storage, data management, programming model, and concurrency control and system management etc.

- **Massive distributed memory computing:** To ensure high availability, high reliability and economy, cloud computing uses distributed storage mode to store data, redundant data storage is used to ensure the reliability of data, high reliability software is used to make up unreliable hardware and provide cheap and reliable system.
- **Parallel programming model:** For efficient use of cloud computing resources and make users easily enjoy service that the cloud computing brought, cloud computing programming model must ensure that the background of complex parallel and task

scheduling is transparent to users and programmers, cloud computing uses Map Reduce programming model to divide the task automatically into multiple sub-tasks, achieve the scheduling and allocation that task is in large-scale computing nodes.

- **Data management technology:** Cloud computing system process, analysis large data sets and provide efficient services for users. Therefore, data management technology must be able to efficiently manage large data sets and to find specific data from the large data is the problem.
- **Distributed resource management technology:** In multimode Concurrent execution environment, distributed resource management system is to ensure the correctness of key technology system state. The state requires synchronization between multiple nodes, failure of key node needs relocated services, distributed resource management technology coordinate multi-task for the use of resources through the lock mechanism, then ensuring consistency of data operations.

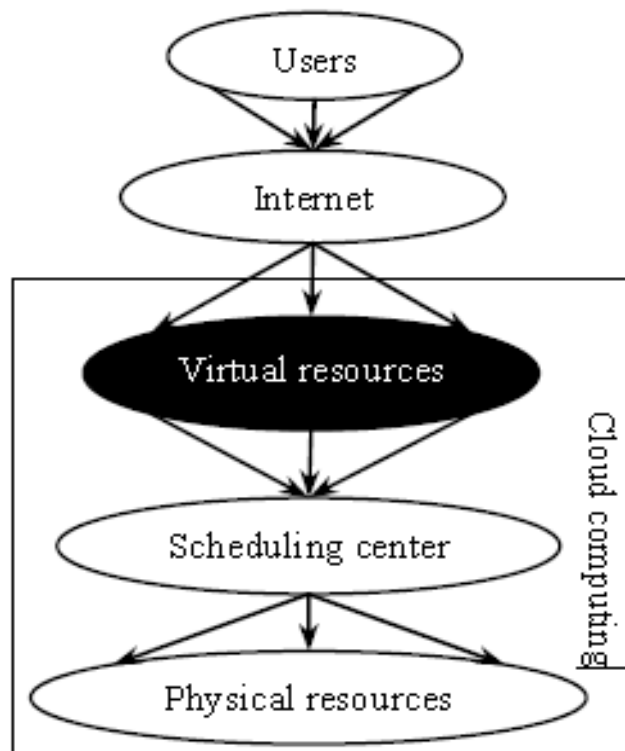


Fig. 1.2 Cloud computing computation [7]

1.2. Scheduling Types in Cloud Computing

There are various scheduling practiced in the cloud computing and their description. These scheduling are present at different levels of computation and on the different levels.

1.2.1. Cloud Service Scheduling

Cloud service scheduling is categorized at user level and system level. At user level scheduling deals with problems raised by service provision between providers and customers. The system level scheduling handles resource management within datacenter. Datacenter consists of many physical machines. Millions of tasks from users are received; assignment of these tasks to physical machine is done at datacenter. This assignment or scheduling significantly impacts the performance of datacenter. In addition to system utilization, other requirements like QoS, SLA, resource sharing, fault tolerance, reliability, real time satisfaction etc should be taken into consideration.

1.2.2. User Level Scheduling

Market-based and auction-based schedulers are suitable for regulating the supply and demand of cloud resources. Market based resource allocation is effective in cloud computing environment where resources are virtualized and delivered to user as a service. A suite of market oriented task scheduling algorithms to an AuctionNet for heterogeneous distributed environments has been proposed. Development of a pricing model using processor-sharing for clouds, the application of this pricing model to composite services with dependency consideration and the development of two sets of profit-driven scheduling algorithms have been proposed. Service provisioning in Clouds is based on Service Level Agreements (SLA). SLA represents a contract signed between the customer and the service provider stating the terms of the agreement including non-functional requirements of the service specified as Quality of Service (QoS), obligations, and penalties in case of agreement violations. Thus there is a need of scheduling strategies considering multiple SLA parameters and efficient allocation of resources. A novel scheduling heuristic considering multiple SLA parameters for deploying applications in cloud has been presented.

A novel cloud scheduling scheme uses SLA along with trust monitor to provide a faster scheduling of the over flooding user request with secure processing of the request. A novel approach of heuristic-based request scheduling at each server, in each of the geographically distributed data centers, to globally minimize the penalty charged to the cloud computing system has been proposed. This approach considers two variants of heuristics, one based on the simulated annealing method of neighborhood searches and another based on gi-FIFO scheduling. Based on the queuing model and system cost function, considering the goals of both the cloud computing service users and providers, proposes an algorithm to get the approximate optimistic value of service for each job in the corresponding no-preemptive priority M/G/1 queuing model. This approach guarantees the QoS requirements of the users as well as the maximum profits for the cloud computing service providers. To deal with dynamically fluctuating resource demands, market-driven resource allocation has been proposed and implemented by public Infrastructure-as-a-Service (IaaS) providers like Amazon EC2.

In this environment, cloud resources are offered in distinct types of virtual machines (VMs) and the cloud provider runs an auction-based market for each VM type with the goal of achieving maximum revenue over time. A case study of single cloud provider and how to best match customer demand in terms of both supply and price in order to maximize the providers revenue and customer satisfactions while minimizing energy cost has been proposed.

1.2.3. Static and Dynamic Scheduling

Static scheduling allows for pre-fetching required data and pipelining different stages of task execution. Static scheduling imposes less runtime overhead. In case of dynamic scheduling information of the job components/task is not known before hand. Thus execution time of the task may not be known and the allocation of tasks is done on fly as the application executes. A job execution environment Flextic that exploits scalable static scheduling techniques to provide the user with a flexible pricing model and at the same time, reduce scheduling overhead for the cloud provider has been presented in. The service request scheduling strategies in three-tier cloud structure, which consists of resource providers, service providers and

consumers, should satisfy the objectives of the service providers and consumers. A dynamic priority scheduling algorithm (DPSA) has been proposed to achieve above objectives.

A new fault tolerant scheduling algorithm MaxRe has been proposed. This algorithm incorporates the reliability analysis into the active replication schema, and exploits a dynamic number of replicas for different tasks. A trust mechanism-based task scheduling model is presented. Trust relationship is built among computing nodes, and the trustworthiness of nodes is evaluated by utilizing the Bayesian cognitive method. A feedback dynamic algorithm for preemptable job scheduling mechanism has been proposed. A preemptable scheduling improves the utilization of resources in clouds and feedback procedure in above algorithms works well in the situation where resource contentions are fierce.

In cloud computing, traditional way for task scheduling cannot measure the cost of cloud resources accurately by reason that each of the tasks on cloud systems is totally different between each other. An optimized algorithm for task scheduling based on ABC (activity based costing) in cloud computing and its implementation has been introduced. Also an experiment on different optimization strategies for cost-optimal dynamic scheduling in hybrid cloud environments has been performed. To achieve QoS in cloud environment; propose an improved backfill algorithm using balanced spiral (BS) method.

A scheduling algorithm has been proposed that measures both resource cost and computation performance and also improves the computation/communication ratio by grouping the user tasks according to a particular cloud resource's processing capability and sends the grouped jobs to the resource. Due to job grouping, communication of coarse-grained jobs and resources optimizes computation/communication ratio. A large number of cloud computing servers waste a tremendous amount of energy and emit a considerable amount of carbon dioxide. Green task scheduling is necessary to significantly reduce pollution and substantially lower energy usage. As study considering the realistic network topology and communication model, proposes the Deadline, Reliability, Resources-aware (DRR)

scheduling algorithm. Considering the failure and recovery scenario in the Cloud computing entities, proposes a Reinforcement Learning (RL) based algorithm to make job scheduling fault-tolerable while maximizing utilities attained in the long term. A new framework of task scheduling strategy for tree network has been proposed.

1.2.4. Heuristic Scheduling

Optimization problems in Class NP-hard are heuristic. These problems can be solved by enumeration method, heuristic method or approximation method. In enumeration method, an optimal solution can be selected if all the possible solutions are enumerated and compared one by one. When number of instances is large, exhaustive enumeration is not feasible for scheduling problems. In that case heuristic is a suboptimal algorithm to find reasonably good solutions reasonably fast. Approximation algorithms are used to find approximate solutions to optimized solution. These algorithms are used for problems when exact polynomial time algorithms are known. Enhancing task data locality in large scale data processing systems is crucial for the job completion time. Most of the approaches to improve data locality are either greedy and ignore global optimization, or suffer from high computation complexity. This problem is addressed by proposing a heuristic task scheduling algorithm called Balance-Reduce (BAR).

Load balancing task scheduler balance the entire system load while trying to minimizing the make span of a given tasks set. Two different load balancing scheduling algorithms based on ant colony have been proposed. Another ant colony based algorithm aims to minimize job completion time based on pheromone has been proposed. Cloud Loading Balance algorithm, adds capacity to the dynamic balance mechanism for the cloud environment. The decision, which workloads to outsource to what cloud provider, should maximize the utilization of the internal infrastructure and minimize the cost of running the outsourced tasks in the cloud, while taking into account the applications quality of service constraints. A set of heuristics, to cost-efficiently schedule deadline constrained computational applications, has been proposed. Multi-objective meta-heuristics scheduling algorithm for multi-cloud environment has been proposed. This algorithm tries to achieve application high availability and fault-tolerance while reducing the application cost and keeping the

resource load maximized. Because of the increasing large Web graph and social networks, cost-conscious large graph processing scheduling is important and a heuristic for the same has been proposed.

An optimized algorithm based on GA to schedule independent and divisible tasks adapting to different computation and memory requirements has been proposed. Multi-agent genetic algorithm (MAGA) is a hybrid algorithm of GA which solves the load balancing problem in cloud computing. COA (Course of action) planning involves resource allocation and task scheduling. A robust COA planning with varying durations based on GA has been proposed. Reducing energy consumption is an increasingly important issue in cloud computing, more specifically when dealing with High Performance Computing (HPC). A multi-objective genetic algorithm (MO-GA), proposed, optimizes the energy consumption, carbon dioxide emissions and the generated profit of a geographically distributed cloud computing infrastructure.

Simulated annealing is a generic probabilistic meta heuristic for the global optimization problem of locating a good approximation to the global optimum of a given function in a large search space. An optimized algorithm for task scheduling based on genetic simulated annealing algorithm in cloud computing has been proposed. The scalability of a computing system can be mainly identified by size, geographical distribution, administrative constraints, heterogeneity, energy consumption and transparency. A low complexity energy efficient heuristic algorithm for scheduling, proposed, performs efficiently demonstrating their applicability and scalability. In batch mode, tasks are scheduled only at some predefined time. This enables batch heuristics to know about the actual execution times of a larger number of tasks. Min-min and Max-min are heuristics used for batch mode scheduling. Heuristics based improved Maxmin algorithm has been proposed and the QoS Min-Min scheduling algorithm has been proposed. Bag of tasks (BoT) applications are the one which execute independent parallel tasks. Heuristics proposed aims to maximize resource utilization while executing BoTs in heterogeneous sets of Cloud resources allocated for different numbers of hours. Another budget constraint scheduler proposed schedules large bags of tasks onto multiple clouds with different CPU performance and cost, minimizing completion time while respecting an upper bound

for the budget to be spent. When providers cannot disclose private information such as their load and computing power, which are usually heterogeneous, the meta-scheduler needs to make blind scheduling decisions. In this case a deadline-constrained BoT application scheduling approach has been proposed.

1.2.5. Real Time Scheduling

The primary objectives of real time scheduling are to increase throughput and minimize average response time instead of meeting deadlines. The real-time tasks are scheduled non-preemptively with the objective to maximize the total utility. Two different time utility functions (TUFs)-a profit TUF and a penalty TUF- are associated with each task at the same time. This approach rewards for the early completions and also penalizes the abortions or deadline misses of real-time tasks. Similarly a preemptive algorithm and a real time workload driven approach have been proposed. Quality of service (QoS) guarantees for some applications such as signal data processing is very important. A novel self-adaptive QoS-aware scheduling algorithm called SAQA considers the adaptability for real-time tasks with QoS demands on heterogeneous clusters.

1.2.6. Workflow Scheduling

A workflow enables the structuring of applications in a directed acyclic graph form, where each node represents the constituent task and edges represent inter task dependencies of the applications. A single workflow generally consists of a set of tasks each of which may communicate with another task in the workflow. Workflow scheduling is one of the key issues in the management of workflow execution.

1.2.7. Resource Scheduling

Resource scheduling is the basic and key process for clouds in Infrastructure as a Service (IaaS) as the need of the request processing is must in the cloud. Every server has limited resources so jobs/requests needs to be scheduled. Each application in the cloud computing is designed as a business processes including a set of abstract processes. To allocate the resources to the tasks there need to schedule of the resources as well as tasks coming to the resources. There need to be a Service Level

Agreements (SLAs) for Quality of Service (QoS). Till now no algorithm is been introduced which considers reliability and availability. According to the paradigm of cloud there has been a lot of task scheduling algorithms, some are being fetched on the basics of scheduling done on the operating system. The basics of operating system job scheduling is taken and applied to the resources being installed in the cloud environment.

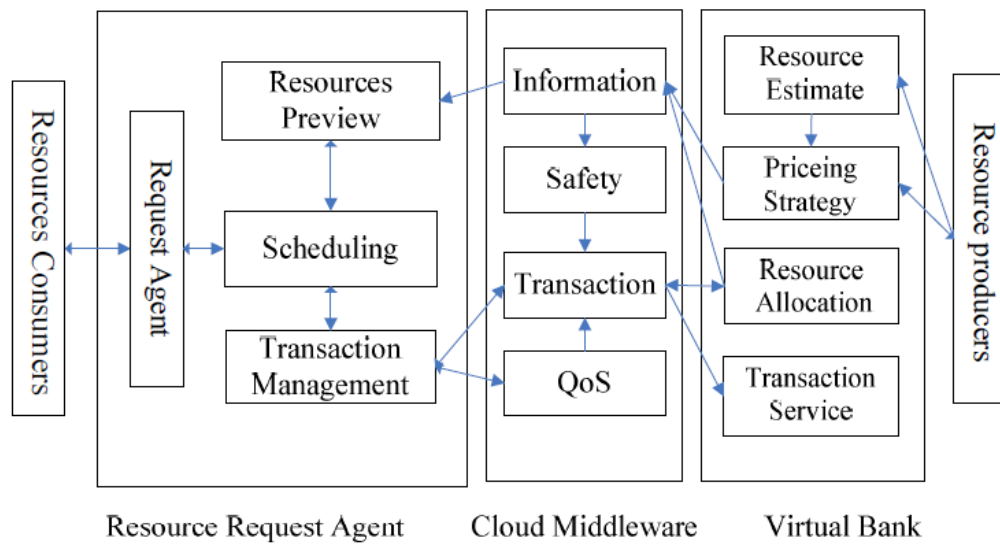


Fig. 1.3 Resource Scheduling in Cloud Environment [25]

Cloud computing has a base of distributed, grid and virtualization. Till now unbalanced strategies are being introduced. The cost for transferring data and information should also be included. It should be secure, optimal and convenient. The main objective is to satisfy providers and consumers in optimized strategies as to gain resource efficiency and maximum profit.

1.3. Need of Resource Scheduling in Cloud Computing

There is enormous need for the cloud services to schedule the resources as this scheduling will further followed by the job/task scheduling inside of the resources. There may be many instances of the single resource that they can be run at the same time. There is need of checking of availability and reliability and also the load must be balanced among the resources of the same type. For the above parameters there

need to be a procedure or function that could check them and allocation should be done in the best and optimal way.

There are network strategies that could provide services like compute, storage and bandwidth management at less cost. The best way is combine together the computability of network strategies with scheduling algorithms. Usually when tasks are scheduled they are done according to user's requirements and requests but while looking into all the aspects the computation needs to be done. Application scalability is the main aim for the cloud services to achieve. In cloud scalability of resources allows real time provisioning of the resources for services. Cloud has complex execution environment but it has to provide the QoS to its users. Virtual resources are used best for the fully customizable configuration environment for application.

There have been a lot of work done on resource scheduling in cloud computing. New algorithms and management techniques for resource scheduling in cloud computing are being advised to make cloud computing a best experience for providers as well as customers. The surveys on scheduling strategies, techniques, methods have been done and a lot of job/task scheduling algorithms are introduced. The resource scheduling is been a tough job in cloud specially as it is the one which decides which process will be allocated to which resource and for how much time.

There are also resource allocation strategies that take into consideration the input parameters and on the basis if whether they are related to either of customer and providers. These parameters are execution time, policy, virtual machine, utility function, gossip, application, auction, hardware resource dependency, SLAs. While making a strategy the allocation methods should keep into consideration resource contention, fragmentation, over provisioning and under provisioning [4].

The various task scheduling methods in cloud computing are Cloud Service, User Level, Static and Dynamic, Heuristic, Workflow and Real Time scheduling [1]. Some of the scheduling algorithms in cloud whether or task or job or workflow or resources are Compromised-Time-Cost, Particle Swarm Optimization based Heuristic, Improved cost based for tasks, RASA workflow, Innovative transaction intensive cost constraint, SHEFT workflow, Multiple QoS Constrained for Multi- Workflows [2]. There are also the workflow scheduling algorithms that are described some of which

are ant colony, deadline constrained, market oriented hierarchical etc [3]. These surveys concluded that there is still a need for reliable and available resource scheduling algorithms as none of them concentrates on both parameters.

1.4. Thesis Organization

Chapter 1: This chapter presents the introduction to cloud computing, various scheduling in cloud environment and need of resource scheduling.

Chapter 2: This chapter presents the various algorithms available till now for the cloud computing resources scheduling and comparison between them.

Chapter 3: This chapter presents the problem statement i.e. proposed method, its objectives, research gaps and methodology to achieve objectives.

Chapter 4: This chapter presents the design and implementation details of the platform to achieve the objective and the detailed explanation of proposed method.

Chapter 5: This chapter presents the various test cases and graph depicting the performance.

Chapter 6: This chapter finally contains the conclusion of the work done and future work possible for the proposed method.

Chapter 2

Literature Review

This chapter includes the brief description of algorithms present till now for scheduling resources in the cloud computing environment. Further there is comparison between some of the algorithms in the cloud.

2.1. Algorithm based on Energy Efficient Optimization Methods: This algorithm is being implemented in Hadoop distributed file system with Energy Management and Regulation also called as GreenHDFS. This algorithm concentrates on usage of the resources that are not fully utilized while execution of the environment. Due to fast advancement in technology the old methods of saving energy has been challenging. The works introduced till now are taken into account with hardware but not with software.

While this algorithm checks the energy consumption of the various computing resources that are involved in cloud like node, storage, switch and network. The resources CPU, main memory and storage has been worked till now and future work includes temperature and fan speed. Node in cloud computing is similar to servers composed of more than one multi core CPU which provides parallel services. The energy consumption depends on the type of the job whether compute intensive or I/O or storage. The clustering is done in a way to save energy.

The user first chooses type of the job and then the job is in execution mode again the type is analyzed by counting the number of instruction execution speed. The basics of Round Robin algorithm are used. There are three phases in this algorithm: Infrastructure Preparation, Job Preprocessing and Job Execution. These estimates are approximate as the monitoring method used is indirect i.e. by sensors. This algorithm till now is implemented on Eucalyptus and data processing program is Hadoop. The readings of this algorithm are compared with the basic round robin algorithm in original environment [5].

2.2. Dynamic priority scheduling algorithm (Service request scheduling): This algorithm is applied on three tier containing service providers, resource providers and consumers. This algorithm gives more optimal then First Come First Serve (FCFS) and Static Priority Scheduling Algorithm (SPSA). The consumer response time for services has been tried to reduce in this algorithm as running instance is charged as it runs per unit time. The delays in provider side happens but are not counted under the cost charged to the customer so they need to be reduced. In three tiers there needs to be two scheduling: service request scheduling and resource scheduling.

The FCFS concentrates on fairness to task units but it may result in low priority task units perform before than high priority tasks and SPSA makes task units prioritized before the process of scheduling. The DPSA evaluates task unit scheduled and recalculates and set task unit's priority thus optimizing the scheduling process. Though tasks has their initial priorities but the new priorities being set include SLA between user and cloud, task's features, task's source and operations in cloud. This algorithm considers three queues having highest priority, middle priority and lowest priority. Every queue has a threshold i.e. time a task unit will wait in particular queue. When the some task unit crossed that threshold value then the task unit automatically is moved to higher queue. When task reaches the highest priority queue it is send to the required component it requested. Finally by comparing the average values and variance of priorities by processing time the DPSA comes out to be more efficient than FCFS and SPSA [6].

2.3. Non-Dominated Sorting Genetic Algorithm II: This algorithm has been proposed as a solution for Multi-objective optimization for virtual resources. When one request is made for any resource then the virtual resources scheduling is mapped onto physical resources with proper load balancing which is very complex to achieve. This algorithm is in comparison with rank, random and static algorithm. The layer of virtualization occurs between users and physical layer and it has three characteristics usability, safety and moving. They come from independency of virtualization. The virtual resources are abstracted by making number of instances of actual physical resource nodes with attributes.

This algorithm is considered to be heuristic so it contains object functions, code and searching method. Object functions are there for load balancing calculation. NDSA II includes firstly non dominated sorting (set finding the lowest value of object function) and then crowd degree (lower). The GA has selection, crossover and mutation. In this algorithm the selection is tournament selection, crossover is two point crossovers and in mutation if the random number being chosen is the original gene is replaced by randomly generated one. By taking into consideration the CPU usage, memory and bandwidth the NDSA II comes out to be a better algorithm then rank, random and static algorithm as it provides many choices by running just once efficiently [7].

2.4.Optimizing Virtual Machine for High Performance Computing: It is a HPC aware novel scheduler implemented on Open Stack Scheduler. It is topology awaked and homogenously allocating virtual machines. Cloud computing is of the lot of help to those who cannot afford large clusters has replaced supercomputers in some cases. Commodity interconnects performance variability and performance virtualization which indicates that cloud is suited for some HPCs. There are only few efforts on virtual machine algorithms that take into account the HPC. Open stack and Eucalyptus provide a minor effect of HPC. HPC aware strategies (topology awareness and hardware awareness) have been implemented which improves performance by allowing cloud providers to better utilize the infrastructure making more profits. Open stack is a scheduler which selects a physical resource where VM is provisioned.

Open stack receives VM request as part of RPC message. Host capability is an important input to scheduler which contains list of physical servers and their capabilities. The scheduling algorithm contains two steps: Filtering (exclude hosts which do not have required capability) and Weighing (computes fitness of filtered list using cost functions (e.g. free memory in a host)). Then by sorted list of hosts VM provisioning takes place. While scheduling Open Stack do not consider application type, priorities, processor heterogeneity and network topology.

HPC-Aware Scheduler: There are two techniques involved: Topology awareness (as user is unknown of the cluster the VMs are packed to nodes in same rack compared to any placement policy which distribute them over the cluster) and hardware

awareness/homogeneity (cloud users unaware of underlying hardware where VMs are placed by ensuring that all VMs are allocated some task).

The first modification is to switch the use of group scheduling for considering k VMs problem as a single scheduling problem. Firstly topology aware algorithm runs as described next filtering phase (making a list then maximum number of servers) then using this build plan. For homogeneity the scheduler groups the hosts then applies algorithm to those groups taking into consideration the configuration (currently CPU frequency).

The suitability of platform for an HPC application depends on application characteristics, performance requirements and user preferences. The main focus is HPC applications which are comprised of k parallel instances requiring synchronization and allocating VMs in topology aware manner to provide good list of VMs to application user. Its future work includes mixture of HPC and non-HPC applications [8].

2.5. Scheduling with Parallel Genetic Algorithm (PGA): This algorithm was devised to solve the problem of Unbalance Assignment problem to achieve the maximum efficiency. The existing strategies are not good to handle the scheduling so the GA turns out to be a good choice in case of scheduling. PGA improves performance and scalability. It can be implemented on parallel mainframes and heterogeneous computers. This algorithm helps in finding the best possible scheduling sequence on IaaS (Infrastructure as a Service) cloud giving better results than Rank algorithm, Round Robin algorithm, greedy technique, PBS and SGE.

This algorithm contains three main steps: firstly system sets an idle resource and VM list, update each time VM requests occur. Secondly run PGA to get optimal allocation sequence. Thirdly allot physical machines to VM requests. Chromosome representation, fitness function and design are the according to the requests. The PGA operators include some of the parameters like topology, migration rate, and migration scheme and migration interval. It can run on more than one virtual machine at any instant. This algorithm till now has been implemented on Windows XP using Java

Genetic Algorithm Package (JGAP). PGA improves the speed of resource allocation then Round Robin and Greedy algorithm [9].

2.6. Balance Reduce Algorithm (BAR) (Fault Tolerant): This algorithm is based on data locality driven reducing network access thus reducing bandwidth usage and job completion time. This algorithm also handles the machine failure. Initial local task allocation in balanced phase takes place and then job execution time can be reduced by matching initial task allocation in reduced phase. The machine failure is handled by algorithm similar to primary backup approach.

This algorithm helps in reducing the required output by using the special key. In this algorithm the large computations can be broke into small pieces and run parallel in multiple machines inexpensively so a machine failure may lead to whole computation failure and thus leading to large job completion time if server fails. The BAR algorithm has two phases: Balance (all tasks allocated to preferred servers and make the data local i.e. read from servers local disk) and Reduce (balanced allocation is modified to minimize job completion time). Fault tolerant function is a reactive measure and solves failures. In this algorithm the failure detection needs to be done with some mechanism or device and it tolerates the failure of one node in the system. The fault tolerant algorithm reduces the make span to a quantifiable time from infinite time and tasks on the failed machine are rescheduled to other server for completion. As for the cloud based services are more of the data processing and network bandwidth is limited so the scheduling based on data locality is crucial [10].

2.7. Heavy Traffic Optimal Algorithm: The join-the-shortest-queue routing and power-of-two-choices routing with MaxWeight scheduling is optimal in throughput and they are queue length optimal in high traffic loads. Calculating the exact queue length is quite difficult so the system in heavy traffic regime (exogenous arrival rate is almost same as boundary of capacity region) was studied. Use of state space collapse (multi dimensional state reduces to single dimension) was there. The algorithm is applied on multiple models supported by multiple servers. Above models assume system is work conserving. Then the result converges to regulated Brownian notation and simple path optimality in scaled time. The method in the heavy traffic optimality

is simpler and also in unscaled time consists of three steps: lower bound (weighted sum of expected queue length by comparing with a single server queue), state-space collapse (state of system collapsing to single dimension, queue length in particular direction increases and in perpendicular direction it is bounded) and upper bound (obtained by natural Lyapunov function). Heavy traffic is obtained when upper and lower bound coincides. The solution contains one routing and one scheduling algorithm.

This is the stochastic model for load balancing and scheduling in clusters. The JSQ and MaxWeight is throughput optimal and traffic optimal when all servers identical. And also the power-of-two-choices is also heavy traffic optimal [11].

2.8. Optimized Resource Scheduling: The optimization can be done on the basis of SLA (Service Level Agreement) as the resource scheduling problems are NP hard problems. The stochastic integer programming which further uses Grobner bases theory to extend Minimized Geometric Buchberger Algorithm is applied addressing SLA aware resource composition problem. This optimal solution is based on reasonable short time. This whole model used is further proposed for an optimal solution of resource composition model [12].

2.9. Cost Based RS paradigm: In this scheduling technique the resources are allotted as leveraging marketing theory to make the maximum use of the resources available. This algorithm as well as protocol is designed for IaaS. The allocation is according to the resource availability and price. This algorithm reduces overhead of running the algorithm in cloud environment resulting in perfect balance of complexity and performance. This whole algorithm is implemented on a private cloud environment [13].

2.10. Double Auction-based Scheduling of scientific applications in Distributed Grid and Cloud environments: This paper introduced a negotiation protocol between scheduler and resource manager using market-based Continuous Double Auction model. Different scheduling strategies are analyzed and based on it the results are demonstrated. It manages the access of resources in open market in order to minimize the cost of application execution. The self limitation based scheduling

brings small improvements. Improvement in execution is achieved by aggressive strategy. The strategy can be chose by scheduler's attitude for risks [14].

2.11. Algorithm based on Trust Degree: This algorithm takes into consideration the functional characteristics and provides better stability and low risk while completing tasks. It reduces threshold and risks in small and medium enterprises. The trust degree is determined by execution time and reliability. Scheduling logs stores trust degree at any time and sort it decreasingly and then the computer slots are called according to whichever's trust degree is greater first. This algorithm is stable and reliable [15].

2.12. PSO based hierarchical strategy: The Particle Swarm Optimization (PSO) based algorithm includes both transmission cost and present load. A novel inertia weight is also included for getting the global as well as local search effectively. This all supports minimizing inter network costs and balancing the load. This algorithm optimizes schedules of workflow application in cloud. The proposed algorithm yields effective performance on scheduling algorithm [16].

2.13. Dynamic resource scheduling and workflow management: An economic algorithm with business parameters which determines the trade off between effectiveness and performance. An market oriented workflow architecture is introduced to meet customer demands and enhances the efficiency of the algorithm. This enhancement is done to improve dynamic algorithm with already predictive resource mechanisms. This solution helps in sustaining the consumers operations with different priorities [17].

2.14. SLA Restriction for cloud based in Pareto Optimality $M \times N$ production: This method is economic based with realization of optimal allocation of resources over a cloud. The strategy focus on resource scheduling in Cloud Bank model solving problem of resource allocation among self interested individuals. It tries maximizes the benefits of the parties involved. Its allocation is dynamic and takes into consideration the time consumption nature of the scheduling as to cater to the needs of

the users and it is one of the important parameter in scheduling algorithms. The user requirements are characterized by a utility function [18].

2.15. Loyalty based resource allocation: The trust concept is introduced in architecture and loyalty which improves the successful transaction rate of the system while meeting the requirements. Using Master Slave framework a role based access control has been proposed considering the trust of the node and meets the requirements using the services. The unreliability of hardware should be provided by highly reliable software. It assesses the real time condition of the system and allocates resource according to condition. This dynamic feedback mechanism provides stability and reliability of services [19].

2.16. Dynamic and integrated load balancing algorithm: It treats CPU, bandwidth and memory for physical as well as virtual machines. Total measurement of cloud datacenter imbalance level and average for server imbalance level are presented. This algorithm shows good performance with regard to imbalance level and overall running time as it has extra good features [20].

2.17. Adaptive optimal global resource scheduling: It employs linear programming algorithm to reduce extra cost for power consumption and other expenditures with solid restrictions on networking environment. It promotes resource utility through finely grained resources and in depth restrains expenditure analysis for remote access by taking into account resource configuration, real time load and service deployment trade offing between performance, computation cost and response time. A greedy algorithm for small scale pool with many networking resources is provided [21].

2.18. Load balance based algorithm: The data processing power of the nodes and data transferring power of the nodes and transfer delay between nodes is considered. Algorithm selects best node to complete the task to improve efficiency, minimize average response time of the tasks. These calculations are made on the basis of the dynamic load of the nodes in particular cloud. The prediction of time needed to complete the task is done resulting in increasing efficiency, reducing average response

time and increasing throughput. The supposition that time to finish the task can be predicted is considered for this algorithm [22].

2.19. Green power management for virtual machines: It includes 3 phases: Virtualization management, Dynamic resource allocation mechanism and green power management. The green power management is presented to reduce the load balancing for the virtual machine management. It supports green power mechanism applied on virtual machine resource monitor. Expected improvement contains violent CPU highly loading solution. It shows energy saving feature with setting of sensitivity parameters and also considering perfect smooth virtual changes [23].

2.20. Component based resource allocation: This allocation model provides future resource allocation and managing need in cloud computing. The future perspective refers to whenever a new node is added to the cloud it combines them with the existing without much complication. The information generated by the component resource when new nodes are being added to the cloud will be of utmost importance. Functionality of node can be added to any component at any time to provide enhancements [24].

2.21. Pareto based optimal scheduling: The cloud banking model is introduced with features like multi dimensional Pareto optimal theory and optimization analysis aiming at improving resource utilization as well as consumer satisfaction. This algorithm characterizes the user's requirements based on above features. It takes into consideration resource prices and execution time [25].

2.22. Smart Dynamic resource scheduling algorithm: This algorithm contains 2 steps mainly prediction technique to fits the cloud data center well and novel and efficient migration technique to strike a balance between cost, instantaneity and efficiency. In this management prototype resource scheduling is just a module which has high forecast accuracy and can deal with load balancing and load consolidation. This paper presents a dynamic scheduling strategy which employs Single Exponential Smoothing (SES) algorithm for prediction of resource utilization and Vector Projection (VP) for second step. It doesn't consider spending of migration [26].

2.23. Genetic algorithm with MultipleFitness: It is a pre migration strategy which considers 3 parameters: disk I/O rate, network throughput and CPU utilization. For optimal solution a hybrid of genetic algorithm and knapsack problem is considered. This algorithm raises resource utilization and lower energy consumption cost by runtime resource scheduling under cloud environment. It smoothes load utilization [27].

2.24. Hybrid multidimensional algorithm for network aware scheduling: This distributed resource allocation algorithm is capable of handling multiple resources requisites for tasks that are arriving to computing environment. The tradeoff between execution time and cost of data intensive is considered by taking performance parameters at system and network level on economic and computational basis. It has an advantage of knowledge of grid infrastructure [28].

2.25. Pricing algorithm: Cloud bank as resource agency provides analysis and guidance for participants and a price update iterative algorithm analysis the historical utilization ratio and iteration current prices. It also gets the availability of resources and final price to consumers. This algorithm is designed to safeguard the interests of the participants in cloud. With this resources achieve macro control. It is not adaptive as it cannot adapt rapid changes of demand and supply. It reduces the cost of providers, maximizes revenue and is more conducive to keep the providers interests [29].

2.26. Load adaptive model based on ant colony algorithm: This algorithm monitors real timely virtual machines on performance parameters and schedules fast resources using any colony algorithm. It is made accordingly to bear load on a load free node to meet the changing load requirements improving resource utilization's efficiency. The detection of overload exceeds the threshold limit. This algorithm finds the nearest idle node and allows it to bear some load meeting the performance and resource requirements of load thus achieving the goal of load balancing [30].

Table 2.1 Comparison between some of the existing algorithms [1], [3]

Scheduling algorithms	Description	Scheduling parameters	Cloud environment
A Particle Swarm Optimization-based Heuristic for Scheduling Workflow Applications	Heuristic approach to schedule cloud resources taking into account both computation cost and data transmission cost, used for workflow application by varying its computation & communication costs	Resource utilization, Time	Amazon EC2
A Revised Discrete Particle Swarm Optimization for Cloud Workflow Scheduling	optimizes the schedules of workflow application, each particle learns from different exemplars, but also learns the other feasible pairs for different dimensions	Cost, Makespan	Amazon EC2
RASA Workflow Scheduling	Works in Batch Mode, used to reduce span, tasks are grouped,	Makespan	GridSim
Ant Colony Optimization Based Service flow Scheduling	Various quality of service (QoS) requirements, adopt the use of an ant colony optimization (ACO) algorithm to optimize service flow scheduling	Reliability, Response time, Cost, Security	Java environment
SHEFT workflow scheduling algorithm	Used to schedule a workflow elastically on a Cloud computing environment	Execution time, Scalability	CloudSim
Improved cost-based algorithm for task scheduling in cloud computing	Used for making efficient mapping of tasks to available resources in cloud, measures both resource cost and computation performance, improves the computation/communication ratio	Cost, Performance	CloudSim

A Compromised-Time- Cost Scheduling Algorithm	Considers the characteristics of cloud computing to accommodate instance-intensive cost constrained workflows by compromising execution time and cost with user input enabled on the fly	Cost, Time	SwinDeW-C
A Multiple QoS Constrained Scheduling Strategy of Multiple Workflows Scheduling	Schedules multiple workflows which are started at any time and the QoS requirements are taken into account, affect the total makespan and cost of workflow greatly.	Success rate, Cost, Time, Makespan	CloudSim
Scheduling with Parallel genetic Algorithm	Used to solve Unbalance Assignment Problem, improves speed of resource allocation	Topology, migration Rate, Scheme, Interval	Windows XP with JGAP
Genetic Algorithm with Multiple Fitness	Hybrid of genetic algorithm and knapsack problem	Disk I/O rate, network throughput , CPU utilization	—

Chapter 3

Proposed Priority Scheduling Algorithm

In this chapter, gaps in existing system, problem statement, objectives to achieve and methods to achieve the objectives are discussed.

3.1 Research Gaps

In the previous chapter of Literature Review the different algorithms were discussed and explained accordingly how they were applied in different environments. Following are the gaps in existing algorithms:

- 1) The existing algorithms are not all fault tolerant as in real time environment the fault may occur at any node at any time.
- 2) The existing algorithms are not reliable enough to provide efficient resource scheduling environment.
- 3) These algorithms are not at all energy efficient.

3.2. Problem Statement

There are various algorithms, models and techniques already implemented based on features like cost, availability, efficiency and virtualization but these techniques lack reliability and availability. Also there are not much energy efficient algorithms in cloud resource scheduling. The fault tolerance property is yet not implemented in the any of the algorithms except Balance Reduce Algorithm (BAR). The fault tolerant property of the algorithm determines its reliability. Therefore there is need of more energy efficient and fault tolerable algorithms.

3.3. Objectives

- 1) To study existing algorithms for resource scheduling
- 2) To propose an algorithm with property of fault tolerance
- 3) To implement proposed algorithm

- 4) To test implemented proposed algorithm

3.4. Methodology

- 1) Literature Review

The present algorithms till now proposed are being explored and the functionality of the particular algorithms are being studied and also parameters on which they work is also explored.

- 2) Propose algorithm using attributes and cloud.

The attribute of priority is taken into consideration with fault tolerance functionality.

- 3) Implementation is as follow:

- Install cloud on system
- Takes into account few servers and tasks on those servers
- Training algorithm for various inputs

- 4) Analyze the results for other inputs.

Various test cases are designed to explore functionality of the algorithm.

3.5. Proposed Algorithm of Priority Scheduling

S[i] list of servers (max n)

T[i] list of tasks (max m)

P[i] priorities if tasks= number of T[i]

Sh[i] scheduled list= number of T[i]

Shf[i] scheduled list of faulty server

W[i] waiting list if sh[i] is full (max p)

S be Status of server (working or faulty) [W/F]

Loadserver[i]= number of tasks in particular server

$$\text{TotalCloudLoad} = \sum_{i=1}^m \text{Loadserver}[i]$$

Let the current load of the cloud at some instant be cloud

Algorithm

Priority scheduling

- Initialize the priorities for tasks for every server $s[i]$ ($t[i]$, $p[i]$, $sh[i]$, s , $loadserver$, $w[i]$)
 $Loadserver = \text{count } sh[i] + w[i]$
 $Sh[i] \text{ total} = t[i] \text{ total}$ i.e. number of tasks on the server.
- Perform scheduling according to the priority given with the highest priority first and update $Sh[i]$ list for every server.
- When new task arrives check $loadserver$
 If $loadserver \leq 15$
 Skip task to next server
 End if
 Else
 Check $t[i]$, $Sh[i]$. if the lists are full put the task in waiting list
 End else

Fault tolerance

- After regular interval check every server's status
- If status = 'w'
 Go to next server status
 End if
 Else if status = 'f'
 Fetch the current server's $Sh[i]$ to $Shf[i]$
 Find min $loadserver \ s[j]$
 Allocate $shf[i]$ to $s[j]$ server
 End else if

Min loadserver (s[i])

- Check for minimum loadserver for every server and return the particular server name

Allocate task to server(s[i])

- shf[i] to s[j]'s sh[j]
- while shf[i] not empty & sh[j] not full
- The s[j]'s new sh[j]=sh[j]+shf[i]
- Update loadserver of s[j]

This algorithm will firstly schedule the tasks present on each server at some instant and then it will further move to addition of the new tasks by assigning their priorities. This algorithm mainly focuses on the fault tolerance nature of the algorithm to handle the fault. The number of servers in fault mode that cloud can hold can be determined by a following given formula.

Scenario when the number of tasks running on each server is same:

Number of faulty servers that can be handled

$$= \frac{\text{Total Cloud Load} - \text{Current Cloud Load}}{\text{maximum number of tasks running on a single server}}$$

In this Current Cloud Load will be calculated by adding all the individual loads of the servers at that instant.

Scenario when number of tasks running on each server is different:

Number of faulty servers that can be handled

$$= \frac{\text{Total Cloud Load} - \text{Current Cloud Load}}{\text{number of tasks running on a server with maximum load handling capacity}}$$

Now the priority if the scheduled list of the faulty server is allocated to another server the priorities allocated to them is as follows:

1. The priorities of tasks of the server $S[j]$ to which the tasks are allocated will remain same.
2. The new list $shf[i]$ to be embedded from other server will be embedded just after the current list $s[j]$ of the server.
3. And when the whole $shf[i]$ is embedded after $s[j]$ next coming tasks will be given the next available priorities.

This algorithm takes number of tasks per server as a parameter for load calculation and determine on server value of load and helps in determining server with minimum load for reallocation. In this algorithm the fault of the server is being detected by virtual machine selection policy Minimum Migration Time (MMT). This policy understands that when the tasks migration time remains the minimum it is considered that task is not been transferred or the machine is faulty. So this policy works in here.

Design and Implementation Details

This chapter includes the implementation details for priority scheduling with fault tolerance algorithm. Java NetBeans with CloudSim is used for implementation. It integrates the functionality of whole algorithm with input one gives. As discussed in Literature review there are many algorithms to schedule different aspects of cloud but none of them except Balance Reduce Algorithm was implemented with a fault tolerance technique. All the algorithms have their pros and cons. This section gives brief introduction about CloudSim and presents a flowchart of functionality of the algorithm.

4.1 CloudSim

The layered implementation of the CloudSim software framework and architectural components is shown in Fig. 4.1. At the lowest layer is the SimJava discrete event simulation engine that implements the core functionalities required for higher-level simulation frameworks such as queuing and processing of events, creation of system components (services, host, data center, broker, virtual machines), communication between components, and management of the simulation clock. Next layer implementing the GridSim toolkit that support high level software components for modeling multiple Grid infrastructures, including networks and associated traffic profiles, and fundamental Grid components such as the resources, data sets, workload traces, and information services. CloudSim provides novel support for modeling and simulation of virtualized Cloudbased data center environments such as dedicated management interfaces for VMs, memory, storage, and bandwidth. CloudSim layer manages the instantiation and execution of core entities (VMs, hosts, data centers, application) during the simulation period. This layer is capable of concurrently instantiating and transparently managing a large scale Cloud infrastructure consisting of thousands of system components. The fundamental issues such as provisioning of hosts to VMs based on user requests, managing application execution, and dynamic monitoring are handled by this layer. A Cloud host can be concurrently shared among

a number of VMs that execute applications based on user-defined QoS specifications. The top-most layer in the simulation stack is the User Code that exposes configuration related functionalities for hosts (number of machines, their specification and so on), applications (number of tasks and their requirements), VMs, number of users and their application types, and broker scheduling policies. A Cloud application developer can generate a mix of user request distributions, application configurations, and Cloud availability scenarios at this layer and perform robust tests based on the custom Cloud configurations already supported within the CloudSim.

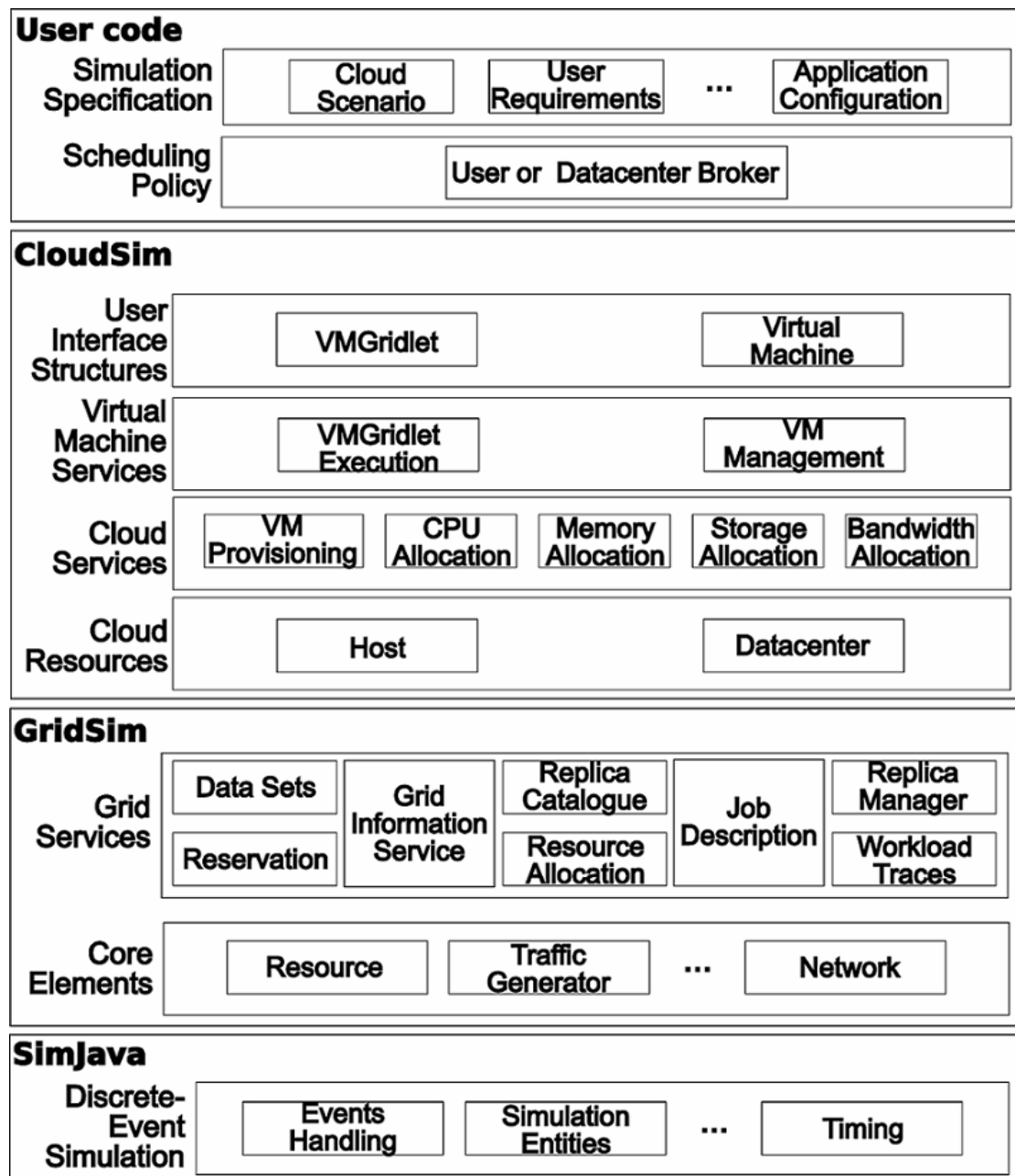


Fig. 4.1 Layered Architecture of CloudSim [31]

4.2. Flowchart of the Proposed Algorithm

These flowcharts will determine the flow of control as how the initialization and the navigation of the algorithm take place in the cloud system.

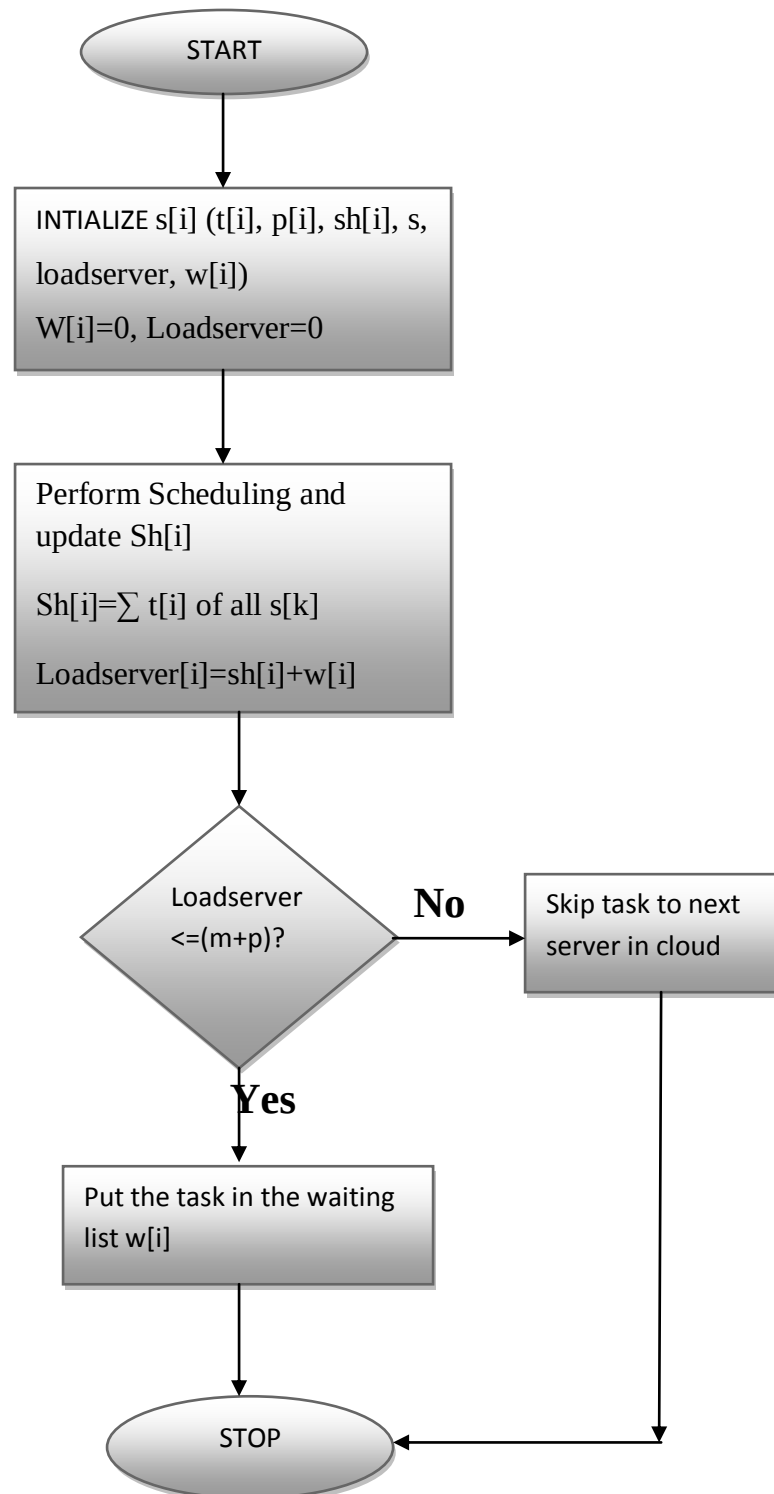


Fig. 4.2 Flowchart for priority scheduling of initial arriving tasks

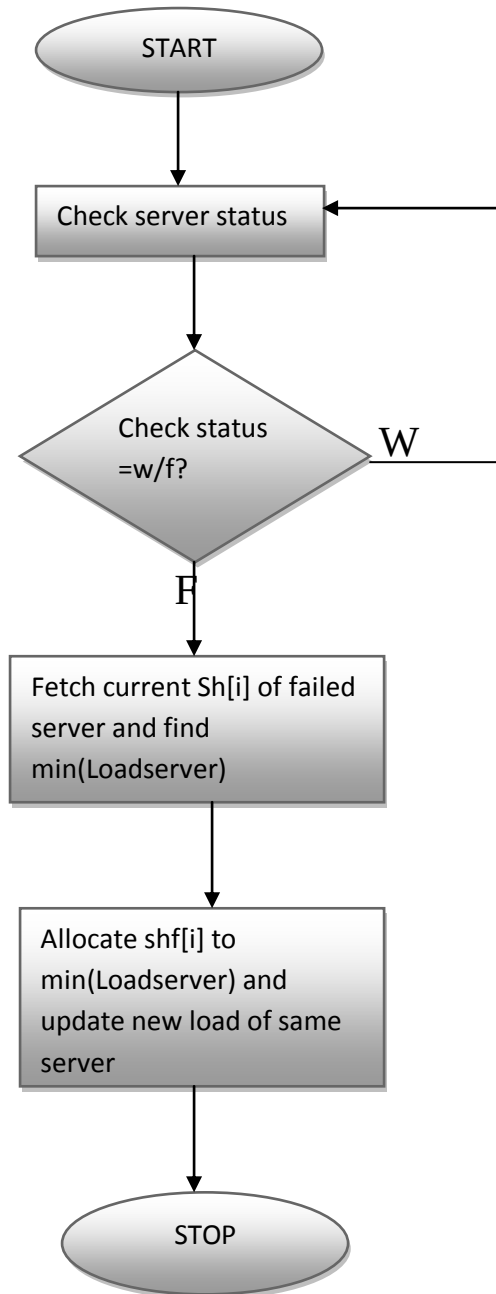


Fig. 4.3 Flowchart for detecting fault in any of the server and reallocating the tasks to another server

4.3. Implementation Details

This part provides somewhat details about some of the implementation that has been done using NetBeans, CloudSim and changes made.

Some of the implementation done is on the basis of the firstly the NetBeans 7.1.2 is installed successfully and then the CloudSim is imported into the NetBeans. After that the different jar files are imported into the libraries of CloudSim and jdk used is 1.7. After successful installation the CloudSim will run successfully and finally the algorithm implemented can be run into the cloud. Input to the algorithm can be given from the example workloads given in the CloudSim package itself.

The fault detector Minimum Migration Time detects through tasks migration timing. These tasks are migrated to the next priority with the execution of the each task. If the migration time of some particular task is minimum then that server is considered to be faulty as it is not providing resources resulting in non execution of the tasks.

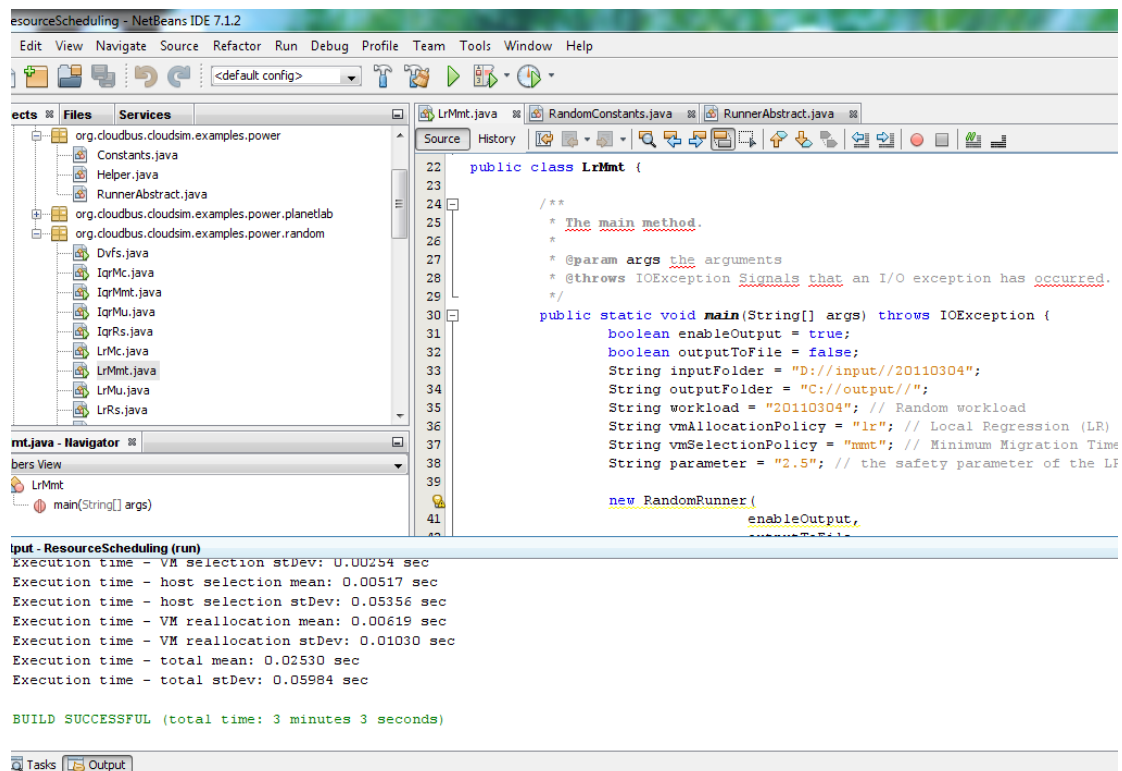


Fig 4.4 Screenshots describing input and assigning values

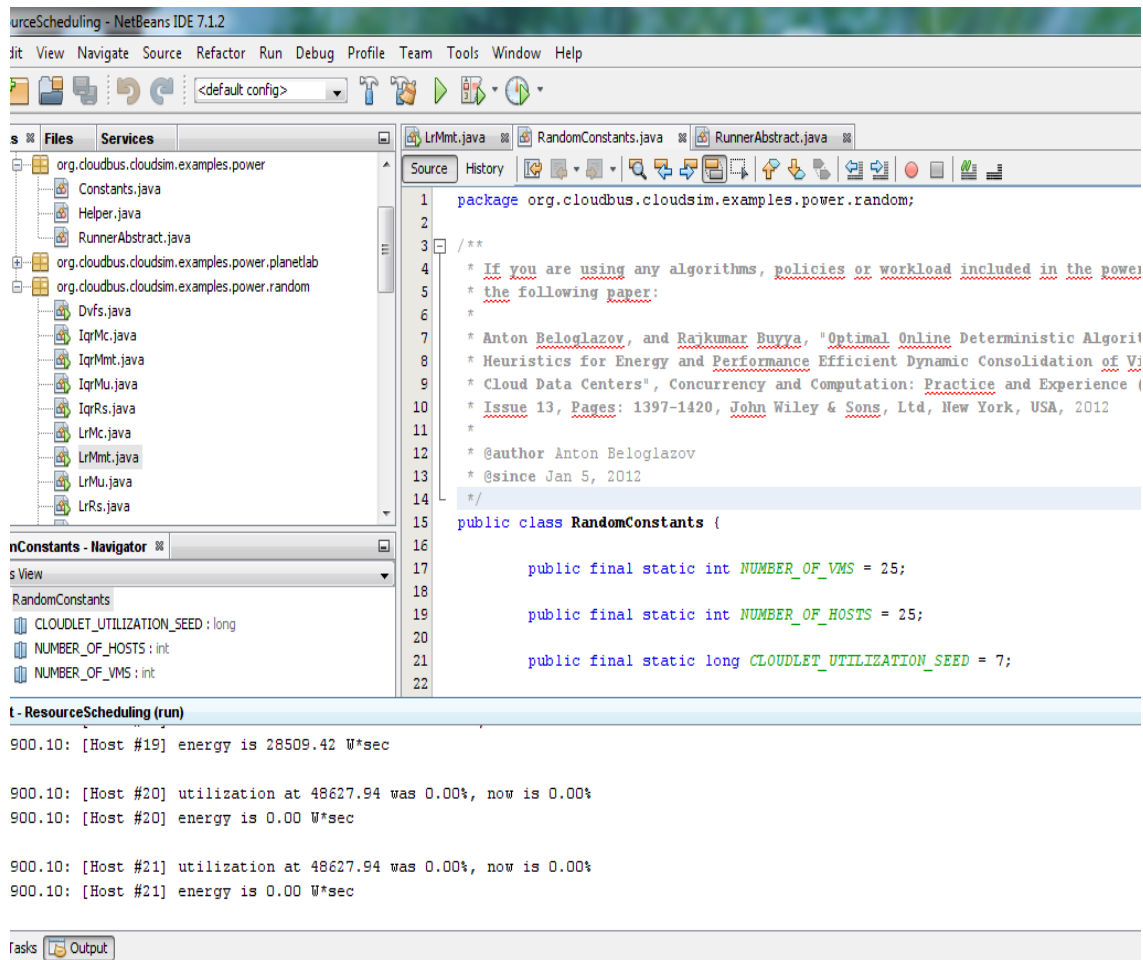


Fig 4.5 Screenshots describing assignment of different constants

This chapter presents the types of testing and experimental results that are being taken by inserting different test cases into the implemented method.

5.1. Types of testing performed

This presents types of testing done for the implementation system.

5.1.1. Unit Testing

Each part/unit of the algorithm has been tested:

- priority scheduling
- fault tolerance
- finding minimum load server

This includes testing of each function by providing it the input in different environments and input.

5.1.2. Integration Testing

The system has been tested by integrating the different units and collectively calculating the results by inserting different test cases. In this all the priority scheduling with fault tolerance functionality is considered and fault detection is provided by virtual machine selection policy Minimum Migration Time (MMT).

5.1.3. System Testing

This testing has been done after integrating the whole system into a single system and providing different inputs to the system as a whole system to be considered.

5.1.4. Alpha Testing

The system has been thoroughly tested by developer as to provide the functionality it is being asked for. The objectives are tested by the developer one by one to complete the final execution. This testing is done by inputting different test cases as parameters.

5.2. Test Cases

In this part there are different sets of test cases and the results. Accordingly to the results the analysis of the different scenarios is described.

In this the parameters that are taken into account is total mean, total standard deviation of the execution time and safety parameter which into a single scenario the different readings are taken.

5.2.1. Test Case 1

In this Test case: Mean and standard Deviation is in milliseconds (ms) and Execution time (sec)

Number of Virtual Machines: 5

Number of Hosts: 5

Number of Utilization Seeds: 2

Table 5.1 Experimental Results for first test case scenario

Sr. No.	Safety Parameter	Mean(ms)	Standard Deviation(ms)	Execution Time(sec)
1.	0.5	3.85	5.78	53
2.	1.0	3.97	6.31	56
3.	1.5	5.16	6.69	61
4.	2.0	5.21	6.67	60
5.	2.5	6.96	6.50	69
6.	3.0	5.61	6.36	66
7.	3.5	5.57	5.96	63

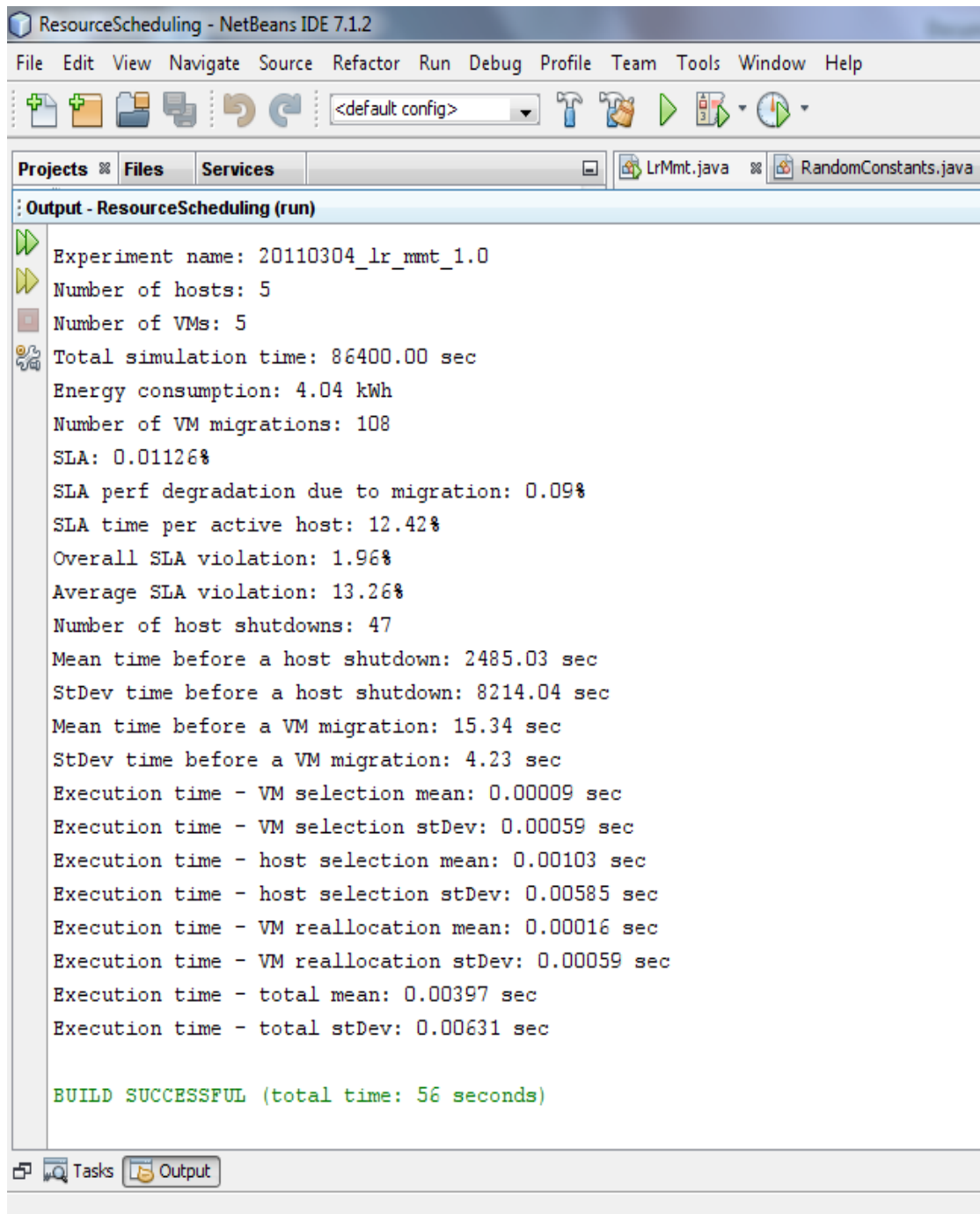


Fig 5.1 (a) Screenshots depicting results for one case in first test case scenario

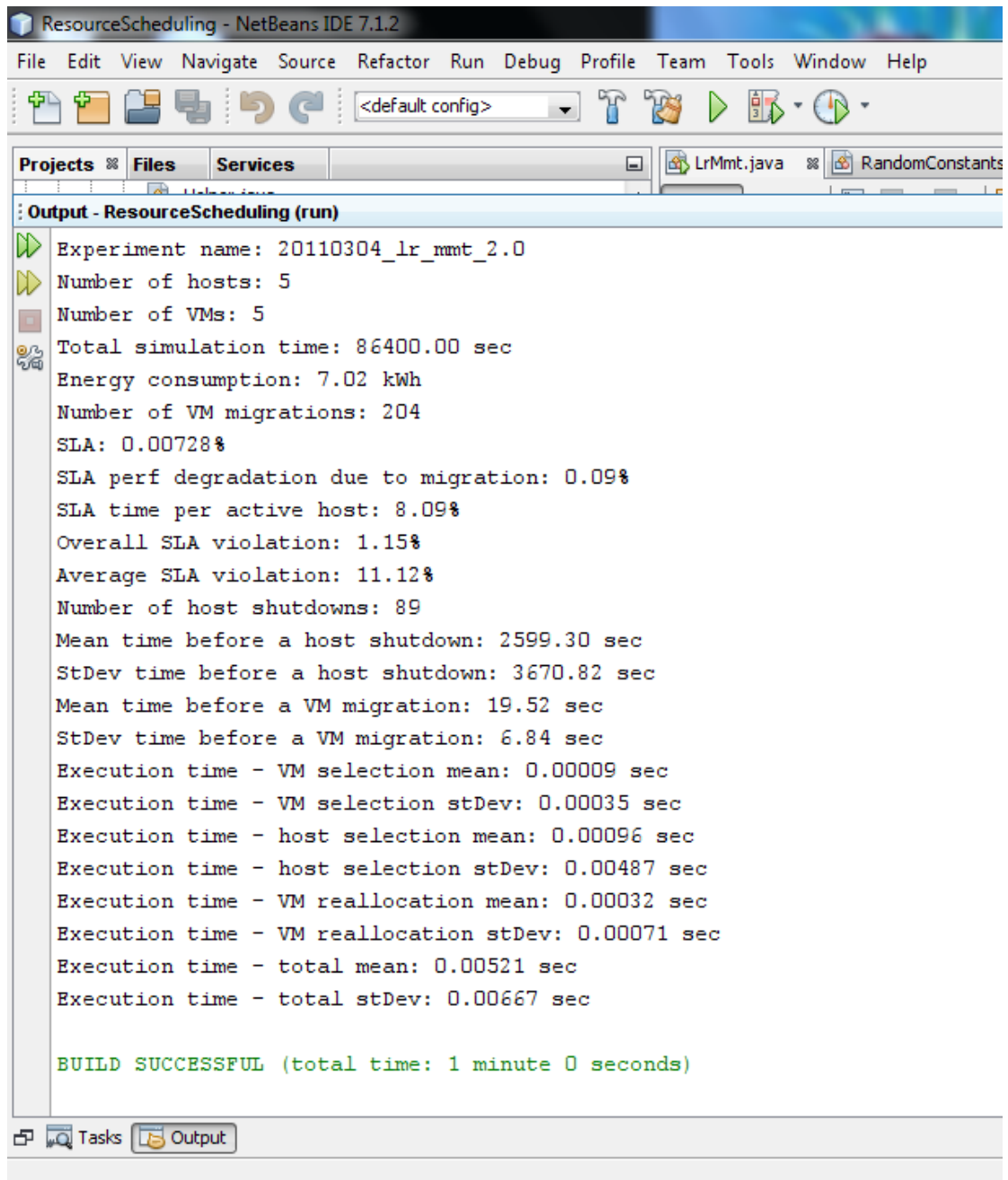


Fig 5.1 (b) Screenshots depicting results for one case in first test case scenario



Fig 5.1 (c) Tradeoff between Safety Parameter and Mean for first test case scenario



Fig 5.1 (d) Tradeoff between Safety Parameter and Standard Deviation for first case scenario

5.2.2. Test Case 2

In this Test case: Mean and standard Deviation is in milliseconds (ms) and Execution time (sec)

Number of Virtual Machines: 10

Number of Hosts: 10

Number of Utilization Seeds: 5

Table 5.2 Experimental Results for second test case scenario

Sr. No.	Safety Parameter	Mean(ms)	Standard Deviation(ms)	Execution Time(sec)
1.	0.5	5.56	6.07	68
2.	1.0	6.23	6.08	79
3.	1.5	7.35	6.22	81
4.	2.0	9.04	12.16	91
5.	2.5	9.22	8.79	88
6.	3.0	10.54	15.56	105
7.	3.5	11.07	16.53	112

```

ResourceScheduling - NetBeans IDE 7.1.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
[Icons] <default config> [Icons]
Projects Files Services LrMmt.java RandomConstants.java
Output - ResourceScheduling (run)
Experiment name: 20110304_lr_mmt_1.5
Number of hosts: 10
Number of VMs: 10
Total simulation time: 86400.00 sec
Energy consumption: 9.64 kWh
Number of VM migrations: 444
SLA: 0.00480%
SLA perf degradation due to migration: 0.10%
SLA time per active host: 4.61%
Overall SLA violation: 0.70%
Average SLA violation: 11.43%
Number of host shutdowns: 158
Mean time before a host shutdown: 1912.23 sec
StDev time before a host shutdown: 3258.72 sec
Mean time before a VM migration: 18.01 sec
StDev time before a VM migration: 7.46 sec
Execution time - VM selection mean: 0.00019 sec
Execution time - VM selection stDev: 0.00084 sec
Execution time - host selection mean: 0.00112 sec
Execution time - host selection stDev: 0.00480 sec
Execution time - VM reallocation mean: 0.00056 sec
Execution time - VM reallocation stDev: 0.00126 sec
Execution time - total mean: 0.00735 sec
Execution time - total stDev: 0.00622 sec

BUILD SUCCESSFUL (total time: 1 minute 21 seconds)
  
```

Fig 5.2 (a) Screenshots depicting results for one case in second test case scenario

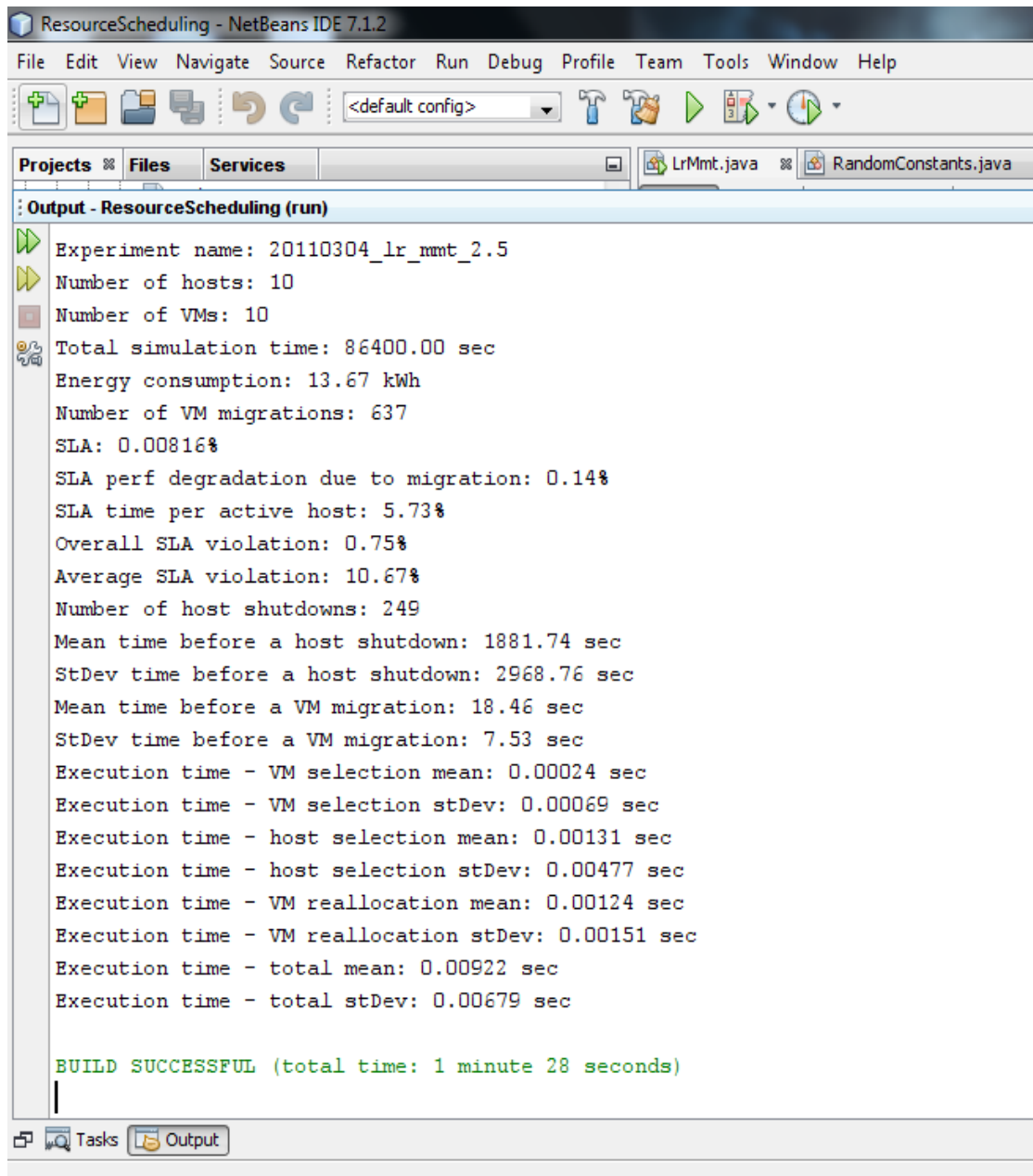


Fig 5.2 (b) Screenshots depicting results for one case in second test case scenario



Fig 5.2 (c) Tradeoff between Safety Parameter and Mean for second test case scenario



Fig 5.2 (d) Tradeoff between Safety Parameter and Standard Deviation for second case test scenario

5.2.3. Test Case 3

In this Test case: Mean and standard Deviation is in milliseconds (ms) and Execution time (sec)

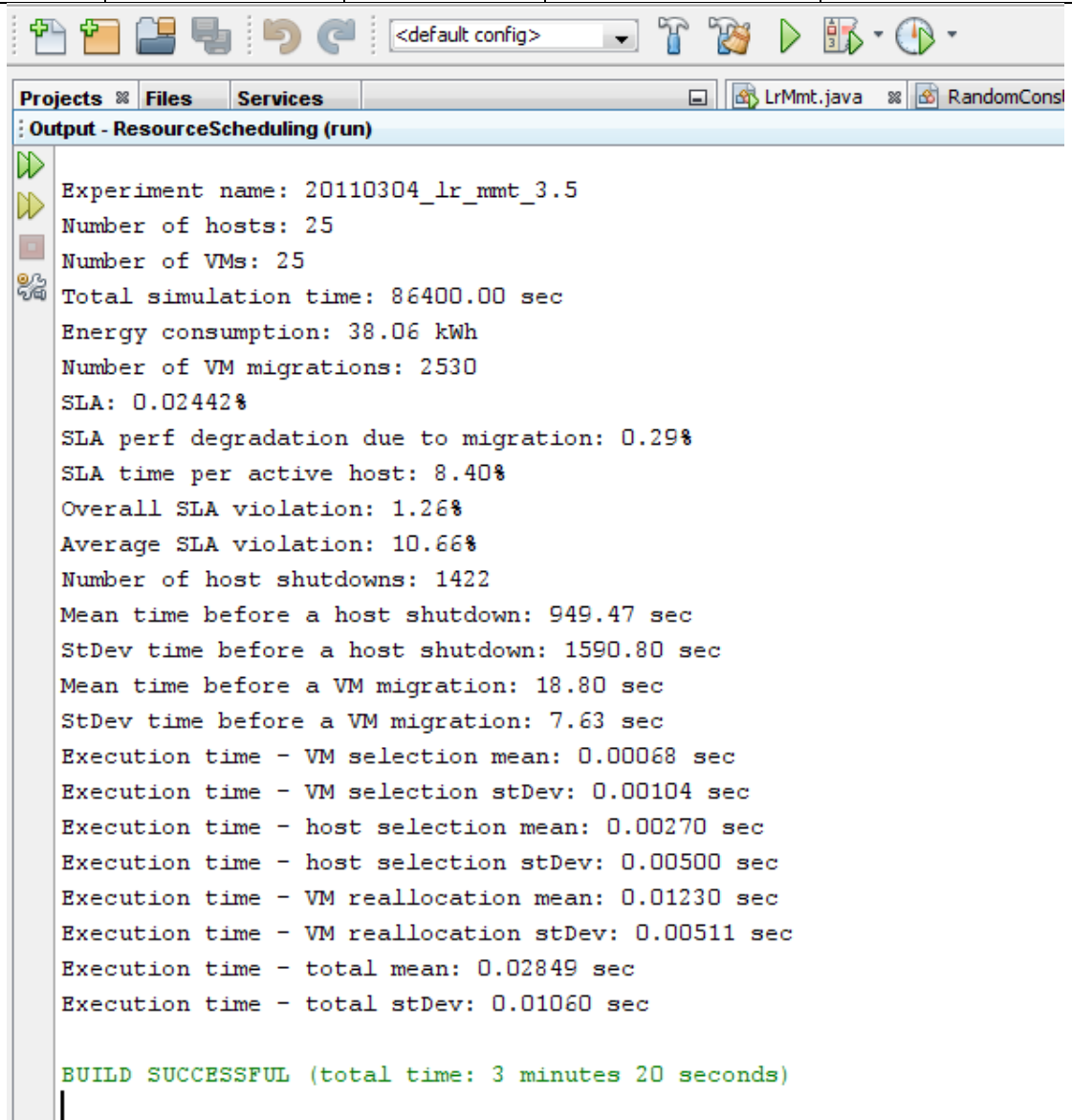
Number of Virtual Machines: 25

Number of Hosts: 25

Number of Utilization Seeds: 7

Table 5.3 Experimental Results for third test case scenario

Sr. No.	Safety Parameter	Mean(ms)	Standard Deviation(ms)	Execution Time(sec)
1.	0.5	9.2	14.71	83
2.	1.0	12.36	15.71	184
3.	1.5	14.77	17.59	207
4.	2.0	18.81	19.77	206
5.	2.5	22.55	11.97	199
6.	3.0	25.51	24.84	235
7.	3.5	28.49	10.60	200



```

Experiment name: 20110304_lr_mmt_3.5
Number of hosts: 25
Number of VMs: 25
Total simulation time: 86400.00 sec
Energy consumption: 38.06 kWh
Number of VM migrations: 2530
SLA: 0.02442%
SLA perf degradation due to migration: 0.29%
SLA time per active host: 8.40%
Overall SLA violation: 1.26%
Average SLA violation: 10.66%
Number of host shutdowns: 1422
Mean time before a host shutdown: 949.47 sec
StDev time before a host shutdown: 1590.80 sec
Mean time before a VM migration: 18.80 sec
StDev time before a VM migration: 7.63 sec
Execution time - VM selection mean: 0.00068 sec
Execution time - VM selection stDev: 0.00104 sec
Execution time - host selection mean: 0.00270 sec
Execution time - host selection stDev: 0.00500 sec
Execution time - VM reallocation mean: 0.01230 sec
Execution time - VM reallocation stDev: 0.00511 sec
Execution time - total mean: 0.02849 sec
Execution time - total stDev: 0.01060 sec

BUILD SUCCESSFUL (total time: 3 minutes 20 seconds)

```

Fig 5.3 (a) Screenshots depicting results for one case in third test case scenario

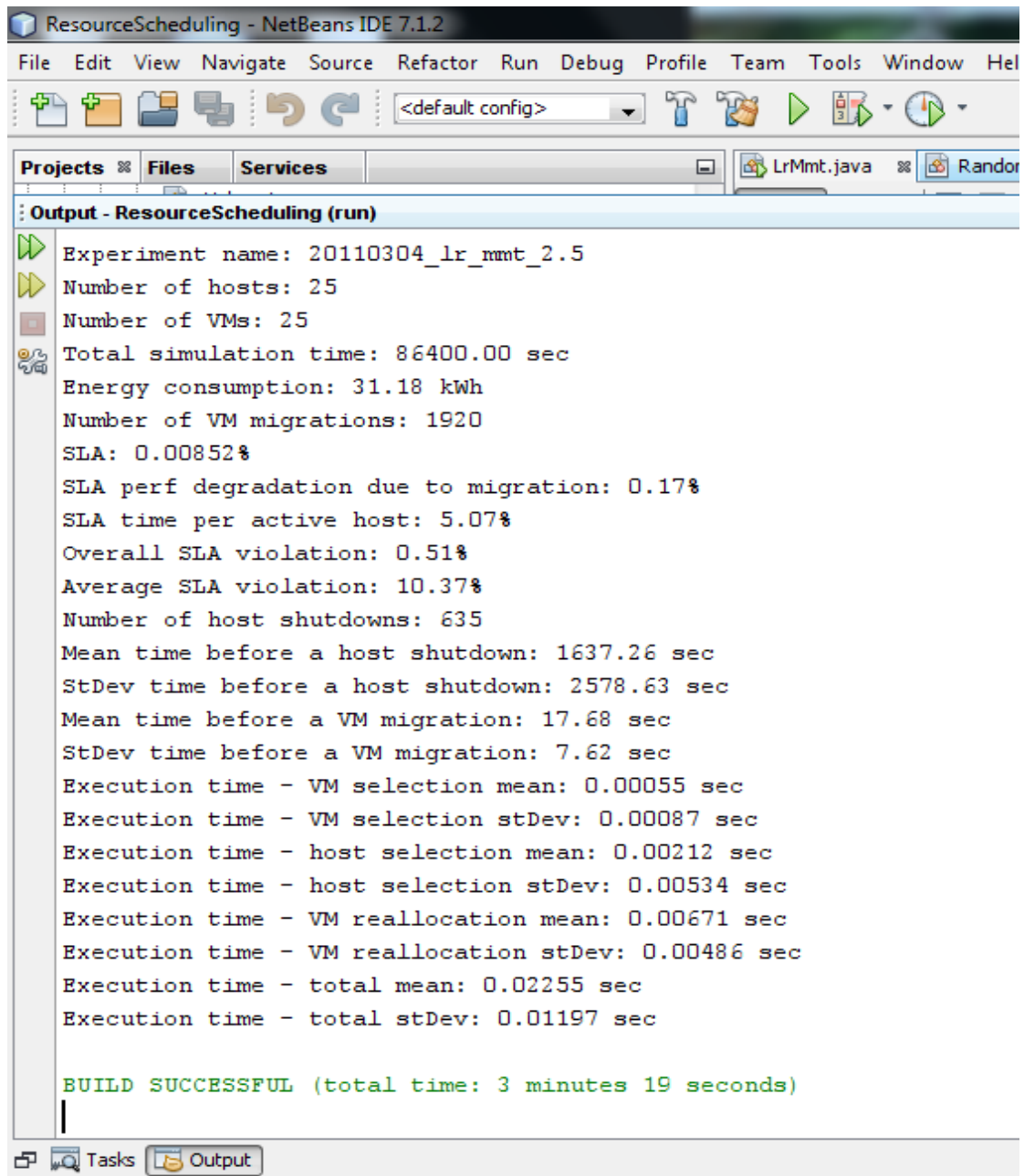


Fig 5.3 (b) Screenshots depicting results for one case in third test case scenario

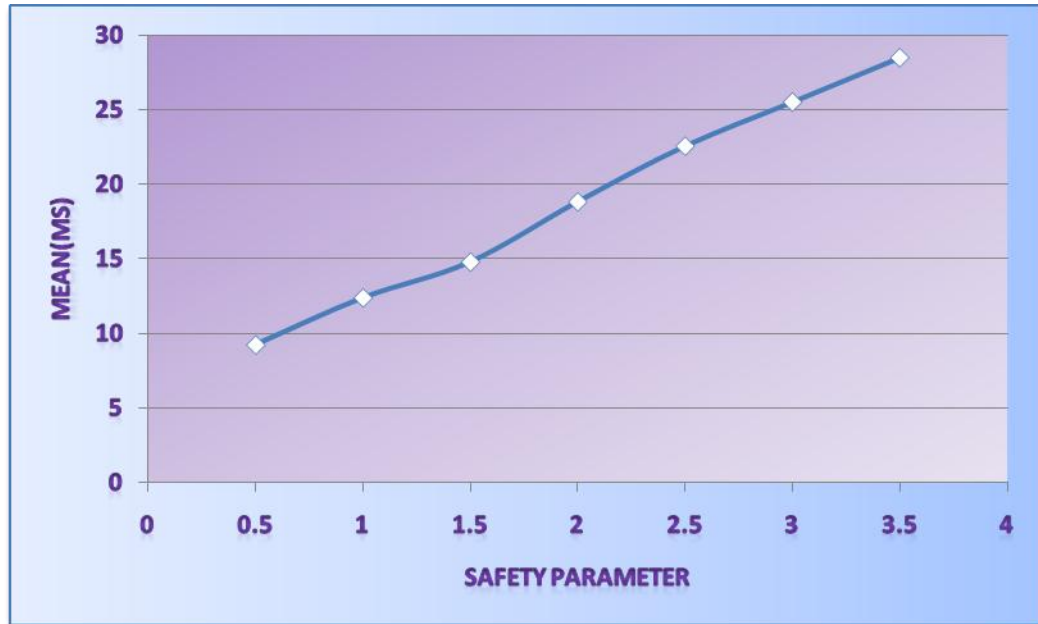


Fig 5.3 (c) Tradeoff between Safety Parameter and Mean for third test case scenario



Fig 5.3 (d) Tradeoff between Safety Parameter and Standard Deviation for third test case scenario

5.3. Analysis of the Results

The readings taken from the above results tell us that when the number of virtual machines, number of hosts and number of utilized is increased the time taken for execution increases. Also the effect of the server where the cloud is installed is also there. As the value of safety parameter is increased so increased is the execution time.

Also when number of VMs, utilization seed and hosts are less and we keep on increasing the value of safety parameter the total mean and standard deviation of the execution time firstly increases and then keeps on decreasing after certain value.

This chapter presents the conclusion and future work related to the proposed method.

6.1. Conclusion

In this thesis the Priority scheduling algorithm has been proposed with the functionality of tolerating a fault in the server included in a particular cloud. The algorithm firstly schedules the tasks according to priority and then reallocates tasks from faulty server to another server. This is better than other algorithms as it provides the fault tolerance functionality and also the results depict the total performance of the cloud increases with the proposed algorithm then available scheduling algorithms as it provides more reliability than other algorithms present till now.

6.2. Future Work

There is need of more energy efficient algorithms in the future as cloud computing applicability is increasing more energy will be needed for the further increasing load in use and more of the services will provided in future. This algorithm can be further improved by introducing dynamic nature to its priority assignment functionality. This algorithm can further extend to different number of architectures that exist in cloud computing system. Also considering topology as a parameter this algorithm works as it works on priority.

- [1] Sujit Tilak, Prof. Dipti Patil, “A Survey on Various Scheduling Algorithms in Cloud Environment”, International Journal of Engineering Inventions, ISSN: 2278-7461, www.ijeijournal.com, Volume 1, Issue 2 (September 2012), pp: 36-39.
- [2] Yogita Chawla and Mansi Bhonsle, “A Study on Scheduling Methods in Cloud Computing”, International Journal of Emerging Trends & Technology in Computer Science (IJETTCS), ISSN 2278-6856, www.ijettcs.org, Volume 1, Issue 3, September – October 2012, pp: 12-17.
- [3] Amid Khatibi Bardsiri and Seyyed Mohsen Hashemi, “A Review of Workflow Scheduling in Cloud Computing Environment”, International Journal of Computer Science and Management Research, ISSN 2278-733X, www.ijcsmr.org, Vol. 1 Issue 3 October 2012, pp: 348-351.
- [4] V.Vinothina, Sr.Lecturer, Dr.R.Sridaran, Dr.PadmavathiGanapathi, “A Survey on Resource Allocation Strategies in Cloud Computing”, (IJACSA) International Journal of Advanced Computer Science and Applications, www.ijacsa.thesai.org, Vol. 3, No.6, 2012, pp: 97-104.
- [5] Liang Luo, Wenjun Wu and Dichen Di, Fei Zhang, Yizhou Yan, Yaokuan Mao, “A Resource Scheduling algorithm of Cloud Computing based on Energy Efficient Optimization Methods”, IEEE 978-1-4673-2154-9, Vol. 12 (2012).
- [6] Zhongyuan Lee, Ying Wang, Wen Zhou, “A dynamic priority scheduling algorithm on service request scheduling in cloud computing”, 2011 International Conference on Electronic & Mechanical Engineering and Information Technology, IEEE 978-1-61284-088-8, Vol. 11 (2011), pp: 4665-4669.
- [7] Jianfeng Zhao, Wenhua Zeng, Min Liu, Guangming Li, “Multi-objective Optimization Model of Virtual Resources Scheduling Under Cloud Computing and

It's Solution", 2011 International Conference on Cloud and Service Computing, IEEE 978-1-4577-1637-9 (2011), pp: 185-190.

[8] Abhishek Gupta, Laxmikant V. Kale, "Optimizing VM Placement for HPC in the Cloud", ACM 978-1-4503-1754-2, September 21, 2012, San Jose, California, USA, pp: 1-6.

[9] Zhongni Zheng ,Rui Wang, Hai Zhong, Xuejie Zhang, "An Approach for Cloud Resource Scheduling Based on Parallel Genetic Algorithm", IEEE 978-1-61284-840-2 (2011), pp: 444-447.

[10] Simy Antony, Soumya Antony, Ajeena Beegom A S, Rajasree M S, "Task Scheduling Algorithm with Fault Tolerance for Cloud", International Conference on Computing Sciences, IEEE 978-0-7695-4817-3 (2012), pp: 180-182.

[11] Siva Theja Maguluri and R. Srikant, Lei Ying, "Heavy Traffic Optimal Resource Allocation Algorithms for Cloud Computing Clusters", ARO MURI W911NF-08-1-0233 and NSF grant CNS-0963807.

[12] Qiang Li, Yike Guo, "Optimization of Resource Scheduling in Cloud Computing", 12th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, IEEE 978-0-7695-4324-6/10, 2010, pp: 315-320.

[13] Zhi Yang, Changqin Yin, Yan Liu, "A Cost-based Resource Scheduling Paradigm in Cloud Computing", 12th International Conference on Parallel and Distributed Computing, Applications and Technologies, IEEE 978-0-7695-4564-6/11, 2011, pp: 417-422.

[14] Radu Prodan, Marek Wiecek, Hamid Mohammad Fard, "Double Auction-based Scheduling of Scientific Applications in Distributed Grid and Cloud Environments", J Grid Computing (2011) 9, pp: 531–548.

- [15] Mingshan Xie, Mengxing Huang, and Bing Wan, “A Resource Scheduling Algorithm Based on Trust Degree in Cloud Computing”, *Software Engineering Research, Management and Applications*, 2012, pp: 177–184.
- [16] Hongli Zhang, Panpan Li, Zhigang Zhou, and Xiangzhan Yu, “A PSO-Based Hierarchical Resource Scheduling Strategy on Cloud Computing”, *ISCTCS 2012, CCIS 320*, pp: 325–332, 2013.
- [17] Xuelin Shi and Ying Zhao, “Dynamic Resource Scheduling and Workflow Management in Cloud Computing”, *WISE 2010 Workshops, LNCS 6724*, pp: 440–448, 2011.
- [18] Huixi Li and Hao Li, “A Research of Resource Scheduling Strategy with SLA Restriction for Cloud Computing Based on Pareto Optimality $M \times N$ Production Model”, *WISM 2011, Part I, LNCS 6987*, pp:155–165, 2011.
- [19] Yanbing Liu, Shasha Yang, Qingguo Lin, and Gyoung-Bae Kim, “Loyalty-Based Resource Allocation Mechanism in Cloud Computing”, *Recent Advances in CSIE 2011, LNEE 125*, pp: 233e–238.
- [20] Wenhong Tian, Yong Zhao, Yuanliang Zhong, Minxian Xu, Chen Jing, “A DYNAMIC AND INTEGRATED LOAD BALANCING SCHEDULING ALGORITHM FOR CLOUD DATACENTERS”, *Proceedings of IEEE CCIS2011, 978-1-61284-204-2/11*, 2011, pp:311-315.
- [21] Lingli Deng, Qing Yu, and Jin Peng, “Adaptive Optimal Global Resource Scheduling for a Cloud-Based Virtualized Resource Pool”, *STA 2011, CCIS 186*, pp: 231–240, 2011.
- [22] Haihua Chang and Xinhuai Tang, “A Load-Balance Based Resource-Scheduling Algorithm under Cloud Computing Environment”, *ICWL 2010 Workshops, LNCS 6537*, pp: 85–90, 2011.

- [23] Chao-Tung Yang, Kuan-Chieh Wang, Hsiang-Yao Cheng, Cheng-Ta Kuo and Ching-Hsien Hsu, "Implementation of a Green Power Management Algorithm for Virtual Machines on Cloud Computing", UIC 2011, LNCS 6905, pp: 280–294, 2011.
- [24] Sumeet S. Vernekar and Pravin Game, "Component Based Resource Allocation in Cloud Computing", Proceedings of the InConINDIA 2012, AISC 132, pp: 907–914, 2012.
- [25] Hao Li and Guo Tang, "Pareto-Based Optimal Scheduling on Cloud Resource", ICHCC 2011, CCIS 163, pp. 335–341, 2011.
- [26] Lei Xu, Wenzhi Chen, Zonghui Wang, Shuangquan Yang, "Smart-DRS : A Strategy of Dynamic Resource Scheduling in Cloud Data Center", IEEE International Conference on Cluster Computing Workshops, IEEE 978-0-7695-4844-9/12 (2012), pp: 120-127.
- [27] Shi Chen, Jie Wu, Zhihui Lu, "A Cloud Computing Resource Scheduling Policy Based on Genetic Algorithm with Multiple Fitness", IEEE 12th International Conference on Computer and Information Technology, IEEE 978-0-7695-4858-6/12 (2012), pp: 177-184.
- [28] D. Adami, C. Callegari, S. Giordano, M. Pagano, "A Hybrid Multidimensional Algorithm for Network-aware resource Scheduling in Clouds and Grids", IEEE ICC 2012-Communication QoS, Reliability and Modeling Symposium, IEEE 978-1-4577-2053-6/12 (2012), pp: 1297-1301.
- [29] Hao Li, Jianhui Liu, Guo Tang, "A Pricing Algorithm for Cloud Computing Resources", International Conference on Network Computing and Information Security, IEEE 978-0-7695-4355-0/11 (2011), pp: 69-73.
- [30] Xin Lu, Zilong Gu, "A LOAD-ADAPATIVE CLOUD RESOURCE SCHEDULING MODEL BASED ON ANT COLONY ALGORITHM", Proceedings of IEEE CCIS2011, IEEE 978-1-61284-204-2/11 (2011), pp: 296-300.

[31]<http://www.google.co.in/url?sa=t&rct=j&q=cloudsim&source=web&cd=10&cad=rja&sqi=2&ved=0CFEQFjAJ&url=http%3A%2F%2Farxiv.org%2Fpdf%2F0903.2525&ei=6ZbiUdfFDIjIrQecvYGoDw&usg=AFQjCNEApuhyrVNOe2i2Rta6zJo0j0USBA&bvm=bv.48705608,d.bmk> (0903.2525.pdf).

List of publications

[1] Nimisha Singla, Seema Bawa, “Review of Efficient Resource Scheduling Algorithms in Cloud Computing”, International Journal of Advanced Research in Computer Science and Software Engineering (communicated).

[2] Nimisha Singla, Seema Bawa, “Priority Scheduling Algorithm with Fault Tolerance in Cloud Computing”, International Journal of Advanced Research in Computer Science and Software Engineering (communicated).