Autonomic Fault Tolerant Scheduling for Multiple Workflows in Cloud Environment

A thesis submitted
in partial fulfillment of the requirements for the award of degree of
DOCTOR of PHILOSOPHY

by

Anju Bala
(950903014)

under the guidance of

Dr. Inderveer Chana
Associate Professor
Computer Science and Engineering Department
Thapar University, Patiala -147004

Computer Science and Engineering Department
Thapar University, Patiala - 147004, INDIA

January 2015

Dedicated to my family

Certificate

I hereby certify that the work which is being presented in this thesis titled, "**Autonomic Fault Tolerant Scheduling for Multiple Workflows in Cloud Environment**" in fulfillment of the requirements for the award of degree of "**Doctor of Philosophy**" 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.Inderveer Chana and refers other researchers works which are duly listed in the reference section.

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

(Anju Bala)
Regn. No. 950903014

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

08/06/15

(Dr.Inderveer Chana)
Associate Professor
Department of Computer Science & Engineering
Thapar University, Patiala, 147004
Punjab, INDIA

Acknowledgements

This PhD thesis is the outcome of an exigent journey, upon which the almighty has bestowed upon me wisdom and perseverance to accomplish this work successfully. I am eternally grateful to God for his everlasting blessings. I would like to express my sincere gratitude to my esteemed supervisor, Dr. Inderveer Chana, Associate Professor, Thapar University, Patiala for her systematic guidance as well as constant advice and encouragement throughout my PhD candidature. Her profound knowledge and expertise in this field and generosity have given me the audacity to explore new research directions and successfully complete this thesis. In particular, I appreciate her enthusiasm and endless amount of energy that inspired me to put maximum effort into research.

I am thankful to my Doctoral committee member Dr. Seema Bawa Professor, Computer Science and Engineering Department and committee member who directed me to select a best guide like Dr. Inderveer Chana. I wish to express my gratitude to Dr. Deepak Garg Head, Computer Science and Engineering Department, Thapar University, Patiala and Dr. Rajesh Khanna for being my Ph.D committee members and guiding me at the early stage of my research studies. I am gratefully acknowledge their valuable feedback while ensuring the progress of my research work.I would also like to thank Dr.O.P.Pandey, Dean of Research and Sponsored Projects for academic support and invaluable comments. My sincere thanks to Dr. Prakash Gopalan, Director, Thapar University for all the facilities that have been instrumental in creation of a healthy research environment in the university.

I also want to thank Dr. Maninder Singh, Associate Professor, Computer Science and Engineering Department, for productive and stimulating ideas that motivated me to explore the knowledge and current technologies for carrying out the experiments during this research process. I would also like to thank Dr. Jhilik Bhattacharya, Dr. Damandeep Kaur, Nidhi Jain, Taranpreet Kaur, for sharing

knowledge and participating in many fruitful discussions with me.

I wish to further extend my thanks to all the faculty members of the Computer Science and Engineering Department, Thapar University who helped me in one way or the other to carry out my work successfully. I also acknowledge the co-operation rendered to me by the office specially to Sh. Rakesh Kumar and the laboratory staff of this department.

This thesis would never have been contemplated without the unconditional support of my family. My hardworking parents deserve a special mention here. My father Sh. Tarsem Jain and Mother Mrs. Siksha Devi inculcated moral, ethical and religious values in me. I also want to thank my brothers, sister and friends. They made me smile and realize their faith in me during the rough road of this journey.

I would extend my deep sense of gratitude to my father-in law Sh. Prem Kumar Garg and mother-in law Smt.Shanti Goyal who extended their cooperation and encouragement to me. My warmest thank to my husband Dr. Pardeep Garg whose unconditional support during all these years is so appreciated. He inspired me in all dimensions of life and support me to successfully complete this journey. Finally, I want to mention the name of my five year child Durvish Garg and ten year child Divyanshu Garg, who have suffered a lot during this journey. I owe everything to them and this thesis would not have been complete without their everlasting support.

Anju Bala

Abstract

Cloud Computing is becoming an increasingly admired paradigm that owns the characteristics of existing paradigms through strong support for virtualization along with various additional features such as on demand resource provisioning, reduced cost, computing flexibility etc. Most of the scientific communities employ workflow technologies to cope up with the complexity and heterogeneity of large scale scientific applications. As, the scientific workflows need a suitable paradigm for deployment and execution in conjunction with high availability of Cloud services. Thus, Cloud is a current benchmark for effective facilitation of the execution of scientific workflows through flexibility of accessible services such as Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS) without allusion to the infrastructure on which these applications are hosted. For the successful execution of the scientific workflows on Clouds, Cloud platform should be able to manage the faults through autonomic fault tolerant approaches during the scheduling of workflow tasks on Cloud resources. Cloud providers also entail efficient scheduling algorithms to schedule these workflows along with autonomic fault tolerant approaches. Although, Cloud Computing technology has evolved but still some of the key challenges like autonomic fault tolerance and workflow scheduling need to be achieved.

To achieve the set of challenges for the fault tolerant workflow scheduling, a comprehensive study of workflow scheduling algorithms along with the required set of Quality of Service(QoS) parameters is carried out. In addition, Cloud platforms and workflow engines are extensively explored and metrics relevant to the Cloud services are ascertained. Furthermore, a thorough study of failure prediction approaches, fault tolerant techniques and fault tolerant scheduling has been performed. Based on the literature survey, it is evident that the key challenge in scheduling workflow applications on Cloud that needs to be addressed is fault tolerant scheduling in autonomic way. Hence, to address the assorted challenges, autonomic fault tolerant scheduling for multiple workflows is the main focus of this research work.

The proposed solution has been implemented in three stages. At first, intelligent failure prediction model has been proposed to predict the task failures for scientific applications. Multiple scientific workflow data has been collected, analysed and pre-processed for classifying as task failed or task not failed. Various machine learning approaches such as Naive Bayes, Artificial Neural Networks (ANN), Logistic Regression (LR) and Random Forest have been executed and compared underneath to predict the task failure. The performance metrics have been evaluated to compare the accuracy of the available machine learning approaches for predicted and actual failures. The comparative analysis based on the performance evaluation parameters has shown the effectiveness of Naive Bayes with maximum accuracy and minimum error in comparison to other implemented approaches. Thus, the proposed failure prediction model using Naive Bayes has been employed to predict the task failures intelligently before the occurrence of actual task failures.

Secondly, VM migration policy has been intended through proposed task failure model for implementing an autonomic fault tolerant technique. The proposed policy evaluates the correlation between resource utilization parameters such as RAM, CPU, Bandwidth and Disk I/O with allocated virtual machines and also finds the inter-correlation between VMs. Then, the VM which has maximum correlation and minimum inter-correlation factor is migrated to another host as to handle the task failures automatically. The proposed technique is further utilized for scheduling multiple workflow applications.

Finally, autonomic fault tolerant workflow scheduling algorithm has been proposed that firstly schedules multiple scientific workflows using earliest deadline first based approach, and then proposed hybrid heuristic schedules the tasks of the highest priority workflow. Hybrid heuristic combines the features of Max-Child, Min-Min and FCFS heuristic. To optimally utilize the resources, the proposed heuristic first schedules and executes that task which has maximum number of child nodes, if two or more of the tasks have same number of child nodes, then Min-Min heuristic would be used, again if the two tasks have same execution time, then FCFS has been considered. Furthermore, the proposed algorithm uses the autonomic VM

migration approach to migrate VMs to other working host before the occurrence of the actual task failure. Various scheduling factors have been defined to evaluate the performance of proposed scheduling algorithm and multiple scientific applications have also been detailed as workflows to schedule on Cloud resources using proposed approach.

The performance of the proposed intelligent failure prediction model has been evaluated on Pegasus and Amazon EC2. Pegasus bridges the scientific domain and the execution environment by automatically mapping high-level workflow descriptions onto Cloud resources and these workflows are further deployed on Amazon EC2. The experimental results have proved that the average accuracy for all the workflows using Naive Bayes is maximum (93%) among other approaches ANN, LR and Random Forest whereas existing model has the average accuracy of (85%). Thus, the proposed model is more effective with Naive Bayes approach for predicting task failures as well as resource failures by considering resource utilization for multiple scientific applications. Then, autonomic fault tolerant technique has been validated by reducing mean execution time, standard deviation of mean time, number of VM migrations and SLAV (Service Level Agreement Violations).

The proposed autonomic fault tolerant workflow scheduling technique has also been verified for multiple scientific workflows such as Montage, Cybershake, SIPHT, Epigenome and Inspiral. The proposed technique has been successful in reducing the average makespan for these scientific workflows. These results have been validated using the existing heuristics such as Particle Swarm Optmization (PSO) , Genetic Algorithm (GA) , Min-Min, Max-Min , MCT and proposed hybrid heuristics. It is verified for Epigenome workflow that the PSO performs better than Max-Min and MCT, whereas GA also enhances the performance over Max-Min, Min-Min, MCT, and PSO. Similarly, for Cybershake workflow, PSO performs better than Max-Min, Min-Min, MCT and GA. The proposed hybrid heuristic outperforms all the existing heuristics by reducing the average makespan for large-scale scientific workflows of 1000 jobs such as Cybershake and Epigenome.

List of Figures

List of Tables

Contents

Chapter 1

Introduction

Distributed computing interconnects the geographically distributed resources such as storage devices, data sources and compute resources utilized by users around the world as single amalgamated resource. Nowadays, a number of new concepts and terms related to distributed computing have surfaced that are promising to deliver IT as a service evolving from Cluster to Grid to Utility and now Cloud Computing. Cloud Computing owns the characteristics of existing paradigms through strong support for virtualization along with various additional features such as on demand resource provisioning, reduced cost, computing flexibility etc.

Cloud Computing is becoming an increasingly admired paradigm to deploy various scientific applications by reducing the overhead and cost of procuring infrastructure resources. The workflows describe the association of the individual computational components along with their input and output data in a declarative way and thus, these applications are described in the form of Cloud workflows. The flexibility of the Cloud services such as Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS) formulates a suitable platform for execution of today's workflow applications

This chapter presents an elevated view of the thesis. It introduces Cloud Computing, its evolution, deployment models and its layered architecture in conjunction with its various benefits. It also discusses the related technologies along with the basic workflow concepts. In addition, it offers the key obstacles of Cloud Computing that need to be addressed and need of Autonomic Computing to address these obstacles. The chapter concludes with the organization of the thesis along with its contributions.

1.1 Cloud Computing: An Overview

Computing is being renovated into a model consisting of services that are distributed akin to the conventional utilities like water, electricity, gas, and telephony etc. [1] [2]. Various computing paradigms have been assuring to deliver Utility based computing such as Cluster Computing, Grid Computing and currently Cloud Computing. Cloud Computing is an internet-based computing, whereby shared resources, software and information are provided to computers and other devices on-demand like a public utility. Cloud Computing services are offered by many of the Cloud providers such as Amazon, Yahoo, Google, Microsoft, Salesforce and Zoho etc. as shown in Figure 1.1. Cloud technology uses the internet and central remote servers to maintain data and applications. This technology allows consumers and businesses to use applications without installation and access their personal files at any computer , laptop, PDA, New iMac, Terminal and tablet etc. Today, the computing world is swiftly moving towards developing software using Cloud services with the required features like elasticity, virtualization, low cost, pay per usage etc.

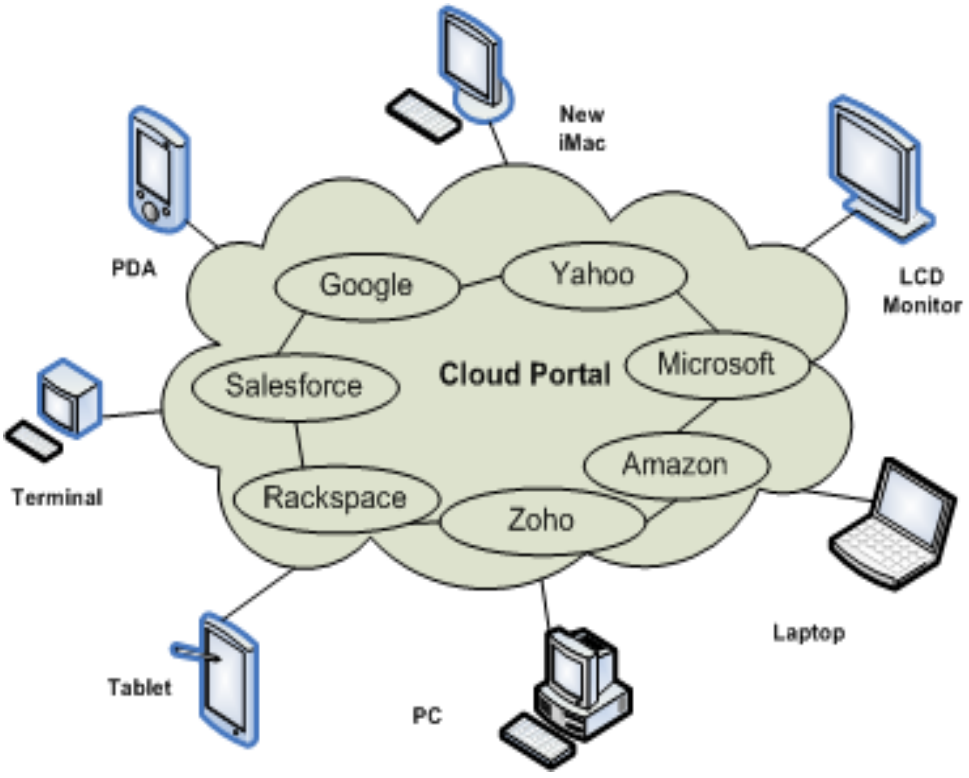


FIGURE 1.1: Cloud Computing Providers

As Cloud Computing has evolved from other technologies, therefore, the next section discusses its evolution and its emergence from Clusters, Grid and Utility Computing, its basic architecture and other features.

1.1.1 Cloud Evolution

Cloud Computing has evolved from various existing technologies such as Cluster Computing, Grid computing and Utility Computing as shown in Figure 1.2.

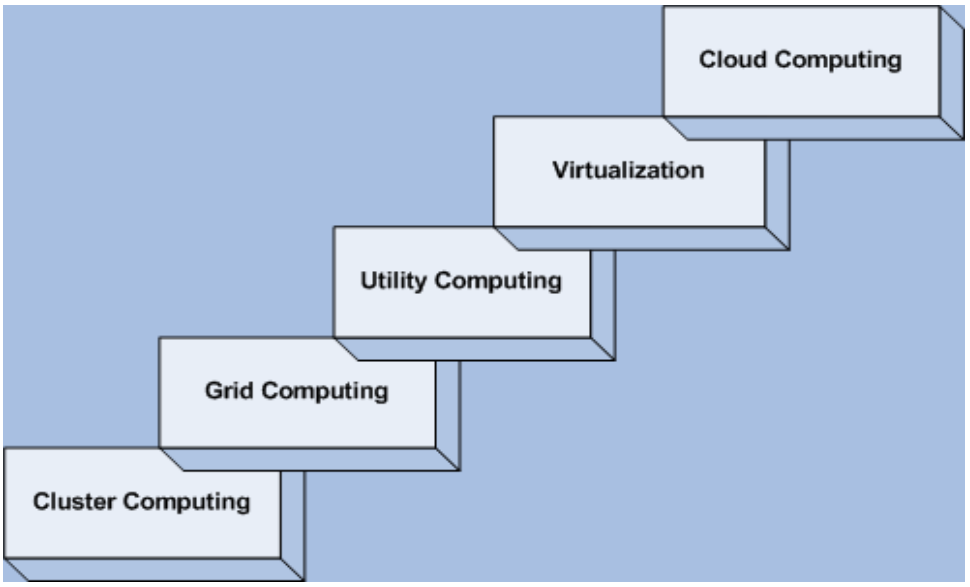


FIGURE 1.2: Evolution of Existing Technologies

Cluster is a collection of parallel or distributed computers [3], which are interconnected through high-speed networks, such as gigabit Ethernet, Infiniband etc. As per the resource requirements for high performance scientific applications, there was a need to extend a heterogeneous infrastructure that combined the resources from multiple organizations in the form of a Grid. Therefore, Grid Computing has evolved from the cluster that is a form of large scale distributed computing environment similar to the power grid [4][5]. It involves coordination and allocation of applications, data and storage or network resources across dynamic organisation, multi-institutional-virtual [6] and geographically dispersed organizations [7][8]. Moreover, to increase the performance of Grid, virtualization has been used to run multiple operating systems on a single physical system and share the underlying hardware resources [9] [10]. Further, Cloud Computing acquires one-step

further from Grid Computing by leveraging virtualization technologies at multiple levels so as to realize resource sharing and dynamic resource provisioning [11][12].

1.1.2 Cloud Architecture

Cloud Computing has transformed the Information and Communication Technology industry by facilitating on-demand services, elasticity, flexibility and provisioning of computing resources based on utility [13]. Cloud Computing has adopted virtualization technologies to provide various services to the user such as Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS)[14]. The layered architecture of the Cloud services, tools for these services along with their challenges have been depicted in Figure 1.3.

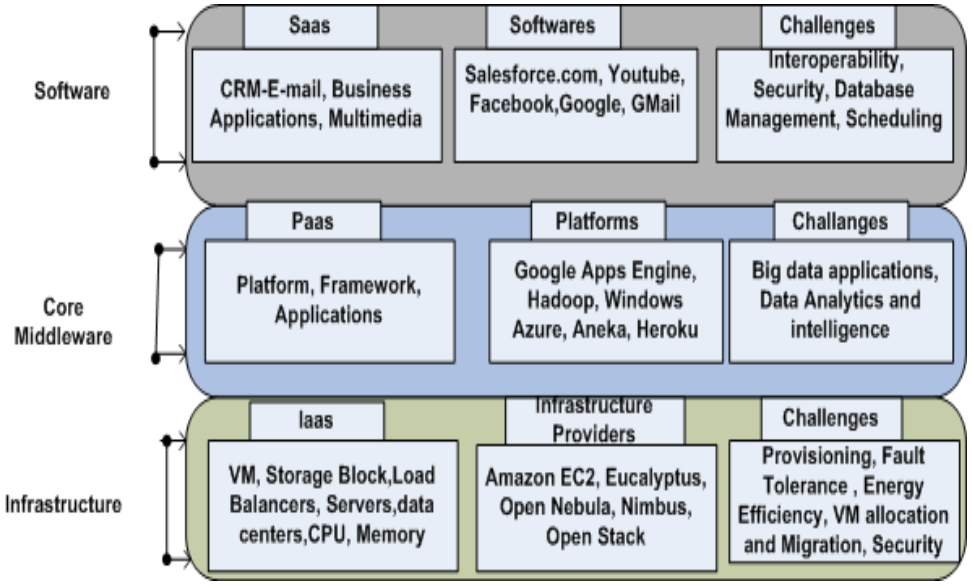


FIGURE 1.3: Layered Architecture of Cloud

- *Infrastructure as a Service(IaaS)*: The infrastructure layer at the bottom consists of IaaS that offers various hardware resources and infrastructures on demand without purchasing. The infrastructure layer services are used to match the power consumption according to demand, reduce cost by providing hardware resources and higher returns in terms of maximum resource utilization. Some of the key challenges on this layer include energy efficiency, virtual machine migration, resource provisioning, security, load balancing, fault tolerance and scheduling etc. Amazon EC2, Eucalyptus, OpenNebula, Nimbus and Open Stack, Rackspace etc. are the examples of infrastructure and hardware providers.

- *Platform as a Service(PaaS)*: The second layer of the architecture acts as a core middleware between software layer and infrastructure layer that presents a platform for testing, deploying and controlling web applications. The developer can focus only on the application code by adopting PaaS. The issues which need to be resolved at this layer are management of big data applications, data analytics and data intelligence etc. The examples of PaaS providers are: Google Apps Engine, Microsoft Azure, Hadoop and Aneka etc.

- *Software as a Service(SaaS)*: As the Cloud offers SaaS on the top layer of Cloud where the user can utilize the services on internet without buying the proprietary rights of software and applications such as scientific applications, E-mail, business and multimedia applications[15]. SaaS is being utilized for the elimination of licensing and version compatibility and reduced upfront cost and hardware cost. Some of the key challenges that need to be addressed at software layer are scheduling, data heterogeneity, fault tolerance, fault prediction, interoperability and security etc. Some established providers in this category are Salesforce.com , Google, Amazon.com etc. provides SaaS to consumers.

Cloud Computing services can be deployed using following Cloud models as depicted in Figure 1.4.

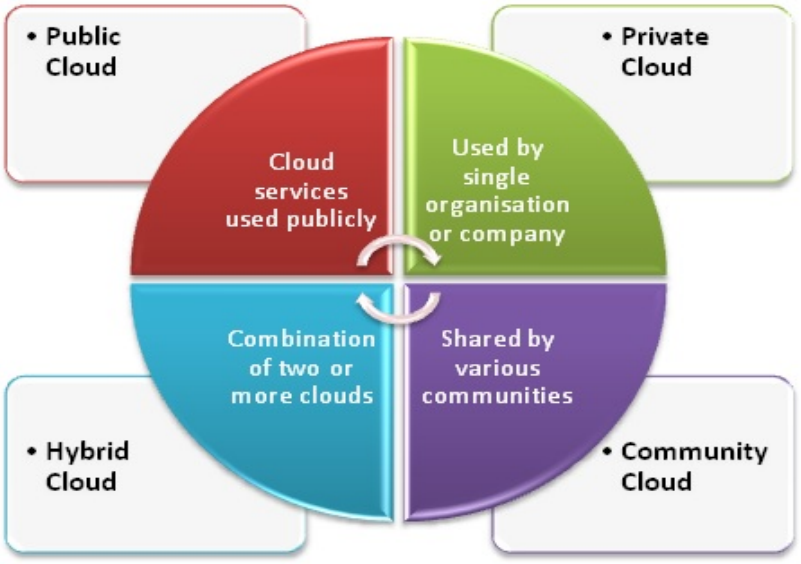


FIGURE 1.4: Cloud Deployment Models

- *Public Cloud*: In public Cloud, the infrastructure is made available to the general public. Within the infrastructure, the Cloud provider also provides other services to the Cloud users at free of cost. The advantages of adopting public Cloud are [16]:
  - No initial capital investment on infrastructure.
  - No data transfer risk.
- *Private Cloud*: The private Cloud is operated and managed by single private organization. Sometimes a third party apart from the organization manages the private Cloud in both on-premise and off-premise and it is also known as the internal Cloud. The benefits of private Cloud are:
  - To maximize and optimize the in-house resource utilization within the organization
  - Data privacy for the security concerns
  - Minimum data transfer cost
  - Reliability
- *Community Cloud*: Community Cloud is established by sharing the specific resources, infrastructure, requirements and policies of same Cloud with a community or a business sector. The community Cloud is different from public Clouds, which serves the different needs of multiple users and is also different from private Cloud, which serves within a single organization. Some of the key features of community Cloud are :
  - The Cloud infrastructure can be hosted by a third party organization
  - Economically scalable
- *Hybrid Cloud*: More than two Clouds such as public, private or community are combined together in the form of hybrid Cloud. Hybrid Cloud has used by several organizations to optimize the resources with security as the private Cloud and some of the organisations have utilized it for business purpose as a public Cloud also. The advantages of using hybrid Clouds are:
  - More flexible than public and private Cloud
  - Provide more security over data than public Cloud

  - Provide on demand service expansion and compaction by provisioning and releasing the resources whenever needed

As per the requirement of Cloud user, various types of Cloud models are discussed above, still there are certain features which are common to any type of Cloud and these features are procured in the next section.

1.1.3 Features of Cloud Computing

The following are key features of Cloud Computing [17][18]:

- *Open Access*: Cloud services are made available via internet. Any device having internet connection such as mobile phones, laptops, and tablet etc. can access these services.
- *Capacity for on-demand infrastructure and computational power*: Users can demand for the computational power, storage and other infrastructure according to their need as pay per use model.
- *Improved resource utilization*: Resources are utilized properly because whenever users don't need a resource they can return it back to the cloud provider. So, in this way elasticity and flexibility can be increased.
- *Reduced information technology (IT) infrastructure needs*: Cloud computing provides Infrastructure-as-a-Service on demand to the user. So, there is no permanent need to purchase the infrastructure for the IT. The user can purchase it from cloud provider whenever required.
- *Resource pooling*: The consumer generally has no information about the locality of the service provider. Thus, the provider serves multiple consumers by assigning resources dynamically and virtually.
- *Computing Flexibility*: It has more flexibility than other network computing systems and saves time and money for people who are in a time crunch and also useful for social networking [19].
- *Mobility*: It allows Cloud users to connect without their own computers and user can work from anywhere in the world as long as they have an internet connection.

- *Reliability*: Use of multiple redundant sites, makes Cloud Computing suitable for work continuity and disaster recovery.

After a concise introduction to Cloud Computing, the succeeding section confers related technologies such as Autonomic Computing and Workflows which have been utilized for this research work.

1.2 Related Technologies

Cloud computing requires reliable, scalable, secure, robust and high performance infrastructure for seamless delivery of cloud services. Autonomic Computing can considerably increase these inherent requirements of Cloud services, therefore, it is a desired technology for Cloud Computing [5][20]. Autonomic Computing also endows with a holistic approach for the design and development of large scale scientific applications that can adopted themselves to enhance the performance by satisfying the intrinsic requirements. Fault tolerance is the one of significant concept for increasing the performance of scientific applications that can be incorporated in autonomic computing through self-healing techniques and methods. Since the workflow technology has achieved more attention towards scientific applications that can be used to conduct scientific simulations in a parallel manner. Thus, these scientific applications are represented in the form of workflows. Therefore, this section discusses Autonomic Computing , Autonomic Fault Tolerance and Workflows.

1.2.1 Autonomic Computing

Autonomic Computing requires high-level guidance from humans and chooses the desired steps that need to be taken to keep the system stable [21] where the term "Auto" originated from biology [22]. To attain Autonomic Computing for Cloud, control loop reference model is utilized as shown in Figure 1.5 where monitor unit checks the status of Cloud nodes, analyze unit predicts the node failure, plan unit executes the recovery plans and execute block executes the jobs after automatic recovery from failure. An Autonomic Computing paradigm includes self-healing, self-configuration, self-protection and self-optimization as depicted in Figure 1.6. Self-healing enables to detect, analyze, and repair existing faults within the Cloud

environment without human intervention. Self-configuration automatically configures the Cloud components and self-protection detects and protects the security threats at run time for private, public and hybrid Clouds [23]. To ensure optimal functioning, self-optimization can monitor and control the Cloud resources automatically. Therefore, to increase the overall performance of existing Cloud technologies, Autonomic Computing paradigms could be used effectively. As this research work is directed towards fault tolerance, hence, the concept of self healing has been discussed to implement an autonomic fault tolerance technique.

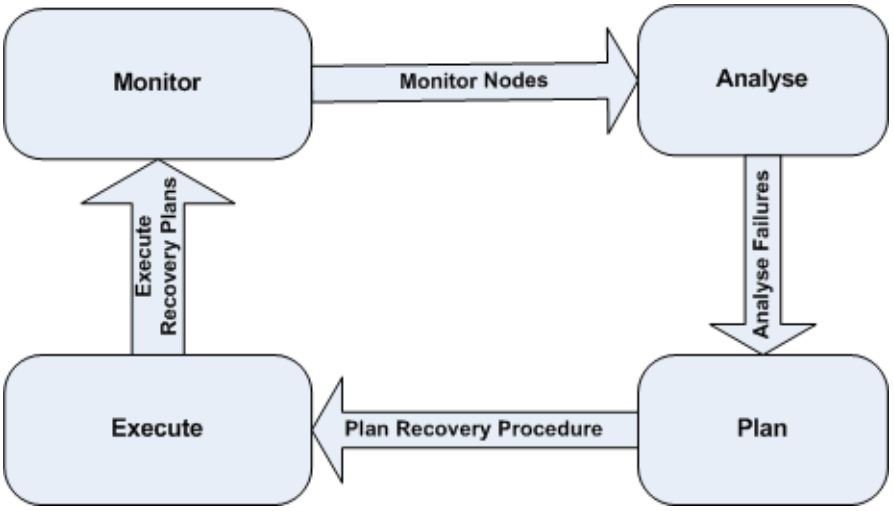


FIGURE 1.5: Control Loop Reference Model

(i) *Autonomic Fault Tolerance*: Autonomic Fault Tolerance in Cloud is the ability to handle the failures autonomically without human intervention. Self-healing concept of Autonomic Computing paradigm deals with fault tolerance for dynamic systems. These systems must have the knowledge and information about its expected behaviour in order to examine whether its actual performance deviates from its expected performance in relation to its environment [21]. As the research on self-healing systems has its commencement with fault-tolerant systems that can tolerate transient as well as permanent failures proactively during the deployment of high performance scientific applications on Cloud [24][25]. Thus, an autonomic adoption of self-healing techniques provide autonomic fault tolerant features that could be able to predict the failures and heal themselves using workflow concepts and terminology that has been discussed in next section.

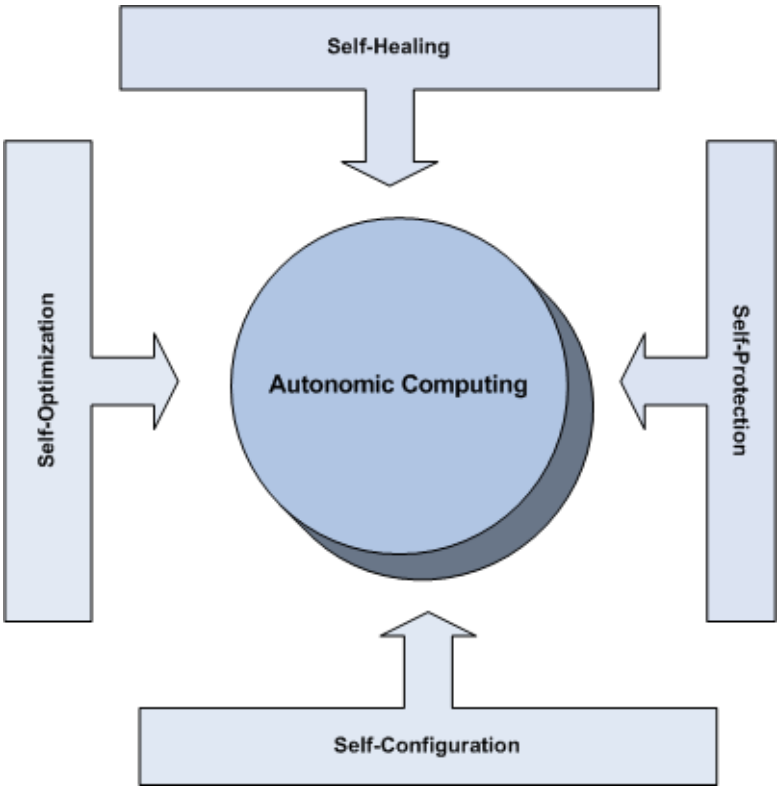


FIGURE 1.6: Categories of Autonomic Computing

1.2.2 Workflows: Concept and Terminology

The ever-growing demand and heterogeneity of Cloud Computing is garnering popularity with scientific communities to utilize the services of Cloud for executing large scale e-scientific applications in the form of set of tasks known as workflows. Workflows stipulate a process or computation to be executed in the form of data flow and task dependencies that allow users to simply articulate multi-step computational and complex tasks. Workflows are concerned with the automation of procedures where the documents, information or tasks are conceded between participants according to a defined set of rules and enables the structuring of various applications in a directed acyclic graph whereas each node represents the constituent task and edges represent inter task dependencies of the applications [26]. The workflows can be single or multiple whereas single workflows consists of single or multiple instances using same structure and multiple workflow considers multiple instances of different workflow structures. Workflow Management System (WMS) is appropriate for management of these workflows on the Cloud resources. The major components of WMS described in Figure 1.7 are Workflow Design, Information Retrieval, Workflow Scheduling and Fault Tolerance etc. are

detailed in subsequent sections [27].

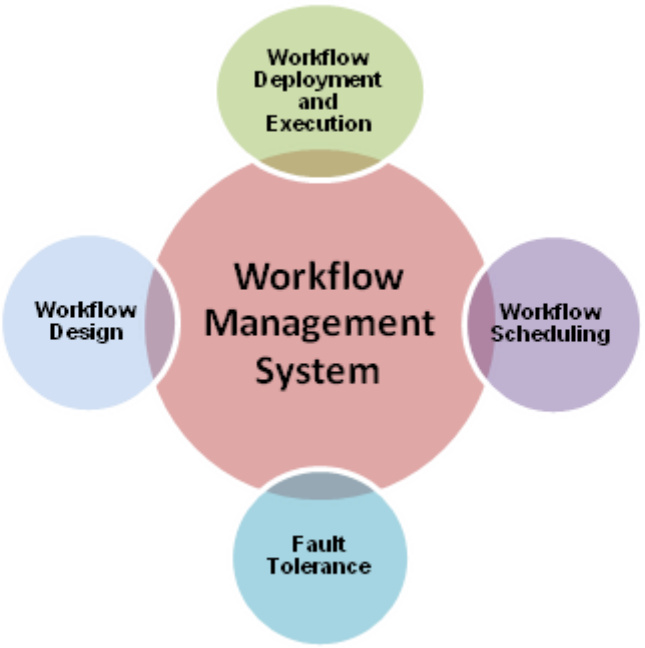


FIGURE 1.7: Components of Workflow Management System

(i) *Workflow Design*: Workflow design depicts how the components of workflow can be defined and composed as shown in Figure 1.8. Workflow Structure describes the relationship between the tasks of the workflow and can be categorized into two types Directed Acyclic Graph (DAG) and Non DAG [28]. DAG can further be classified as Sequence, Parallelism and Choice whereas Non DAG is being categorised as Iteration, Parallelism and Choice. In the sequence structure, the tasks accomplish in a sequence whereas in parallelism structure, tasks of workflow can execute concomitantly. In the choice structure, the workflow tasks can be executed either in sequence or parallel. Iteration pattern structure implements the tasks in iterative manner [29]. Another component of workflow design is workflow model that defines a workflow both at task and structure level into abstract and concrete workflows. Abstract workflows are represented as an abstract form without consigning to Cloud resources for task execution whereas Concrete workflows are referred to as executable workflows. Workflow composition allow the users to gather various components through user directed systems and autonomic systems. Cloud user can adapt the workflow structure directly through user directed systems and Cloud system spawns the workflow automatically by autonomic systems.

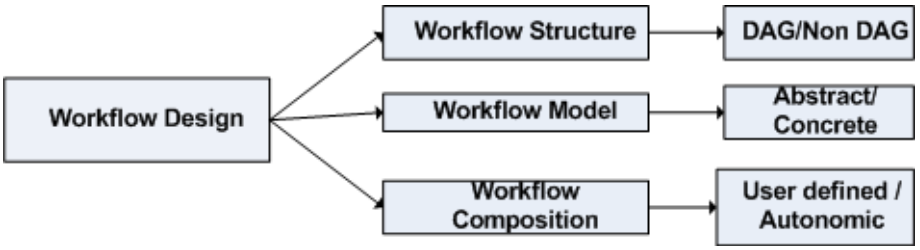


FIGURE 1.8: Components of Workflow Design

(ii) *Workflow Scheduling*: Workflow Scheduling is required to schedule the workflow task onto Cloud resources used for workflow execution. Components which need to be defined for workflow scheduling are discussed below are depicted in the Figure 1.9:

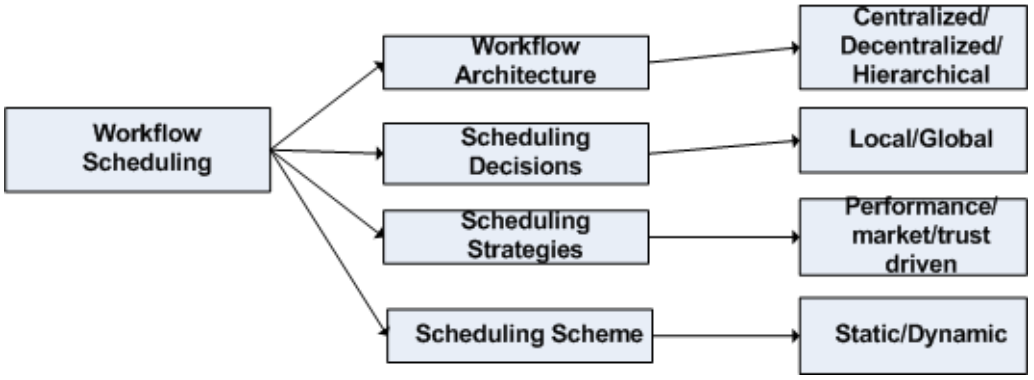


FIGURE 1.9: Components of Workflow Scheduling

- *Scheduling Architecture*: The scheduling architecture is further categorised into centralized, hierarchical and decentralized, where one centralized scheduler handles the tasks through centralized manner. There are sub workflow schedulers under central scheduler which can make decisions when central scheduler fails in hierarchical scheduling. There are multiple schedulers exclusive of central scheduler in decentralized scheduling that can further optimize the performance of workflow scheduling.

- *Scheduling Decisions*: The scheduling decisions about the allocation of resources to tasks can be local or global. If the decisions are relating to the task or sub workflow, they are known as local whereas in case of global the scheduling decisions are made by considering the entire workflow.

- *Scheduling Strategies*: Scheduling strategies are classified into performance driven, market driven and trust driven. Performance driven strategies can be utilized to achieve the optimal performance by reducing execution time whereas market driven strategies can be employed for managing resource allocation and trust driven schedulers can be used to select the resources based on the trust levels.

- *Scheduling Scheme*: Scheduling scheme is a method for mapping the abstract workflows to concrete workflows that can be static or dynamic. In case of static scheme, the concrete models are produced using current available information before the execution where in dynamic scheme; the concrete models are produced based on the static and dynamic information about the resources.

(iii) *Fault Tolerance in Workflows*: Fault tolerance is related to handling the failure which can occur during the scheduling and execution phase of workflows due to several reasons such as resource unavailability or resource failure , task failure, overloaded resources, network fault, and lack of memory etc. Fault tolerant techniques are classified into task level and workflow levels that can be utilized to tolerate the task as well as workflow failures. Task level fault tolerant techniques can handle task failures are retry, checkpointing, alternate resource, VM migration and replication etc. Workflow level fault tolerant techniques can be used to handle the workflow failures rather than the task or sub-workflow failures and some of these techniques are alternate task, redundancy, exceptional handling and rescue workflow etc. These fault tolerant techniques can be utilized for various scientific workflow applications during deployment and execution phase of workflows.

(iv) *Workflow Deployment and Execution*: Workflows have been used in scientific community for various applications such as Montage workflow for astronomical physics, Cybershake workflow for earthquake hazards , SIPHT workflow for bioinformatics, Inspiral workflows for detecting gravitational waves, Epigenome for genome sequence operations and Broadband to simulate the impact of an earthquake. The workflows have also been used for business community such as financial modelling, data center automation, data modelling, ATM processing etc. To deploy and execute these scientific and business workflow applications, various workflow tools such as Pegasus, Hamake, Oozie, YAWL can be utilized among existing Cloud platforms such

as Amazon EC2, Nimbus, Openstack, Opennebula etc.

This section emphasized the related technologies of Cloud Computing such as Autonomic Computing and Workflows. As these technologies are adopted by Cloud along with their features, still there are some of the research issues. Therefore, the following section illuminates the open challenges that recline in the way of successful espousal of the aforementioned technologies in Cloud Computing environment.

1.3 Cloud Computing: Open Challenges

There are several challenges related to the growth of Cloud Computing which are discussed below:

- *Reliability and Availability*: Reliability and Availability can be improved through the use of multiple redundant sites, which makes the Cloud Computing suitable for business continuity and disaster recovery. As high reliability and availability is expected from Cloud providers, hence, the service availability is also very important for consumers. So, to ensure high reliability and availability of all the services for multiple Cloud Computing providers and consumers, fault tolerant techniques can be used to handle and recover various faults [30][31]. Fault tolerant techniques can be further combined within scheduling algorithms so as to increase the efficiency [32]. But these fault tolerant techniques should be autonomic which can reduce the failure effect on Cloud resources automatically. The security and trust models can also be implemented to enhance the reliability cloud services[33].

- *Integration and Interoperability*: There is huge amount of sensitive data in an enterprise, which is unlikely to migrate to the cloud due to privacy and security issues [34][35]. As a result, there is a need to find out the research issues related to integration and interoperability between the software on premises and the services in the Cloud. If one company wants to transfer the data to another then there may be need to reformat data or change the logic in applications. Although industry standards do not exist for Cloud Computing for Application Programme Interfaces (APIs) or data import and export [36]. Thus, there is a need to implement the models which can resolve the portability issues.

- *Scalable Monitoring of System Components*: Scalability is the ability of an application to be scaled up so as to meet demand through replication and distribution of requests across a pool or farm of servers. Most of the previous works on web scalability has been reported through an implementation of static load balancing solutions within server Clusters. A dynamic scaling algorithm needs to be proposed that can be used further to handle sudden load surges, enhances resource utilization, reduces infrastructure, manages cost and energy efficiency. To address the high increase in the energy consumption from the perspective of the Cloud Computing, there is also a need to implement energy efficient algorithms and approaches along with dynamic scaling algorithms which can further reduce the power consumption [37] in a Cloud environment.

- *Big data Analytics and Intelligence*: To facilitate data discovery where data have been produced by different systems, an effort to standardize on data provenance representation is still underway. Hence, intelligent techniques, models and frameworks can be developed and implemented for visualizing and mining provenance data for high performance big data and scientific applications [27].

- *Virtual Machine Migration Policies*: Virtual machine (VM) migration can be implemented with the help of virtualization technology by balancing the load across the data centers. Since, there are various VM migration policies like Minimum Migration Time, Random Choice and Maximum Correlation Coefficient policy, still there is a need to implement efficient VM migration approaches that can further reduce the overall execution time during the migration process of virtual machines [38].

- *Failure Prediction Models*: As the Cloud environment has been used by many of the researchers to deploy multiple scientific applications through Workflow as a Service (WaaS), however, there are some of the key challenges such as task and resource failure prediction that need to be addressed. Self healing based machine learning techniques can be used for intelligent failure prediction and for efficient handling of task and resource failures autonomically [39].

- *Efficient Scheduling Heuristics and Resource Management*: As the Cloud provides various services to multiple users at the same time and different

users have different Quality of Service(QoS) requirements for multiple workflow applications, hence, the scheduling strategy needs to be implemented for multiple workflows having different structure along with different QoS requirements should be taken into account [40] [41]. As the multiple workflows were considered for concurrent execution of various scientific applications which are composed of hundreds and thousands of tasks during the execution [42]. For the execution of these workflow tasks in the Cloud environment, scientists need to use efficient heuristics along with scheduling strategy so as to schedule concurrent tasks and workflows onto the Cloud resources.

This section recapitulated the open issues in Cloud. The next section depicts the adoption of Autonomic Cloud concepts to resolve above key issues in the Cloud environment.

1.3.1 Adoption of Autonomic Cloud : Concepts and Obstacles

Autonomic Cloud has emerged as a result of espousing the paradigms of autonomic techniques for above discussed open challenges of Cloud Computing that are shown in Figure 1.10. In order to pinpoint the performance bottlenecks in Clouds for scientific applications [43], autonomic concepts can be used for resolving the key issues which are shown in Table 1.1.

Cloud Computing has on demand resource reservation that also supports both centralized and decentralized resource allocation and scheduling approaches but further needs to be optimized with autonomic resource allocation and scheduling techniques. As Cloud services could be accessed through public network [6][8], therefore security issues should be managed by self-protection techniques and trust models[34][44]. Cloud monitoring is not simple, so autonomic monitoring might be a upcoming challenge for Clouds that can be resolved through self-healing. Fault-Tolerance is also current research issue for Cloud which can further be handled proactively by autonomic monitoring and prediction. Although, the Cloud Systems have more scalability as compared to existing technologies yet they require to be dynamically scalable through autonomic concepts. Interoperability is also a serious issue for today's Cloud services which can be resolved by adopting self-configuration concepts [45]. As most of the companies are increasingly opting

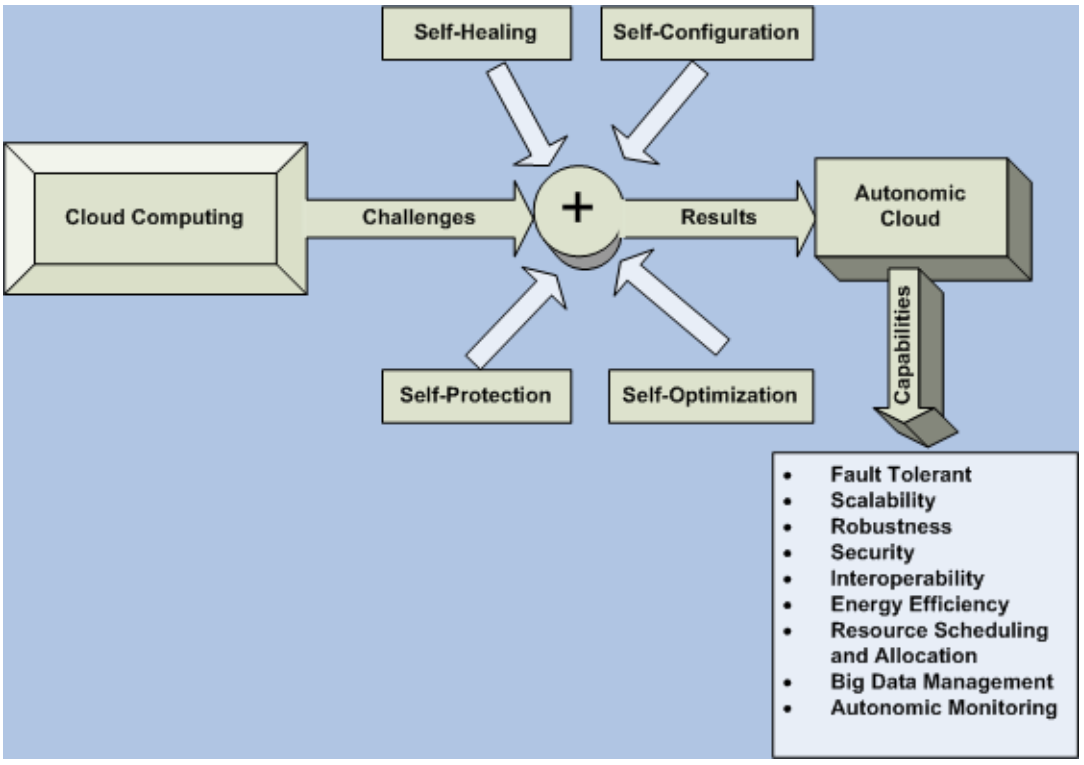


FIGURE 1.10: Adoption of Autonomic Cloud

Cloud environment to address big data problems, so self-optimization and self-healing techniques could be used for current big data management issues. Cloud addresses energy conservation by applying different energy efficiency techniques but these techniques should be autonomic that would dynamically increase the energy efficiency by optimizing through self-protection and self-healing. To enhance the scheduling success rate of various large scale scientific and business applications, self-healing and self-protection techniques can be integrated along with scheduling algorithms [46]. Henceforth, existing Cloud Computing systems could be autonomic to handle the challenges of fault tolerance, security, interoperability, scalability, big-data, energy consumption and workflow scheduling etc. [47]

This section summarizes the leading issues by adopting Cloud services which can be resolved by espousing various autonomic Cloud features, thus, the following section illustrates autonomic fault tolerant scheduling in the Cloud environment, which has been selected as the research area for this thesis.

Table 1.1: Realization of Cloud Computing Issues through Autonomic Concepts

Current Research Issues	**Adoption of Autonomic Concepts**
Resource Allocation and Scheduling	Self-Optimization, Self-Configuration
Security	Self-Protection
Autonomic Monitoring	Self-Healing
Fault Tolerance	Self-Healing
Scalability	Self-Configuration
Interoperability	Self-Configuration
Power and Storage Capacity	Self-Optimization
Big Data	Self-Optimization , Self-Healing
Energy Consumption	Self-Optimization, Self-Healing
Workflow Scheduling	Self-Protection, Self-Healing

1.4 Autonomic Fault Tolerant Scheduling : Research Motivation

Most of the large-scale scientific applications executed on present-day Clouds are expressed as workflows that stipulate a process or computation to be executed in the form of data flow and task dependencies. For the successful execution of the scientific applications on Clouds, Cloud platform should be able to handle the faults autonomically through fault tolerant approaches for the scheduling of workflow tasks on Cloud resources. As multiple scientific applications can have different workflow structures, thus, there is an inherent need to implement and design an autonomic fault tolerant approach that handles and predicts failure of the resources and tasks proactively for multiple workflows. Cloud providers also entail efficient scheduling algorithms to schedule these workflows along with autonomic fault tolerant approaches. Hence, development of an effective scheduling algorithm along with an autonomic fault tolerant technique is a challenging task for different workflow applications in Cloud environment. The other issues correlated with the scheduling and fault tolerance are summarized below:

- Fault tolerant technique must be capable to predict the task failures proactively for scientific applications by adopting autonomic techniques and should be able to tolerate the faults proactively.
- Fault Tolerant scheduling must persist efficient VM migration approach that can migrate the faulty VMs autonomically to other working hosts.
- Fault tolerant scheduling must use efficient scheduling algorithms so as to

schedule multiple scientific workflows onto cloud resources. Henceforth, efficient scheduling heuristics should be assured for implementing autonomic fault tolerant scheduling.

- Fault tolerance must be further optimized by minimizing total execution mean time, standard deviation time and makespan during scheduling of scientific workflow applications in Cloud environment.

Due to the factors sketched above, Autonomic Fault Tolerant Scheduling has been the motivation of this thesis and the following section presents the contribution of the thesis.

1.5 Thesis Contributions

This thesis contributes in the following ways:

- The state-of-art approaches for workflow design and scheduling have been extensively explored and compared through evaluated metrics for Cloud Computing along with Cloud platforms and workflow engines.
- The detailed literature review on failure detection and prediction approaches in Cloud Computing has been done along with research challenges and its solutions. Existing fault tolerant policies, autonomic techniques and scheduling have also been discussed along with their future directions.
- Intelligent failure prediction model for multiple scientific workflow applications has been proposed using machine learning approaches such as Nave Bayes, Artificial Neural Networks, Logistic Regression and Random Forest. The proposed model has been validated and compared through performance metrics. The approach which has maximum accuracy and minimum error is being utilized for predicting failure tasks.
- An autonomic fault tolerant technique has been proposed that migrates the identified VM proactively after predicting task failures using proposed failure prediction model.
- Workflow Scheduling algorithm has also been proposed that schedules multiple scientific workflow applications such as Montage, Cybershake, SIPHT, Broadband, Epigenome and Inspiral based on Earliest Deadline First and

proposed hybrid heuristic. The proposed autonomic fault tolerant approach also handles task failures during multiple workflow execution and scheduling of concurrent workflows to Cloud resources.

- The experimental results demonstrate the effectiveness of the proposed approach in terms of makespan, execution time mean and standard deviation of execution time.

The following section details the organisation of the thesis.

1.6 Thesis Organisation

This thesis is organised into following six chapters:

Chapter 1 *Introduction*: The introduction to the thesis presented in Chapter 1 is partially obtained from:

- Anju Bala, Inderveer Chana (2015). Autonomic Cloud Computing: Existing Techniques and Future Directions, Concurrency and Computation: Practice and Experience, Wiley, Impact factor: 0.784, **(SCI indexed)**- *Communicated.*

The rest of the thesis is organised as follows:

Chapter 2 *Literature Review*: This chapter details the literature survey on the Workflow Scheduling Algorithms on cloud infrastructures, various evaluation metrics, existing tools and Cloud platforms have been analysed for implementing workflow scheduling in Cloud environment. Then, existing failure prediction , fault tolerance techniques have also been surveyed along with their autonomic behaviour. Furthermore, autonomic fault tolerant scheduling approaches have also been compared to analyse the problem of research work. This chapter finally ends up with the Problem Formulation that derives from:

- Anju Bala, Inderveer Chana (2012). Fault Tolerance-Challenges, Techniques and Implementation in Cloud Computing, International Journal of Computer Science Issues, Volume 9, Issue 1, pp. 288-293, 2012, Impact Factor:0.242, Citations: 21.

- Anju Bala, Inderveer Chana (2012). Design and Deployment of Workflows in Cloud Environment.International Journal of Computer Applications, Volume 51, Issue 11, pp.9-15, 2012, Impact Factor:0.8, Citations: 02.

- Anju Bala, Inderveer Chana (2011). A Survey of Various Workflow Scheduling Algorithms in Cloud Environment.International Journal of Computer Applications Proceedings on 2nd National Conference on Information and Communication Technology, Volume NCICT4, pp. 26-30, 2011, Impact Factor:0.8, Citations: 31.

Chapter 3 *Proposed Autonomic Fault Tolerant Approach*: This chapter confers an Intelligent failure prediction model using machine learning approaches. Various evaluated metrics have also been defined to validate and compare the accuracy of machine learning approaches to implement an intelligent failure prediction model. The model which has maximum accuracy is utilized to implement Autonomic Fault Tolerant technique through proposed VM migration policy. The proposed technique migrates the faulty VMs to another hosts where the task failures have been predicted. This chapter derives from the following papers:

- Anju Bala, Inderveer Chana (2015). Intelligent Failure Prediction Models for Scientific Workflows, Expert Systems with Applications, Elsevier Publications **(SCI)**, Volume 42, Issue 3, pp. 980-989, 2015, DOI: 10.1016/ j.eswa.2014.09.014, Impact Factor: 1.965.

- Anju Bala, Inderveer Chana. (2013). VM Migration Approach for Autonomic Fault Tolerance in Cloud Computing. International Conference of Grid and Cloud Applications (GCA13), USA, Las Vegas, pp. 3-10, 2013, Citations:04.

- Anju Bala, Inderveer Chana (2015). Prediction Based Proactive Fault Tolerant Approach , Computing and Informatics, Impact Factor:0.319, **(SCI indexed)**- *Communicated.*

Chapter 4 *Autonomic Fault Tolerant Scheduling Approach for Multiple Workflows*: This chapter illustrates the proposed workflow scheduling heuristic for multiple scientific workflow applications. Firstly, deadline based scheduling has been used for prioritizing the workflows having shortest deadline first. The workflow having maximum priority among multiple workflows is scheduled first on the Cloud resources and other workflows have also been scheduled concurrently on available resources. For workflow execution and scheduling, the hybrid heuristic has been proposed that combines the features of Max-Child, Min-Min and FCFS heuristic. The proposed autonomic fault tolerant technique has also been utilized in the workflow scheduling algorithm to handle task failures due to the resource overutilization. Various scheduling factors have been defined to evaluate the performance of proposed scheduling algorithm and multiple scientific applications have also been detailed to schedule on Cloud resources using the proposed approach. The content of the chapter are acquired from:

- Anju Bala, Inderveer Chana (2015). Autonomic Fault Tolerant Scheduling Approach for Scientific Workflows in Cloud Computing, Concurrent Engineering: Research and Applications **(SCI)**, Sage Publications, pp 1-13, 2015, DOI: 10.1177/1063293X14567783, Impact Factor: 0.531.

- Anju Bala, Inderveer Chana (2015). Multi-level Priority Based Task Scheduling Algorithm for Workflows in Cloud Environment, ICECECE 2015 : International Conference on Electrical, Computer, Electronics and Communication Engineering, New Delhi, India, February, 7-8, 2015-*(Accepted).*

Chapter 5 *Experimental Results and Discussion*: This chapter details the experimental results by gathering task failure data after running scientific workflow applications in WorkflowSim. Then, actual task failures have been compared with predicted task failures using Pegasus and Amazon EC2 Cloud. Further, autonomic fault tolerant approach has been implemented using CloudSim by predicting host failures. The performance has been measured and compared using evaluated metrics. Further, the chapter demonstrates the proposed autonomic fault tolerant

scheduling results obtained after deploying multiple scientific workflow applications in Cloud environment. It also discusses the experimental results obtained after implementing proposed scheduling heuristics and VM migration approach in cloud environment. The content of the paper acquires from:

- Anju Bala, Inderveer Chana (2015). Intelligent Failure Prediction Models for Scientific Workflows, Expert Systems with Applications **(SCI Indexed)**, Elsevier Publications, Volume 42, Issue 3, pp. 980-989, 2015, DOI: 10.1016/j.eswa.2014.09.014, Impact Factor: 1.965.

- Anju Bala,Inderveer Chana. (2013). VM Migration Approach for Autonomic Fault Tolerance in Cloud Computing, International Conference of Grid and Cloud Applications GCA13, USA, Las Vegas, pp. 3-10, 2013, Citations: 04.

- Anju Bala, Inderveer Chana (2015). Autonomic Fault Tolerant Scheduling Approach for Scientific Workflows in Cloud Computing, Concurrent Engineering: Research and Applications (**SCI Indexed)**, Sage Publications, pp 1-13, 2015, doi: 10.1177/1063293X14567783, Impact Factor: 0.531.

- Anju Bala, Inderveer Chana (2015). Prediction Based Proactive Fault Tolerant Approach , Computing and Informatics, Impact Factor:0.319, **(SCI indexed)**- *Communicated.*

- Anju Bala, Inderveer Chana (2015). Prediction Based Proactive Load Balancing Approach through VM migration, Knowledge and Information Systems, Springer, Impact Factor:2.67,**(SCI indexed)**- *Communicated.*

Chapter 6 *Conclusion and Future directions*: This chapter finally concludes the thesis and considers the future scope of the work.

Chapter 2

Literature Survey

Cloud Computing is a new standard towards enterprise application development that can efficiently aid the execution of workflows for business as well as scientific applications by rewarding the requirements of modern enterprises. As the workflow applications often require very complex execution environments that are difficult to create and sustain. Therefore, Cloud infrastructures are being evaluated as an execution platform for deploying scientific workflows due to the features such as scalability, provisioning on demand, elasticity, provenance and reproducibility etc.

Clouds being an outgrowth of previous distributed systems require novelty in workflow scheduling capabilities. For the deployment of the workflow applications in Cloud environment, the workflow tasks need to be efficiently mapped on Cloud resources using workflow scheduling approaches. Therefore, the existing literature has been reviewed in order to accomplish the workflow scheduling in resourceful and fault tolerant manner. As the effectiveness of the workflow scheduling directly relates to fault tolerant techniques , henceforth, the survey of fault tolerant techniques is also essential for performing efficient and reliable workflow scheduling.

Thus, this chapter conducts an extensive analysis of workflow scheduling algorithms along with workflow design approaches, workflow engines and Cloud platforms. Further, the state of the art of failure prediction and fault tolerant policies along with scheduling has also been discussed to find out the gaps. Finally, the concluding section summarizes the gaps of the existing approaches and sketches the objectives of the thesis.

2.1 Workflow Scheduling : State of the art

Cloud technologies have an advancement towards service-oriented paradigm that facilitate scientific applications to acquire benefit of Cloud resources through the internet. Scientific applications have a need of workflow processing where the workflow tasks are executed according to their task and data dependencies. As the different scientific workflow applications can have different structure, therefore, multiple workflows can be managed concurrently on various Cloud resources. To schedule multiple workflows on the Cloud environment, workflow design is required prior to workflow scheduling. Therefore, this section discussed the workflow design approaches , existing workflow scheduling algorithms along with evaluated QoS parameters, deployment tools and Cloud platforms.

2.1.1 Workflow Design: Concepts and Approaches

Workflow design depicts how the components of workflow can be defined and composed which is the main component of the workflow scheduling, henceforth, various workflow design approaches have been explored as shown in Table 2.1. Piotr Chrz et al. [48] proposed Top-Down Petri Net-Based Approach that aims to increase the soundness of resulting workflow nets and uses various powerful refinement rules. Irene Vanderfeestenm et al. [49] also presented a product-based approach for directly executing the product-based model. Josefina Guerrero Garca et al. [50] demonstrated a model-driven approach to derive user interfaces of a workflow information system from a series of models. A graphical editor has also developed. Klein et al. [51] described a novel knowledge-based approach for helping workflow process designers and participants that could better manage the exceptions. Lin Chen et al. [52] discussed a workflow design tool used for combining graphical process representation and ECA rules in controlling Grid workflow process. Integration adapter of the Grid Workflow system has also been presented to facilitate the composition of all possible services. Macro Comuzzi [53] presented service based approach to design monitoring process. Therani Madhusudan et al. [54] discussed an Agent-based Process Coordination (APC) framework for distributed design process management. This approach embedded autonomous agents in a workflow-based distributed systems infrastructure. Hence, most of the approaches have been designed in Distributed and Grid environment. These approaches can

be designed in Cloud environment using languages like JAVA, XML, WSDL or Petri nets etc.

After designing a workflow, the workflow can be scheduled using workflow scheduling algorithms and analysed using workflow engines. For efficient scheduling, existing workflow scheduling approaches and algorithms need to be explored. Thus, the next part of this section presents a state of the art of workflow scheduling algorithms.

TABLE 2.1: Existing Workflow Design Approaches

Approach	Features	Language	Future Work	Environment
Top-Down Petri Net-Based Approach	Increases the soundness of resulting workflow nets	Petri Nets Based JAVA	Deadlock risks can be reduced	Distributed
Product Based Approach	Enhances flexibility and dynamicity	Petri nets	Optimal strategies can be used	Distributed
Model-Driven Approach	Utilizes for designing user interface	XML	Workflow Analysis Tools can be used	Cloud
Knowledge Based Approach	Manages the exceptions	-	Needs to be implemented	Distributed
ECA Rule Based Design Approach	Solves the complicated business logic or scientific application problems	WSDL	Artificial intelligence techniques need to be merged	Grid Environment
Service based Approach	Designs the monitoring process	Prom	Needs to be implemented	Distributed
Agent Based Approach	Manages the knowledge-intensive workflows	JAVA	Needs to be combined with other approaches	Distributed

2.1.2 Workflow Scheduling Approaches

Workflow Scheduling is the method of mapping the tasks to the Cloud resources and can be implemented either by best-effort based or QoS based constraints. The best-effort based scheduling minimizes the execution time while ignoring other factors such as cost of Cloud resources whereas, QoS constraint based scheduling reduces the performance due to the most important QoS constraints such as time minimization for budget restrictions and cost minimization for deadline limits. An efficient scheduling can have important impact on the performance of the Cloud system, thus, various workflow heuristics and scheduling algorithms for scientific workflows have been surveyed out which are discussed below:-

- *Optimized Resource Scheduling*: Zhong et al. [55] implemented an optimized scheduling algorithm to accomplish the optimization or sub-optimization for task scheduling. In this algorithm, an Improved Genetic Algorithm (IGA) has been utilized for the automated scheduling policy which is further used to increase the utilization rate of resources and speed.

- *Transaction Intensive Cost constraint*: Yang et al. [56] presented a scheduling algorithm which considered the cost and time and the simulation has demonstrated that this algorithm reduces the cost while meeting deadline. Selvarani et al. [57] proposed an improved cost-based scheduling algorithm for making efficient mapping of tasks to available resources in Cloud. This scheduling algorithm evaluated both resource cost and computation performance which enhanced the computation/communication ratio. Further, Liu et al. [58] presented a novel compromised-time cost scheduling algorithm which considered the characteristics of Cloud Computing to accommodate instance-intensive cost constrained workflows by compromising the execution time and cost with user input enabled on the fly.

- *Particle Swarm Optimization-based Heuristic for Scheduling Workflow Applications*: Pandey et al. [59] presented a Particle Swarm Optimization (PSO) based heuristic to schedule the applications to Cloud resources that takes into account both computation cost and data transmission cost. It is used for workflow application by varying its computation and communication costs. The experimental results have shown that PSO can achieve cost savings and good distribution of workload onto the Cloud resources. Further they have also extended PSO for deadline based resource provisioning and scheduling [60]. But, they did not describe resource failure and maximum dependency of the parent tasks.

- *Market-Oriented-Hierarchical Scheduling*: Wu et al.[61] proposed a market-oriented hierarchical scheduling strategy that considered service-level scheduling and a task-level scheduling where the service-level scheduling pacts with the Task-to-Service assignment. The task-level scheduling deals with the optimization of the Task-to-VM assignment in the local Cloud data centers.

- *Scalable-Heterogeneous-Earliest-Finish-Time Algorithm (SHEFT)*: Zhao et al. [62] also considered the performance of (HEFT) algorithm by opting different approximation methods. The experimental results described that

the mean value method is not a better choice because the performance could diverge from one application to another. Lin et al. [63] proposed a SHEFT workflow scheduling algorithm to schedule a workflow elastically on a Cloud Computing environment. The experimental results have shown that SHEFT not only outperforms several representative workflow scheduling algorithms in optimizing workflow execution time, but also enables resources to scale elastically at runtime.

- *Multiple QoS Constrained Scheduling Strategy of Multi-Workflows (MQMW)*: Xu et al. [40] have implemented a strategy for multiple workflows with multiple QoS. They have increased scheduling access rate and also minimized the make span and cost of workflows for Cloud Computing platform.

- *Greedy Randomized Adaptive Search Procedure (GRASP)*: A Greedy Randomized Adaptive Search Procedure (GRASP) is an iterative randomized search technique. Feo and Resende [64] proposed guidelines for developing heuristics to handle various combinatorial optimization problems based on the GRASP concept. Binato et al. [65] have shown that the GRASP can solve job-shop scheduling problems successfully. Recently, the GRASP has been investigated by Blythe et al. [66] for workflow scheduling on Grids by comparing with the Min-Min heuristic for both compute and data-intensive applications but it does not consider the scalability.

- *Genetic Algorithm (GA)*: Spooner et al. [67] have worked on Genetic Algorithm (GA) to schedule sub-workflows in a heterogeneous environment. Futher, Wang et al. [64] have extended GA to optimize the reliability and makespan, but they have not used any fault tolerant technique along with scheduling heuristic. Prodan and Fahringer [68] have engaged GAs to schedule WIEN2k workflow on Grid environment.

- *Optimal Workflow based Scheduling (OWS)*: Varalakshmi et al. [69] proposed OWS algorithm by finding a solution that meets all user preferred QoS constraints along with significant improvement of CPU utilization.

- *Resource-Aware-Scheduling algorithm (RASA)*: Saeed Parsa and Reza Entezari-Maleki [70] proposed a new task scheduling algorithm RASA that is composed of two traditional scheduling algorithms; Max-min and Min-min. The experimental results have shown that RASA outperforms the existing scheduling algorithms in large scale distributed systems.

- *Simulated Annealing (SA)*: Simulated Annealing (SA) [71] derives from the Monte Carlo method for statistically searching the global. The concept is originally developed from the way in which crystalline structures can be formed into a more ordered state by using annealing process that repeats the heating and slowly cooling a structure. For the Grid environment, YarKhan and Dongarra [72] have used SA to select a suitable size of a set of machines for scheduling a ScaLAPACK application whereas Young et al. [73] have investigated performances of SA algorithms for scheduling workflow applications. Further, Abdullah et al. [74] have proposed SA for task scheduling in Cloud environment with the aim of reducing the execution time, but they have not analysed any failures.

- *Workflow Partitioning*: Chen et al. [75] partitioned large-scale scientific workflows in addition with resource provisioning to reduce the workflow makespan and resource cost.

Table 2.2 compares the performance of existing workflow scheduling algorithms in Cloud and Grid environments. These algorithms have been analysed based on scheduling methods, parameters, factors, features, environment and tools. The tabular analysis demonstrates that there are very few algorithms that schedule multiple workflows in Cloud environment and none of the discussed algorithms have optimized both for fault tolerance and scheduling in the Cloud. Furthermore, various evaluation metrics have also been defined so as to summarize the QoS parameters.

(i) *Evaluation Metrics for Workflow Scheduling in Cloud Computing*: Table 2.3 analyses evaluation metrics for the above discussed workflow scheduling approaches. The existing workflow scheduling algorithms consider various parameters like time, cost, makespan, speed, scalability, throughput, resource utilization, scheduling success rate and so on. Reliability and availability is not being considered by any of the authors in the discussed papers. To deploy and execute these workflow applications, a variety of workflow engines and deployment platforms have been used by many authors and are defined in the next section.

TABLE 2.2: Existing Workflow Scheduling Algorithms

Algorithm	Methods	Parameters	Factors	Findings	Environment	Tools
Optimized-Resource Scheduling [T1]	Multiple instances	Speed, Resource Utilization	Request allocation problem	Increases Speed of IGA	Cloud	Eucalyptus
Transaction Intensive Cost-Constraint Scheduling[T2]	Batch Mode	Execution cost and time	Workflow with large number of instances	Minimizes the cost ,Improves the computation/communication ratio	Cloud	SwinDeW-C, CloudSim
A Particle Swarm Optimization-based Heuristic [T3]	Dependency mode	Resource utilization, time	Group of tasks	Utilizes for three times cost savings as compared to BRS	Cloud	Amazon EC2, CloudSim
Market-oriented Hierarchical Scheduling Strategy [T4]	Virtual clusters	Make span, Cost, CPU time	Service level scheduling ,Task level scheduling	Optimizes both make span and cost simultaneously	Cloud	SwinDeW-C
SHEFT workflow scheduling algorithm [T5]	Dependency Mode	Execution time, scalability	Group of tasks	Optimizes workflow execution time,Enables resources to scale elastically during workflow execution	Cloud	CloudSim.
Multiple QoS Constrained Scheduling Strategy of Multi-Workflows [T6]	Batch or dependency	Scheduling success rate, Cost, Time, Make span	Multiple Workflows	Schedules the workflow dynamically, Minimizes execution time and cost	Cloud	CloudSim.
GRASP[T7]	Batch or dependency	Speed, cost	Data Intensive Workflows	Compares the performances of Min-Min heuristics for workflow applications	Grid	GridSim, NS2
GA[T8]	Dependency mode	Reliability,Makespan	Single workflows	Schedules and maps on heterogenous environment	Grid and Cloud	GridSim, CloudSim.
Optimal Workflow based Scheduling (OWS) algorithm [T9]	Virtual clusters	CPU utilization, Execution time	Multiple Workflows	Finds a solution that meets all user preferred QoS constraints, Improves CPU utilization	Cloud	Open nebula.
RASA Workflow scheduling [T10]	Batch mode	make span	Grouped tasks	Used to reduce make span	Grid	GridSim.
SA[T11]	Virtual clusters	Execution Time	Single	Minimizes execution time	Grid,Cloud	GridSim, CloudSim.
Workflow Partitioning [T12]	Large Scale Workflows	Makespan,cost	Multiple Workflows	Integrates resource provisioning and workflow scheduling	Cloud	Pegasus

TABLE 2.3: Metrics considered by existing Workflow Scheduling Algorithms

Techniques	Time	Cost	Scalability	Scheduling Success Rate	Makespan	Speed	Resource Utilization	Reliability	Availability
T[1]	X	X	X	X	X	√	√	X	X
T[2]	√	√	X	X	X	X	X	X	X
T[3]	√	√	X	X	X	X	X	X	X
T[4]	√	√	X	X	√	X	X	X	X
T[5]	√	X	√	X	X	X	X	X	X
T[6]	√	√	X	√	√	X	X	X	X
T[7]	√	√	X	X	√	X	X	X	X
T[8]	X	X	X	X	√	X	X	X	X
T[9]	√	X	X	X	X	X	√	X	X
T[10]	X	X	X	X	√	X	X	X	X
T[11]	√	X	X	X	X	X	X	X	X
T[12]	X	√	X	X	√	X	X	X	X

2.1.3 Existing Workflow Execution and Deployment Tools

Workflows have been used in scientific community for various applications such as Montage workflow for astronomical physics, Cybershake workflow for earthquake hazards , Sipht workflow for bioinformatics, Inspiral workflows for detecting gravitational waves, Epigenome for genome sequence operations and Broadband to simulate the impact of an earthquake. The workflows have also been used for business community such as financial modelling, data center automation, data modelling and ATM processing etc. To deploy and execute these scientific and business workflow applications, various workflow tools such as Pegasus, Hamake, Oozie, YAWL can be utilized among existing Cloud platforms such as Amazon EC2, Nimbus, Openstack, Opennebula etc. These workflow engines and tools have been compared in Table 2.4 based on workflow specification language, Cloud platform, dependency mechanisms and event notification etc. The existing workflow engines are listed below:-

- *Pegasus*[76]: Pegasus encompasses a set of technologies that help workflow-based applications execute in a number of different environments including desktops, campus clusters, grids, and now Clouds. It is used to map abstract workflows to concrete that can be deployed on Cloud platforms such as Hadoop, Future Grid, Nimbus, Amazon EC2 etc.

- *Hamake*[77]: It is a lightweight client-side service that does not entail any installation and has a simple syntax for workflow definition.

- *Oozie*[78]: Oozie has open-source, extensible, scalable and data aware workflow features used to manage the data processing jobs for Apache Hadoop.

- *Azkaban*: Azkaban is the simple batch scheduler for erecting and running Hadoop jobs or other processes.

- *Cascading*[79]: It has a data Processing API, Process Planner, and Process Scheduler used for defining and executing complex, scale-free, and fault tolerant data processing workflows on an Apache Hadoop.

- *OrangeScape Studio*[80]: This tool is useful for designing process-oriented business applications and the process is implemented in a visual modeling environment by creating workflows.

- *JBOSS*[81]: Jboss can be used for executing workflows with Octopus that further can be integrated in any Cloud platforms.
- *YAWL*[82]: Yet Another Workflow Language(YAWL) is an open source workflow management system used for analysis of and validation of workflows.

TABLE 2.4: Existing Workflow Deployment and Execution Tools

Workflow Engines	**Workflow Specification Language**	**Cloud Platform**	**Dependency Mechanism**	**Event Notification**	**Requires Installation**	**Web Service**
Pegasus	Java,XML	Eucalyptus, Amazon EC2,Open Nebula, Haddop	Explicit	Yes	Yes	Java API
Hamake	XML	Hadoop	Data-driven	No	No	Console/log messages
Oozie	XML	Hadoop	Explicit	No	Yes	Web Pages
Azkaban	Text file with key/value pairs	Hadoop	Explicit	No	Yes	Web pages
Cascading	Java API	Hadooop	Explicit	Yes	No	Java API
Orangescape Studio	MYSQL,DB2	Google app Engine, Microsoft Azure,IBM	Data driven	Yes	No	Web pages
JBOSS	Java	Octopus	Explicit	Yes	No	Java API
YAWL	XML	None	Data driven	Yes	Yes	Web Pages

The comparative analysis described that some of the tools such as Pegasus, Azkaban, Oozie, Hamake can be used along with Cloud platforms. Pegasus can be used to deploy and schedule multiple scientific applications in any of the Cloud platforms such as Amazon EC2, Hadoop, Nimbus etc. Oozie, Hamake and Azkaban can be used to deploy and execute workflow applications on Hadoop. To ease the work on these Cloud platforms, some of the authors have proposed simulators for running large scale applications. These simulators include GridSim [83], CloudSim [84], CloudAnalyst [85] and EMUSIM [86]. But, existing simulators fail to provide an organization for different systems to handle failures and task clustering. Therefore, Deelman et al. [87] have proposed a new simulator WorkflowSim which is the extension of CloudSim simulator. They have evaluated the performance of scientific workflow applications using scheduling heuristics such as FCFS , Min-Min , Max-Min , MCT [88][66] and fault tolerant clustering techniques with retry method in WorkflowSim. For the deployment of the workflow applications in Cloud environment, Cloud platforms have been given in detail below and their comparative analysis is also given.

2.1.4 Emerging Cloud Platforms

As the computing industry moves toward granting IaaS, PaaS and SaaS to the users , hence, a number of academic and industrial organizations have developed various Cloud platforms as mentioned in Table 2.5. Google Apps Engine is being utilized as a public Cloud for PaaS to run web applications written in python and also provides web administrative along with auto scaling, provisioning and fault tolerance features. Amazon EC2 affords a virtual environment where user creates a Amazon Machine Image (AMI) and accesses all the services as a public Cloud. It is also being employed for autonomic resource provisioning, fault tolerance and load balancing etc. Hadoop also intends to provide PaaS as public Cloud platform for processing big data applications using map reduce framework. It also offers scalability, resource provisioning and data replication in case of any failure occurred. Microsoft Azure endows with integrated development Cloud Computing environment as to develop both web based and non web based applications using .NET framework along with data replication fault tolerant approach. Aneka presents .NET-based service-oriented resource management platform that support PaaS that further can be deployed on Amazon EC2 whereas OpenStack and Eucalyptus provide infrastructure for web and non web based applications using Java language [89].

This section offered an extensive survey of the workflow scheduling algorithms, evaluated QoS parameters, workflow engines and existing Cloud platforms. There are very few approaches discussed above which can schedule the workflow tasks in a reliable manner. Hence, to enhance the reliability and availability of workflow tasks, failure prediction is an important aspect which needs to be discussed. Hence, the next section illustrates the failure prediction approaches prior to fault tolerant techniques.

2.2 Failure Prediction Approaches

It has been explored that significant work has done by a number of researchers to perform fault prediction using variety of approaches to obtain better prediction results and some of the authors have used machine learning approaches for software fault prediction and resource provisioning. But, these approaches have not been employed for predicting task failures using scientific workflows data in a Cloud

TABLE 2.5: Key Properties differentiating Existing Cloud Platforms

Properties	**Cloud Platforms**						
	Google AppsEngine	**Amazon EC2**	**Hadoop**	**Microsoft Azure**	**Aneka**	**Open Stack**	**Eucalyptus**
Cloud	Public	Public	Private	Private	Private	Private	Private
Focus	Platform	Platform,Infrastructure	Platform	Platform	Platform	Infrastructure	Infrastructure
Application Type	Web	Storage and Compute	Data intensive	Web	Compute	Web and Non Web	Web and Non Web
User access interface	Web-based Administration Console	Hadoop	Explicit	Workbench	EC2, OCCI API	Web-based Administration Console	Web Portal
Programming Framework	Python,Java	AMI,Amazon Map reduce	Java and Python	Microsoft.NET	Microsoft.NET	Java	Hibernate,Java
Load Balancing	Yes	Yes	Yes	Yes	No	Yes	Yes
Provisioning	Yes	Yes	Yes	Fixed Template	Yes	Yes	Fixed Template
Fault Tolerance	Automatic	Checkpointing	Data Replication	Replication	VM Migration	VM migration	Automatic

environment. Therefore, in this section, an extensive survey has been done which is related to predicting and detecting task failures using general approaches, specific approaches in Cloud and machine learning approaches.

2.2.1 General Approaches

Fault detection and prediction approaches have been proposed and implemented by many of the authors in distributed environment. Duan et al. [90] proposed a fault classification and data mining approach to predict different types of faults and another work of Jitsumoto et al. [91] used fault detector to differentiate between hardware, process, and transmission faults. Fu and Xu [92] built a neural network to approximate the number of failures in a given time interval and they have also implemented failure proactive prediction framework for predicting component failures based on the concept of temporal and spatial correlations using various features such as resource utilization, packet count and system information [93]. They have used supervised learning approaches to predict failures with average accuracy of 70.1% with online prediction and 74% with offline prediction. Further, the work of Sindrilaru et al. [94] implemented the technique for detecting faults before they accomplish the actual workflow engine by intercepting SOAP (Simple Object Access Protocol) messages and to provide better reaction times. Another work of Yu et al. [95] proposed Failure Aware workflow scheduling approach to predict online resource failures but needs to be implemented in workflow management tools like Pegasus for predicting task failures also. Guan et al. [96] have also proposed failure detection and prediction approach with health care data using decision trees and Bayesian networks. Although, a number of authors have proposed various approaches for fault prediction using health care data yet they have not implemented task failure predictions after considering scientific workflows data which can be used to increase the performance of workflow scheduling approaches.

2.2.2 Specific Approaches in Cloud Environment

Zhao et al. [20] have implemented heartbeat message protocol for detecting replica failures. A similar work has also been proposed by Jhawar et al. [97] using heartbeat message protocol for detecting crash failures among VM instances in Cloud environment but they have not considered task failures for scientific applications. Another work by Guan et al. [98] has also presented fault prediction mechanism for building dependable Cloud Systems with Bayesian classifiers and decision trees using health data collection. Poola et al. [99] have proposed fault tolerant workflow scheduling to reduce the cost by 70% and also suggested in their future work that failure prediction approaches can be used to reduce the cost for scientific workflows. Further, more machine learning and statistical techniques need to be enhanced to increase the performance of these approaches, henceforth; very few researchers have worked on failure prediction in Cloud.

2.2.3 Learning Machine Approaches

Catal and Diri [100] have shown in their review that the machine learning models have enhanced characteristics for prediction than statistical methods. Therefore, there is a need to explore machine learning algorithms for predicting task failures proactively. Malhotra and Jain [101] verified that Random Forest gave better results for fault prediction. Firstly, the work of Ohlsson et al. [102] applied multivariate analysis techniques for predicting fault-prone classes using software design metric by getting data from Ericsson Telecom AB. Currently, the work of Suresh et al. [103] demonstrates that the multivariate logistic regression is an efficient approach for software fault prediction. Islam et al. [104] proved that the accuracy of ANN is superior for the prediction of resource utilization in Cloud environment. Another similar work of Kousiouris et al. [105] estimated resource provisioning using Artificial Neural Networks (ANN) in Cloud platforms and also confirmed the better accuracy of prediction with ANN. Naive Bayes Model is measured as the robust machine learning algorithm for software fault prediction by the work of Catal [100]. Although, most of the existing work have used machine learning approaches such as Naive Bayes, ANN, Random Forest and LR for resource provisioning and software fault prediction yet very few work from researchers have reported on predicting task failures of workflow applications. Only, one of the

work by Samak et al.[67] have revealed that the failure analysis of jobs with Naive Bayes model can predict the failure probability of workflow jobs with the average accuracy of 85% using scientific applications such as Broadband, Epigenome and Montage. Moreover, they have not compared the performance of Naive Bayes with other machine learning approaches such as ANN, LR and Random Forest and not being considered task failures due to resource overutilization or resource failures. Therefore, none of the above discussed approaches have predicted task failures intelligently using variety of machine learning approaches.

2.2.4 Research Challenges of Task Failure Prediction in Cloud Computing

Accurate failure predictions can help in mitigating the impact of failures for scientific applications. Various Cloud resources, applications, and services can be scheduled efficiently to edge the effect of failures. However, providing accurate predictions sufficiently ahead is a challenging task for intricate applications such as workflows and an accurate prediction of task failure is a pre-requisite for implementing intelligent fault tolerant approach for scientific workflows. The state of the art of existing approaches for failure prediction revealed that some of the research challenges can be resolved using intelligent failure prediction model on large infrastructure such as Cloud and these challenges are sketched out as follows:

- Although one of the authors Samak et al. [67] have implemented job failure prediction model for application failure and data management component failures yet they have not implemented the prediction during task failure occurs due to resource overutilization which can further enhance the accuracy of failure prediction models.

- For efficient forecasting, enhanced failure prediction techniques are required to be implemented which can increase the accuracy of predictions among others. Further, it would also be useful to handle current research challenges of Cloud such as proactive fault tolerance, scientific data management and scheduling etc.

- Cost reduction is an important challenge for scientific applications. Henceforth, efficient mechanisms are required that predict the resource failures

along with execution cost proactively to reduce the maintenance cost for the scientific applications.

- Scientific workflows may consist thousands of tasks along with their dependencies. The main challenge for these workflows is to schedule and execute workflow tasks on Cloud resources which are not currently overutilized or unavailable. Therefore, prior to schedule the task on these resources, prediction based techniques need to be used to intelligently predict overutilized or unavailable resources.
- As the scientific workflows consist of heterogenous data, thus, data management problems can be resolved by predicting resource utilization parameters.

After predicting task failures proactively , various fault tolerant approaches need to be explored as discussed in the next section.

2.3 Fault Tolerant Approaches

Fault tolerance is a major concern to guarantee availability and reliability of critical services as well as application execution. However, it is also possible to anticipate failures and proactively take action before failures occur in order to minimize failure impact on the system and application execution. In this section, the state-of-the-art of fault tolerance in Cloud Computing has been discussed that is based on its policies [106] and tools. Fault tolerance polices have been categorized below:

2.3.1 Reactive Fault Tolerance

Reactive fault tolerance policies reduce the effect of failures on application execution when the failure effectively occurs. Reactive fault tolerance keeps parallel applications alive through the recovery from experienced failures. There are various techniques which are reactive like checkpoint/restart, replay and retry.

(i) *Check pointing/ Restart*: Checkpointing allows task to be restarted from the recently check pointed state rather than from the beginning during any failure. Checkpoints can be created at fixed intervals or at particular points during the computation determined by some optimizing rule. It is an efficient fault tolerance technique for long running applications [107]. Check

pointing, can be used to minimize the cost and volatility of resource provisioning. SHelp is an open source tool that provides lightweight checkpoint and rollback mechanism.

(ii) *Replication*: The basic idea behind replication is to have the replicas of a task running on different resources, so that as long as not all replicated tasks crash, the task execution would succeed. Replication algorithms are simpler to implement and scale much better. On the design perspective, software replication is relatively generic and transparent to the application domain. The limitation can be the failover time, the time it takes to select a new primary in case of failure may be unacceptably high. Replication can be implemented in any of the tools like HAProxy, Hadoop and AmazonEc2.

(iii) *Job Migration*: The alternate task technique submits a failed task to another node. It is the most basic and simple technique among all other techniques. Job migration greatly improves the capacities and the features of Cloud environments. It facilitates fault management, load balancing and low-level system maintenance. Job migration operations imply more flexible fault management. When a VM is deployed on a node, if there is any failed task then it can be migrated to another machine. This technique can be implemented by using HAProxy.

(iv) *SGuard*: SGuard is less disruptive to normal stream processing and leaves more resources available for normal stream processing than previous proposals. SGuard is based on rollback recovery [108] that saves the check pointed state into a new type of distributed and replicated file system Hadoop Distributed File System (HDFS). The overall resource utilization would be enhanced by using SGuard technique that further can be implemented in HADOOP, Amazon EC2 Cloud [109].

(v) *Retry*: It retries the failed task on the same Cloud resource to diminish the number of failures in successive attempts.

2.3.2 Proactive Fault Tolerance

Proactive fault tolerance avoids node failure by executing large scale applications in the Cloud environment. It is being utilized for migrating the application parts from the failure nodes to other working nodes.

(i) *Software Rejuvenation*: Software rejuvenation is a technique of proactive fault tolerance that designs the system for periodic reboots. Micro- rejuvenation is a technique of small reboots done frequently to extend the time without a failure. It restarts the system with clean state. A software system is known to suffer from outages due to transient errors and the state of software system degrades as time goes [5].

(ii) *Proactive Fault Tolerance using Preemptive Migration*: Preemptive migration relies on a feedback-loop control mechanism where the application health is constantly monitored and analyzed. It is reallocated to improve its health and avoid failures. This technique is a closed- loop control similar to dynamic load balancing.

(iii) *Proactive Fault Tolerance using VM Migration*: VM migration is used to migrate the VMs from faulty host to other working host which can be can be reactive or proactive. But proactive fault tolerance using VM migration is more useful to implement autonomic fault tolerant techniques which are currently required by every Cloud users. VM Migration policies are implemented first by applying the overloading detection algorithm to check whether a host is overloaded or not. The general flow of VM Migration policies is shown in Figure 2.1 and has been described in three steps given below:-

- *Host Overload or Underloaded Detection Algorithms*: Overload and underload detection is based on the idea of setting upper and lower utilization thresholds for hosts and keeping the total utilization of the CPU by all the VMs between these thresholds. Therefore, novel techniques are used for the auto-adjustment of the utilization thresholds based on a statistical analysis of historical data collected during the lifetime of VMs. An Adaptive Utilization Threshold Median Absolute Deviation is a measure of statistical dispersion where the Linear regression is used for fitting simple models to localized subsets of data for designing a curve that approximates the original data.

- *VM Migration Policies*: After selecting the host which is overloaded and predicted task failures onto the overloaded virtual machine, the next step is to select particular VMs to migrate from the host. After the selection of a VM to migrate, the host is checked again for being

overloaded. If it is still considered being overloaded, the VM selection policy is applied iteratively. This is repeated until the host is considered as being not overloaded. There are various VM Migration Policies which has been discussed by authors in Cloud environment. Anton Beloglazov and Rajkumar Buyya [110] have proposed VM migration policies like Minimum Migration Time (MMT) policy, Maximum Correlation (MC) Coefficient and Random Choice (RC) policy. They have used MMT policy to migrate the VM that takes minimum time to complete a VM migration process. Whereas, RC policy selects a VM randomly and MC [111] finds the correlation between virtual machines and the resource utilization by the applications running on the Cloud server. But they have not shown the correlation between the overloaded virtual machine with other machines using single parameter and multiple parameters. The overloaded virtual machine can be highly associated with other machines on the same host, hence, there is need to find again the inter-correlation between overloaded virtual machine and other machines through single and multiple parameters as to increase the performance by reducing overall mean time and standard deviation time. They have also not optimized the policy for handling task as well as resource failures proactively due to overutilized hosts. Therefore, there is a need to build VM migration policy that works as fault tolerance technique to migrate and consolidate the virtual machines automatically during the predicted task failures on overutilized host. Once, the list of VMs to be migrated from the overloaded hosts then VM placement approach is invoked to find a new placement for the VMs to be migrated.

- *VM Placement Policies*: VM placement policies are implemented based on Service Level Agreement (SLA) between consumer and provider as to reduce SLA Violations. Jing Tai Piao et al.[112] introduced Network Aware VM migration approach that places the VMs on physical machines with consideration of the network conditions between physical machines and data storage. This approach could improve the task execution time. But this approach does not guarantee the enforcement of SLA. Red Hats System Scheduler [113], and Platforms VM Orchestrator [114] provide placement policies with the aim of load balancing and energy consumption saving. Clark et al. have proposed policy to

continuously monitor the utilization of available hosts as to perform a re-mapping using live migration [115], which guarantees the fulfillment of policies during runtime.

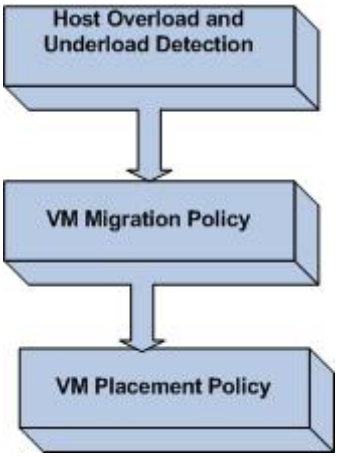


FIGURE 2.1: General Flow to implement VM Migration Approach

Hence, most of the VM migration approaches have been used to optimize the energy. As the VM migration can enhance the performance of various Cloud applications for handling task and resource failures through single resource utilization and multiple resource utilization parameters, henceforth, VM migration can also be used to implement the fault tolerance in Cloud which can further be utilized to enhance the efficiency of workflow scheduling algorithms.

To handle the failures autonomically in the Cloud environment without the human intervention, self-healing techniques are required to be explored for implementing autonomic fault tolerance in Cloud which are discussed in the subsequent section.

2.3.3 Autonomic Fault Tolerance

Autonomic Fault tolerance is the technique to self-heal the system from various failures and continue its operation in spite of system failures through fault tolerant approaches. In order to provide the autonomy to workflow scheduling, self-healing approaches have also been surveyed out and are sketched below.

2.3.3.1 Self-Healing Approaches

Self-healing is the property of healing various faults automatically without human intervention [116]. Table 2.6 shows the categories of self-healing techniques along with their impact, platforms and environment.

- *Embedded Systems Based Approaches*: Glass et al. [117] have proved in their work that previous models have only one resource type dispensed for each task and have also suggested for considering associations between various resources and tasks. Further, self-healing justification for point-to-point network is [116] explained by the research in [118] which specified the recovery strategy with combination of self- reconfiguration and self-routing. An unused shadow task updates status by the accepted tasks using check-pointing technique.

- *Model Based Approaches*: Cheng et al. [119] extended the architecture-based self-adaptation with an adaptive learning approach that accounts and remembers human administrator's decisions [120]. A Model Driven Approach has also proposed for modeling the functional part and the self-healing part of the system but this approach has not considered real world applications [22]. Hence, Nichols et al. [121] presented a model-based approach for autonomic execution of workflows in Grid environment.

- *Architecture Description Languages (ADL) Based Approaches*: Georgiadis et al. [122] implemented a monitoring tool which is dependent upon Darwin ADL and Jackson's language to analyze the runtime structure of a distributed system. Aldrich et al. [123] also suggested that the incorporation of Arch Java language with Java would be simpler to examine the self-healing properties. Dashofy et al. [120] also presented another approach in which the architecture of the software system as symbolized by xADL 2.0 (extensible XML-based ADL). Another approach proposed by Dabrowski et al. [124] which is an architecture based modification in a service discovery system that would be helpful to find each other over a network. Rakic et al. [125] also discussed the design of an exact architectural style for sustaining self-healing systems which is known as PitM.

- *Agent Based Approaches*: Agent based systems provide robustness by redundancy that can handle unpredicted situations in environment and to obtain the objectives. These systems consider multiple cooperative persistent agents [126]. Corsava et al. [127] presented an intelligent architecture for the agents but agent robustness could not only be initiates from agent redundancy. Therefore, Huhns et al.[128] used the concept of robust redundancy by using approaches like distributed approach and centralized. Tesauro et al. [129] implemented a centralized approach and developed software architecture which is known as unity for self-management through component interaction. However, they have not designed any algorithm that can further make accurate and common inferences for prompt-based responses which are employed in the system. Therefore, Ahmad et al.[130] also presented a mulitagent-based approach for particle swarm optimization. However, it also needs comparative analysis with previous one. Erola et al. [131] proposed a multi-agent system for dynamic and simultaneous scheduling and machines in manufacturing systems with comparative analysis but also needs refinement to schedule the applications [132].

- *Web Based Approaches*: Modafferi and Conforti [133] presented a non-intrusive approach, which extends the design language of Business Process Execution Language (BPEL) workflows through appropriate annotations instead of implementing a new. Modafferi et al. [134] also extended a standard BPEL engine by composing management access to a Self-healing BPEL workflow engine. The extension of a presented engine has described in the work of Subramanian et al. [135]. The self heal BPEL engine interfaces in the company of the ActiveBPEL engine that is an extension of the work presented in [136] [137] [134]. Moo-Mena et al. [138] have proposed service related approach of self-healing infrastructure that considers non-intrusive communication flow QoS monitoring for service degradation detection. Halima et al. [139] described the combination of reactive and proactive self-healing policies called QoS-oriented self-Healing. They have used the proposed policy to improve SOAP messages with QoS metadata. Moser et al. [131] proposed Vienna Dynamic Adaptation and Monitoring Extension (VieDAME) approach that focuses on BPEL controlled service invocations and further divides into two parts: the core and the engine adapters but not considers web services failures. Therefore, Mahmoud Hussein Zadeh et al.

[140] presented a web-based approach used to reduce the web services failures in service-oriented architectures and needs to improve the performance. Dai et al. [141] presented an approach for web service composition that has better performance while supporting self-healing. However, this approach does not use monitoring data and recovery factices. Therefore, Marko et al. [142] proposed an analytical product line based approach for web applications and to support monitoring data and recovery factices. However, this approach increases the complexity as well as cost [117].

- *Control Theory Based Approaches*: Benoit Gaudin et al. [143] have proposed the control theory based approach that deals with un-handled exceptions within an executing program. However, it cannot handle software failures.

- *Rule Based Approaches*: Qianxiang Wang [144] projected a rule model used to take out spotted rules using different procedures and to improve the self-adaptive ability of software. To resolve potential conflicts, a visual tool is required to define policy that generates the rule model from policy. The policy and rule have related meanings, where policy is used as a high-level statement of the rules. Kankaanniemi et al. [142] also proposed a policy driven self-healing algorithm that automatically detects, diagnoses and recovers the problems occurred during the context aware systems adaptation processes. An implementation of the action selection phase using reinforcement-learning algorithms is required. Komi S Abotosi et al. [145] presented a rule-based approach and new framework BP-FAMA is proposed. In this work, the set of business rules have been converted into cause effect graph. Nevertheless, there is a need to increase the flexibility. Wlodzimierz Funika et al.[146] introduced the proposition of a new role-based approach to resolve the problems of self-healing system under monitoring. The rules and actions can be stored into ontology.

- *Connector Based Approaches*: Michael E. Shin et al. [147] described the connector-based self-healing method to manage the faults in the components of system. They have also extended self-healing mechanism for reliable components but there should be inter-components communication on the level of software architecture of systems.

- *Process Migration approaches*: Rao et al. [148] have proposed, "VCONF: A Reinforcement Learning Approach to Virtual Machines Auto-configuration"

which is able to locate best configurations in small-scale systems and also shows more flexibility and scalability. However, other parameters like network and disk bandwidth have not been considered. Stefania Montani et al. [149] presented a case based reasoning approach that reduces the knowledge acquisition bottleneck in software systems and needs to be integrating with other approaches. R.H. Bordani et al. [150] presented an approach for designing, deploying, complex distributed software systems that can further implemented in Cloud environment.

- *Biology Inspired Approaches*: Teemu [151] discussed the programming paradigm inspired by properties of the biological cell. It also presents the software architecture for distributed systems based on the model of a biological ecosystem.

- *Machine Learning Based Approaches*: Catal et al. [152] revealed in their review that the machine learning models have improved characteristics for failure prediction to autonomically heal various faults. Malhotra et al. [101] also verified that Random Forest gave better results for fault prediction. Although, most of the existing works have used machine learning approaches such as Naive Bayes, ANN, Random Forest and LR for resource provisioning and software fault prediction to heal various faults.

Most of the existing self-healing approaches have been designed and implemented in distributed environments and only a few techniques such as agent based, web based, process migration based have been implemented in Cloud environment. Therefore, there is a need to implement embedded system based, model based, ADL based, control theory based, rule based, connector based , biology inspired based techniques and machine learning approaches in Cloud for achieving Autonomic Fault Tolerance in Cloud.

2.3.3.2 Research Challenges: Self-Healing Approaches

An autonomic adoption of machine learning and data mining techniques would be able to predict the failures and heal themselves using various techniques discussed below :-

- Embedded system based, model based and control theory based approaches could be used to measure reliability and failure rate with failure models.

TABLE 2.6: Self-Healing Approaches

Approach	Current Impact	Platforms	Environment
Embedded System based Approach	Calculates reliability of system and reconfigures network	Analytical Approach	Distributed
Model Based Approach	Records and remember human administrator decisions , optimally configures and Improves dependability	ASSL language	Distributed, Cluster
ADL-based Approach	Simplifies the concept of self healing with autonomic monitoring	x-ADL,JAVA	Distributed
Agent Based Approach	Monitors, debugs the errors	JADE,MAP-REDUCE	Distributed ,Cloud
Web Based Approach	Tolerates run-time and unexpected faults	BPEL ActiveBPEL, SOAP	Distributed, Cloud
Control theory based Approach	Handles-runtime exceptions	Java	Distributed
Rule-Based Approach	Manages-business process to resolve the monitoring problems	AOP, BbBPDL	Distributed
Connector Based Approach	Isolates objects under repair	Connector based model	Distributed
Process Migration based Approach	Automates the VM configuration process ,reduces the Knowledge acquisition bottlenecks	Xen based	Cloud, Distributed
Biology Inspired Approaches	Combines different approaches	Cell based programming model	Distributed
Machine Learning Based Approaches	Predicts failures	Weka Explorer	Distributed

These approaches can further be utilized for increasing power flexibility for future cloud whereas connector based approaches would be useful for huge database management.

- As for autonomic monitoring, ADL based and web based approaches might handle proactive fault tolerance of Cloud services.
- Rule based approaches can be enhanced for intelligent monitoring of Cloud infrastructure, software's and platforms.
- Multi-agent based approaches need to be added with Cloud system to make it more reliable and robust.
- Biology inspired approaches could be able to optimize energy efficiency and autonomic configuration.

- Machine learning approaches can be used for intelligent failure prediction that further can aid autonomic fault tolerant framework.
- Future outcomes also comprise the creation of Cloud based systems and architectures with intelligent prediction models that can be appended to the system maintenance schema in self-healing systems as to diminish the cost and execution time.

This section details autonomic fault tolerance using self-healing along with the future directions. Further, to implement fault tolerance in Cloud, existing tools have been compared along with discussed fault tolerant techniques in the subsequent section.

2.3.4 Existing Tools: Fault Tolerant Techniques

This section present the state-of-the-art implementations of fault tolerance techniques for Cloud environment. Firstly, the existing tools currently used for implementing fault tolerance have been discussed. Then, the comparative analysis has been performed that analyses these tools for implementing the fault tolerance techniques.

- *HAProxy*: It stands for High Availability Proxy and is used by companies such as right scale for load balancing and server fail over in the Cloud [153]. It is also being utilized to handle the fault tolerance through various technique such as replication, job migration and proactive fault tolerance in virtual machine environments.
- *Assure*: ASSURE [154] introduces rescue points which are locations in existing application codes for handling programmer anticipated failures. It can be used to bypass the path which induces software failures and recover from software failures by using the error virtualization technique to force an error return using an observed value in a function. ASSURE uses a production system and a triage system to implement rescue points and error virtualization.
- *Shelp*: SHelp [155] is a lightweight runtime system that can survive software failures in the framework of virtual machines. SHelp extends ASSURE to a virtualized computing environment with two distinctive features. By

TABLE 2.7: Existing Tools for implementing Fault Tolerance in Cloud

Tools	**Techniques**	**Policy**	**Language**	**Environment**	**Failure**	**Applications**
HAProxy	Self Healing, VM Migration, Replication	Reactive/proactive	Java	Virtual Machine Environment	Process/node failures	Load balancing,fault tolerance
SHelp	Checkpointing	Reactive	SQL,JAVA	Virtual Machine Environment	Application Failure	Fault tolerance
Assure	Check pointing, Retry, Self Healing	Reactive/proactive	JAVA	Virtual Machine Environment	Host, Network Failure	Fault tolerance
Hadoop	Job Migration,Replication,Sguard	Reactive/proactive	Java,HTML,CSS	Cloud Environment	Application/node failures	Data intensive applications
AmazonEC2	Replication, guard	Reactive/proactive	Amazon Machine Image	Cloud Environment	Application/node failures	Load balancing, fault tolerance, workflow scheduling.

introducing this two-level storage hierarchy for rescue points in the virtual machine environment, SHelp can dramatically reduce the redundancy. Shelp can also work in Cloud environment by implementing check pointing technique.

- *Hadoop*: Hadoop [156] is an open-source java-based software platform developed by the Apache Software Foundation. Hadoop implements Googles Map-Reduce programming model on top of a distributed file system called the Hadoop Distributed File System (HDFS) which is designed to reliably store very large files across machines in a large cluster. These blocks of a file are replicated for fault tolerance as data replication technique.

- *Amazon Elastic Compute Cloud (EC2)*: It [157] provides a virtual computing environment that enables a user to run Linux-based applications. Amazon Web Services (AWS) provides a platform that is ideally suited for building fault-tolerant software systems. The AWS platform is unique because it enables to build fault-tolerant systems that operate with a minimal amount of human interaction and with minimum cost. It can be used to implement replication, SGuard technique for Cloud.

Table 2.7 summarized the tools of popular Cloud offerings in terms of the types of application and more importantly their programming framework, policies, fault tolerant technique and fault type. Apparently, these Cloud offerings are based on different types of fault tolerance techniques and implementation of these techniques in different tools. Users can choose any one type of tool and fault tolerance techniques which can be easily implemented in Cloud environment. The research issues which have evolved from the survey of fault tolerant techniques are discussed below in the next section.

2.3.5 Research Challenges: Fault Tolerant Techniques

Providing autonomic fault tolerance requires careful consideration and analysis because of their complexity, inter-dependability for the following reasons:-

- There is a need to implement an autonomic fault tolerance technique for multiple instances of an application running on several virtual machines.

- Different technologies from competing vendors of Cloud infrastructure need to be integrated for establishing a reliable system.
- The new approach needs to be developed that integrates these fault tolerance techniques with existing workflow scheduling algorithms.
- A benchmark based method can be developed in Cloud environment for evaluating the performances of fault tolerance component in comparison with similar ones.
- Autonomic fault tolerant can be implemented using VM migration after predicting the failures through machine learning approaches.
- Most of the existing works have been implemented reactive or proactive fault tolerant techniques, therefore autonomic fault tolerant would be useful for handling faults autonomically in Cloud Environment which has not been implemented by any of the researchers.

Autonomic fault tolerance has many advantages if implemented in Cloud and are discussed in the next section.

2.3.6 Benefits: Fault Tolerant Techniques

The key advantages of implementing fault tolerance in Cloud Computing embraces failure recovery, lower cost, improved performance metrics [158].

- *Failure Recovery*: By implementing fault tolerance in Cloud would prompt response and recovery to any anomalous event that could be triggered due to reconfiguration, fault, performance problems and cyber attack or exploitation.
- *Performance Improvement*: The most common problem that is seen in Cloud virtualized environment is server breakdown that degrades system availability. Autonomic fault tolerance will lead to reduce the execution time and cost of failures by eliminating manual intensive activities. It can also improve the performance metrics such as availability and reliability.
- *Cost Savings*: Fault tolerant services will lead to cost reduction by minimizing the number of failures and managing runtime faults. These may also

reduce number of monitoring and management tools required to manage performance, fault, security and configuration.

- *Scheduling Efficiency*: The incorporation of autonomic fault tolerant approaches with scheduling algorithms would be able to enhance the efficiency in terms of makespan for scientific and business applications.

This section presented an extensive survey of the fault tolerant techniques, next section discusses existing fault tolerant workflow scheduling techniques that address the problem in the Cloud Computing environment.

2.4 Fault Tolerant Workflow Scheduling Algorithms

Fault tolerant workflow scheduling is a significant concern in Cloud Computing environment and efficient approaches are required for the workflow scheduling along with fault tolerance using scientific applications. A number of workflow scheduling algorithms which used the concept of fault tolerance have been analysed in different environments and these algorithms are discussed below after considering their essential features and functionalities:

- *Zheng and Veeravalli* [159] presented a fault-tolerant scheduling approach that schedules Directed Acyclic Graphs (DAGs) in grids with communication delays so that service failures can be avoided in the presence of processor faults. Their approach depends on that as tasks in a DAG have dependency on each other and the task is required to be scheduled to make persuaded and it will succeed when any of its predecessors fails due to a processor failure.

- *Zhang et al.* [160] combined fault tolerant techniques with workflow scheduling algorithms to optimize the performance of workflow applications in Grid environment.They have enhanced the reliability and performance.

- *Poola et al.* [161] [99] proposed a scheduling algorithm for scientific workflows that scheduled tasks onto Cloud resources using cost model to reduce the overall execution cost. They reduced 70% of the total cost using spot instances. Further, they have proposed resource allocation policies along

with robust and fault tolerant behaviour for scientific workflows by reducing the makespan. The robustness can be further increased by using failure probability models.

- *Sindrilaru et al.* [162] designed and developed mechanisms for building an autonomic workflow management system in grid workflow management system that can exhibit the ability to detect, diagnose, notify, react and recover automatically from failures of workflow execution. Autonomic workflow management system needs to be implemented in Cloud also so that the performance can be increased .

- *Yu et al.* [163] proposed a novel adaptive rescheduling concept, which allows the workflow planner works collaboratively with the run time executor and reschedule in a proactive way. Further, it can also be used to integrate the rescheduling with advance resource reservation and resource availability prediction model to implement the collaboration.

- *Zhang et al.* [160] presented a workflow scheduling and execution mechanisms that incorporate a balanced approach toward reliability and performance which is not very sensitive to the underlying resource reliability prediction. But these mechanisms increases the cost with over provisioning scheduling algorithm.

- *Dornemann et al.* [164] presented a approach that automatically schedules workflow tasks to underutilized hosts and provides new hosts using Cloud Computing infrastructures in peak load situations. An implementation based on the Active Business Process Execution Language (BPEL) engine and Amazon‘s Elastic Compute Cloud have also been presented in their work. In future, the failure handling and recovery approach can be used and implemented in other Clouds.

- *Hwang et al.* [165] provided a flexible framework for fault tolerance in Grid. It appeared to be essential to support multiple fault tolerance techniques and user defined exception handling in order to achieve high performance as well as fault tolerance in the presence of failures. Such a framework is required in Cloud too.

- *Chetepen et al.* [166] provided some scheduling heuristics based on task replication and rescheduling of failed jobs but it does not depend on particular

grid architecture and are suitable for scheduling any application with independent jobs. Scheduling decisions are based on dynamic information on the grid status and not on the information about the scheduled jobs.

- *Chen and deelman* [167] proposed task failure and job failures models to increase the scheduling efficiency of scientific workflow applications. By implementing through failure probability models such as weibull distribution models or machine learning models the robustness can be increased further.

- *Rahman* [168] have developed an autonomic workflow management system using reputation based dependable scheduling algorithm that enhances the reliability during workflow execution through proactive resource provisioning in Grid environment. It can further be extended for Cloud using autonomic fault tolerant techniques.

TABLE 2.8: Existing Fault Tolerant Workflow Scheduling Algorithms

Approach	**Parameters**	**Fault Tolerant Approach**	**Environment**	**Tools**	**Future Work**
Fault Tolerant Scheduling	Time and cost	Replication	Grid	GridSim	Other Scheduling Strategies
Combined Approach	Reliability	Overprovisoning, Checkpointing	Grid	GridSim	Prediction Models can be used
Fault Tolerant Scheduling using spot instances	Execution Cost, Makespan, Robustness	Checkpointing	Cloud	CloudSim	Failure prediction models for multiple scientific workflows required.
Failure detection based Scheduling heuristics	Availability	Task Replication	Grid	GridSim	Prediction based approaches can enhanced the performance of scheduling.
Fault Tolerant Clustering	Makespan	Tak Failure Model,resubmission	Cloud	WorkflowSim	Pegasus can be used to validate the performance.
Autonomic Workflow Management System	Makespan, Reliability	Weibull distribution model	Grid	GridSim	Extended for multiple scientific workflow applications in Cloud.

To tolerate the faults for scheduling, various fault tolerant approaches have been adopted by number of authors along with workflow scheduling as discussed in Table 2.8. As most of these fault tolerant scheduling techniques have been implemented in grid and distributed environment, thus, these strategies need to be

implemented in Cloud also. Fault Tolerant scheduling using spot instances and fault tolerant clustering have only been implemented in Cloud environment. The future directions of existing fault tolerant workflow scheduling conferred that the failure models can be used along with fault tolerant techniques so as to enhance the performance of multiple scientific workflow applications. Although one of the authors Rahman [48] has reported a thesis on autonomic workflow management system in Grid, yet the author has not used machine learning models for prediction and autonomic VM migration for fault tolerance along with efficient scheduling heuristics. Therefore, none of the authors have used autonomic fault tolerant approach along with workflow scheduling algorithms.

Based on the findings of the existing literature, succeeding section formulates the problem for this research work.

2.5 Problem Formulation

It is evident from the literature survey that fault tolerant workflow scheduling is inherently a challenging task in a dynamic Cloud environment. The scientific applications are looking towards the Cloud to provide a stable and customizable execution environment as in some cases running in a local environment is sufficient but is not a scalable solution. However, virtual environments offered by Clouds can provide this necessary scalability. These scientific applications are represented as Cloud workflows that consist of few tasks to million of tasks which have the dependencies between them. The problem of fault tolerant scheduling becomes complicated for multiple workflows due to the deployment of large scale multiple scientific applications that often require the availability and efficient scheduling of tasks on Cloud resources. Existing Cloud Computing systems ought to be autonomic, largely, to accomplish the requirements of multiple scientific workflow applications effectively. Thus, there is a need to implement an autonomic fault tolerant workflow scheduling for multiple workflows as to deliver high performance, reliable, scalable and robust Cloud services.

Most of the existing works employed several of fault tolerant techniques for scientific workflows such as replication, checkpointing, job migration, retry, task resubmission etc. Less research has been done to predict and detect task failures

intelligently by adopting machine learning approaches for implementing autonomic fault tolerance. Though, the workflows have been used for simulation, high energy physics, astronomy and many other scientific applications. Hence, the reliability models for software and hardware failures cannot be simply applied to handle the task failures proactively [169]. It is insightful, if the fault tolerant approach is reactive one and is not able to handle the failures intelligently [170]. Henceforth, intelligent task failure detection and prediction is mainly challenging for scientific workflow applications which have lot of job and data dependencies such as Montage, Cybershake, SIPHT, Inspiral, Epigenomics and Broadband.

Reliability and Availability would be reduced, if any task or resource failure occurs during the execution. Most of the existing works have implemented reactive fault tolerant techniques, therefore, there is a need to implement an autonomic fault tolerant approach that can be useful for handling faults proactively through failure prediction models in Cloud environment. Henceforth, an autonomic fault tolerant technique is required to handle resource overutilization or any task failures.

For the execution of workflow tasks in Cloud, scientists need to use efficient scheduling algorithms so as to schedule concurrent workflows onto the Cloud resources. Existing scheduling heuristics do not prioritize the multiple workflows concurrently on Cloud resources using Earliest Deadline First and hybrid heuristic. The hybrid heuristic can schedule the tasks based on features of various heuristics such as Max-Child, Min-Min and FCFS heuristic. It can be utilized to reduce the makespan and to further increase the resource utilization. However, none of the above said works handle the task failures autonomically during workflow scheduling.

The gaps scrutinized above necessitate devising a scheduling approach that can detect, diagnose and handle failures automatically and efficiently schedule workflow tasks onto the Cloud resources.

The objectives of the proposed work are as follows:

- To analyze the various workflow scheduling algorithms and fault tolerant techniques for Clouds.

- To develop an autonomic fault tolerant technique for scheduling in Cloud Computing.
- To design and develop a workflow scheduling algorithm for multiple workflows based on autonomic fault tolerant technique.
- To validate the proposed algorithm in Cloud environment.

2.6 Conclusion

This chapter explored the existing literature for the workflow scheduling in Cloud Computing. It presented failure prediction and fault tolerant approaches in an autonomic way. Further, the chapter analyzed the existing works on fault tolerant workflow scheduling in Cloud environment. Based on the literature survey, research problem has been devised.

The next chapter presents a solution to the research problem by proposing an intelligent failure prediction model and an autonomic fault tolerant technique in Cloud environment. The proposed technique deals with the issues identified in problem formulation and accomplishes the objectives of this research work.

Chapter 3

Proposed Autonomic Fault Tolerant Technique

In order to mitigate failure effect on the Cloud system and application execution, failures should be predicted autonomically. Thus, there is need of autonomic fault tolerance to improve availability and reliability of today's Cloud services.

This chapter focuses on the research problem of designing an autonomic fault tolerant approach. The proposed autonomic fault tolerant approach is a proactive approach that automatically predicts resource utilization parameters and task failures. For autonomic predictions, a machine learning approach is required. Therefore, the first step is to identify an appropriate machine learning approach which has maximum prediction accuracy and minimum error and design a failure prediction model based on that approach. Then, the autonomic VM migration is entailed that autonomically migrates the faulty machines to other working hosts based on failure prediction model.

Firstly, this chapter discusses the design methodology used to identify the best machine learning approach by comparing its failure predictions with actual failures in Cloud environment. Then, an autonomic fault tolerant technique has been designed using a VM migration policy through intelligent prediction of task failures. Finally, the performance parameters have been evaluated to measure the effectiveness of the proposed failure prediction model and autonomic fault tolerant technique.

3.1 Basis of the Proposed Technique

Autonomic fault tolerant technique requires prior knowledge of the task failures, hence, intelligent task failure prediction model has been proposed and implemented for multiple scientific workflow applications. The goal of the proposed model is to predict the task failures intelligently using the best classification based machine learning approach through the comparison of actual failures and the predicted failures in the Cloud environment. Then, the autonomic migration of VMs is implemented for those VMs where the tasks are predicted as failed due to resource overutilization. Thus, the proposed Autonomic Fault Tolerance technique is based on two processes:

- Intelligent Failure Prediction Model
- Autonomic VM Migration Approach

The next section discusses the steps followed to design the failure prediction model followed by the discussion of VM migration approach that automatically migrates the faulty VM on the basis of the prediction of task failures using proposed intelligent failure prediction model in the subsequent section.

3.2 Intelligent Failure Prediction Model

The proposed model considers the scientific applications for predicting task failures. These scientific applications are represented through workflows that stipulate a process or computation to be executed in the form of data flow and task dependencies. To reduce the failure effect of these workflow tasks on the Cloud resources during execution, task failures can be intelligently predicted by proactively analysing the data of multiple scientific workflows using the state of the art of machine learning approaches for failure prediction. During the scientific workflow scheduling, the task failures can occur due to overutilization of resources, unavailable resources, execution time or if execution cost exceeds the threshold value, required libraries are not installed properly, system running out of memory or disk space and so on. In this research work, the proposed model has been designed for task failures (of CPU, RAM, Disk Storage and Network Bandwidth) occurring due to the overutilization of resources.

After a concise discussion about the proposed model, the next section discusses the design methodology that evolved to implement this model.

3.2.1 Design Methodology

The objective of the proposed methodology is to design a failure prediction model that performs real time monitoring of scientific workflow data for identification of task failures. The proposed model is intelligent enough for prediction of task failures prior to scheduling of workflows by analyzing the multiple workflows data stored in Cloud repositories. The proposed model is based on an experimental study that verifies the best machine learning approach to be used for failure prediction. It has been exercised to select the most accurate approach from Artificial Neural Networks (ANN), Logistic Regression(LR), Random Forest and Naive Bayes approaches. The flow of the adopted methodology is depicted in Figure 3.1.

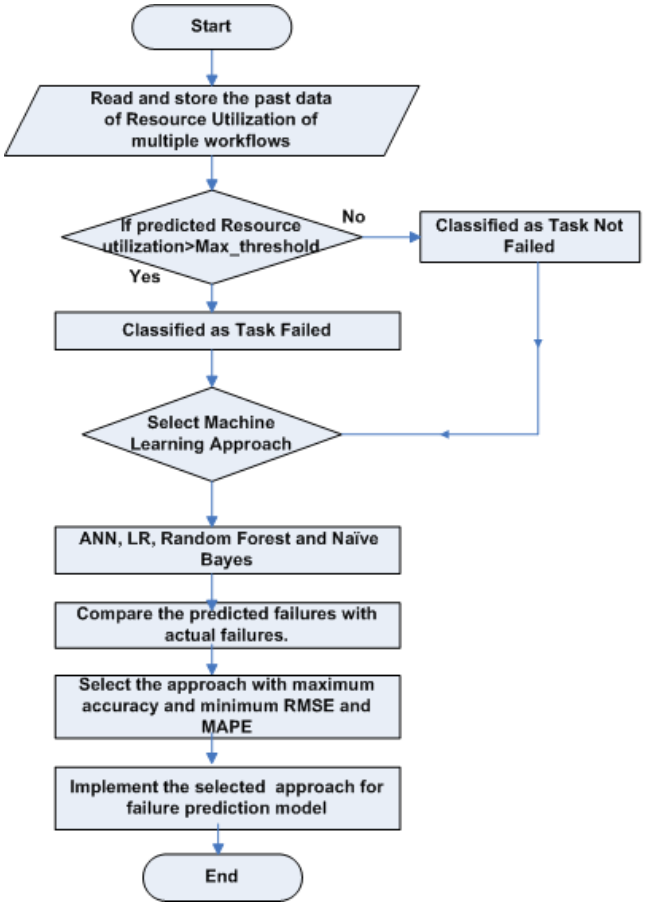


FIGURE 3.1: Flowchart for the Proposed Task Failure Model

The proposed methodology firstly scrutinizes and stores the past values of resource utilization parameters of scientific workflow applications such as Montage, Cybershake, Laser Interferometer Gravitational Wave Observatory (LIGO)-Inspiral [171]and sRNA Identification Protocol using High-throughput Technology (SIPHT) [172]. The resource utilization parameters have been evaluated using threshold value of CPU utilization, Bw utilization, RAM and Disk utilization. The threshold values are selected based on the previous history of task failure due to overutilization of VMs. If the utilization parameter has value more than the maximum threshold value then status would be classified as "Task Failed" otherwise as "Task Not Failed". The machine learning approach which has minimum error in terms of Root Mean Square Error (RMSE) , Mean Absolute Percentage Error (MAPE) and maximum accuracy has been then selected for implementing failure prediction model.

The methodology has been detailed in Figure 3.2 that works at two stages, whereas one module is used to predict the task failures using classification based machine learning approaches and second module is used to identify the actual failures after executing workflow applications on Pegasus and Amazon EC2 [173][174]. Pegasus is used to validate the experimental results that maps and executes the abstract workflows in a Cloud environment and Amazon EC2 is used for deployment as it offers a large selection of instance types such as CPU, Memory, storage and networking capacity to deploy and execute scientific workflows.

(i) *Task Failure Prediction Module*: Task failure prediction module is used to predict the task failures generated from the execution of scientific applications in WorkflowSim and CloudSim toolkits [87] [84]. The details of this model are as follows:-

- *Data Storage Unit*: The data storage stores the execution details of the task which is indexed by a unique task id for every workflow. The historical data details are accumulated in the data repository of the Cloud. For predicting the performance of failure prediction model, data of scientific workflows such as Montage, Cybershake, LIGO-Inspiral and SIPHT has been collected after fixed interval in WorkflowSim where each job can contain hundreds or thousands of tasks.

- *Data Collection Unit*: It is used to gather the data from data storage unit where CloudSim classes have been utilized to read and trace the

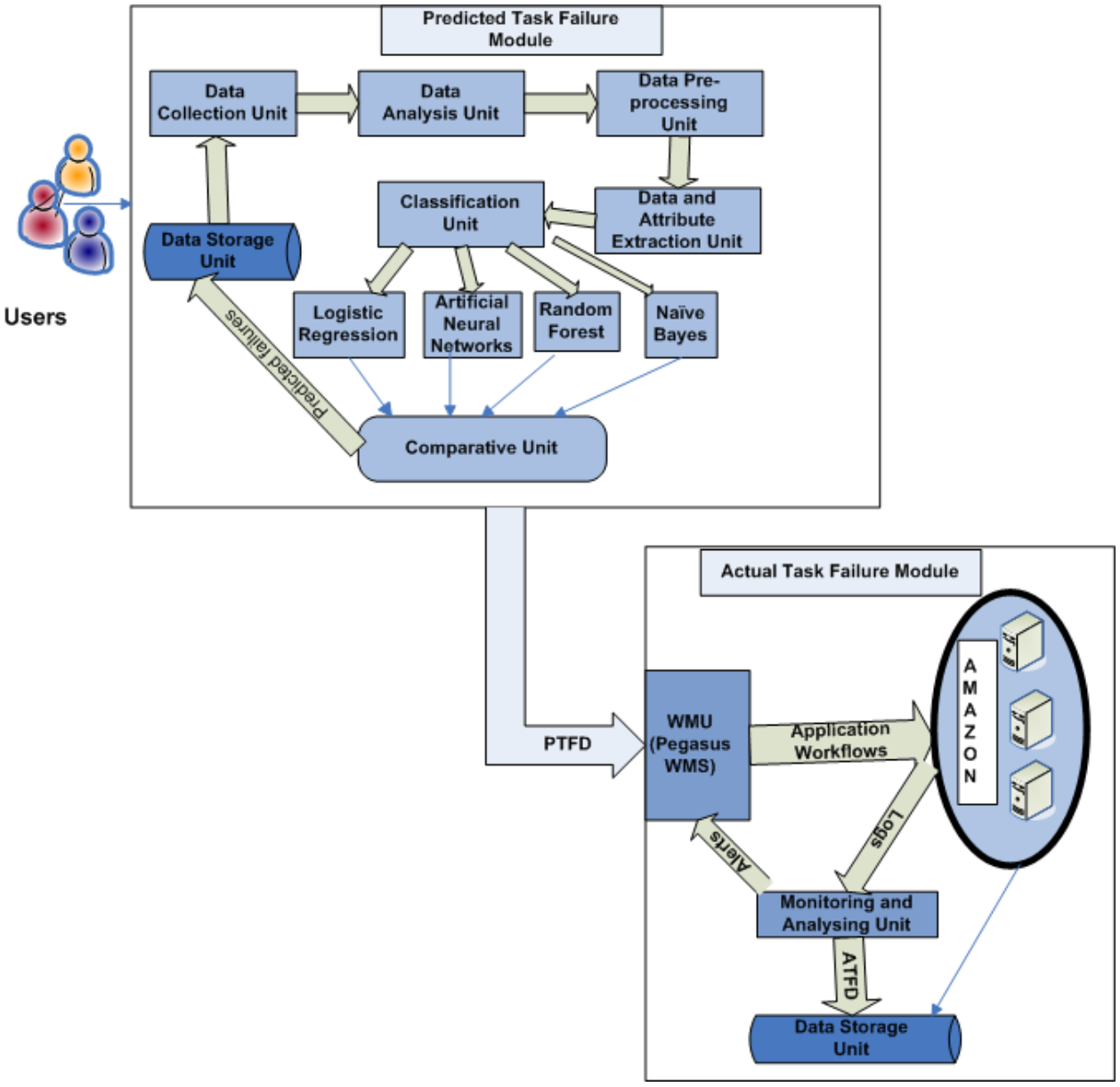


FIGURE 3.2: Methodology for Selecting Best Classification Approach

data values for the workload traces and these data values are further used to analyse the collected data.

- *Data Analysis Unit*: This unit is used to analyse the task failure data for multiple workflows. Further, the data is pre-processed and meaningful data is extracted.

- *Data Pre-processing Unit*: It involves thorough assessment of raw data where distorted values or missed readings often become misleading in the formal analysis. Henceforth, to deal with missing values interpolation method has been used.

- *Data and Attribute Extraction Unit*: Principal Component Analysis (PCA) has been applied for selecting more relevant attributes and to reduce the dimensions which are required for implementing intelligent fault prediction model. Task id indicates task number and VM id signifies VM number on which task has failed and similarly datacenter id also shows datacenter where the task would fail or succeed. As there are multiple levels in the workflows, therefore, depth specifies the level of tasks in workflows.

- *Classification Unit*: Tasks have been classified as "failed" if utilization parameter value exceeds then the maximum threshold value otherwise class is classified as task "not failed". These classes are further applied to machine learning approaches to predict the task failures. Based on the literature survey of existing failure prediction approaches, various classification based machine learning approaches such as Naive Bayes, ANN, LR and Random Forest have been used to predict the task failures. These approaches have been implemented and compared based on the evaluated performance metrics.

- *Comparative Unit*: Different evaluation measures such as Sensitivity, Specificity , Recall, Precision, RMSE, MAPE, F-Measure and Receiver Operating Characteristics (ROC) are used for comparing the performance metrics which further evaluate the accuracy of these approaches in terms of minimum error and maximum precision. The Predicted Task Failure Data (PTFD) is stored into data base which is further compared with Actual Task Failures Data (ATFD) in the actual task failure module of the proposed model.

After discussing the units of predicted task failure model, the next part of the proposed methodology discusses Actual Task Failure Module.

(ii) *Actual Task Failure Module*: Actual Task Failure module is used to assess the actual task failures using Pegasus and Amazon EC2. The details of this module are as follows:-

- *Workflow Management Unit(WMU)*: Pegasus-Workflow Management System (WMS) maps the abstract workflows to concrete workflows that is deployed and executed on Amazon EC2 Cloud with the goal of making

reasonable predictions for scientific applications and is also utilized for actual prediction of task failures by analysing scientific workflows.

- *Monitoring and Analysing Unit*: To access the actual failure, actual task failure log files are monitored using pegasus monitored class and analysed by Amazon Cloud Watch that further generates the ouptut log files of task failures. These files are stored in a relational archive, which standardizes the log files into a close approximation and sent back the alerts to the WMU.

- *Data Storage Unit*: Then, ATFD is stored in data storage unit. These failure records of actual failures are compared with stored predicted failure records to confirm the accuracy of proposed failure prediction model. The performance of various machine learning approaches has been evaluated by comparing actual task failure records and predicted task failure records. The approach which has maximum accuracy and minimum error has then been selected for implementing autonomic fault tolerant technique.

After an explanation of the design methodology for the proposed failure model, the next section discusses the machine learning approaches used for comparing the failure prediction accuracy.

3.2.2 Approaches Used for Predicting Task Failure Data

Various classification based machine learning approaches such as Logistic Regression, ANN, Random Forest and Naive Bayes have been analysed to pick the best approach for implementing intelligent failure prediction model and are detailed below.

(i) *Logistic Regression*: Logistic regression is the commonly used approach for predicting the dependent variable from a set of independent variables. In this work, failure prone classes have been predicted using multivariate with multiple independent variables of resource utilization. To find the optimal set of independent variables, there are two selection methods; one is forward selection that checks all the variables at entry time and another is backward elimination method which comprises the independent variables and deletes the variables one by one. In the proposed approach, the forward stepwise

selection method has been utilized. The multivariate logistic regression formula [175] is shown as follows in Eq.(3.1).

$$F(t) = \frac{e^t}{e^t + 1} = \frac{1}{1 + e^{-t}} \tag{3.1}$$

Logistic function F (t) is written below which takes the values between 0 and 1 then the logistic function can be written in F(x) where x is the explanatory variable as shown in Eq.(3.2)

$$F(x) = \frac{1}{1 + e^{-}(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4)} \tag{3.2}$$

The probability of the dependent variable can be "task success" or "task failure", so the inverse of logistic function is defined as shown below by g(x) in Eq.(3.3) .

$$g(x) = ln\frac{F(x)}{1 - F(x)}, g(x) = (\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4) \tag{3.3}$$

x_1 represents the CPU utilization, x_2 represents the RAM utilization, x_3 represents the Bandwidth utilization, and x_4 represents the Disk utilization where β_0, β_1, β_2, β_3 represent the constants. Hence, the probability of task failure based on four independent variables is calculated by the following formula in Eq. (3.4):

$$Prob(x_1, x_2, x_3, x_4) = \frac{e^{g(x)}}{1 + e^{g(x)}} \tag{3.4}$$

(ii) *Artificial Neural Network (ANN)*: ANN has been trained using multivariate functions. In this approach, a Neural Network model with back-propagation method is trained, with the given previous historical data from the scientific workflow execution in WorkflowSim. A typical three-layer Neural Network is shown in Figure 3.3.

The Neural Network consists of multiple layers such as input layer, hidden layer and output layer. Input layer has four input neurons ,$x = [x1, x2, x3, x4]$ and output layer has two output neurons, $o = [o1, o2]$ and one hidden layer with three hidden neurons, $h = [h1, h2, h3]$ in between them. The neurons at each layer are linked to the neurons of the next layer with a weight w_i

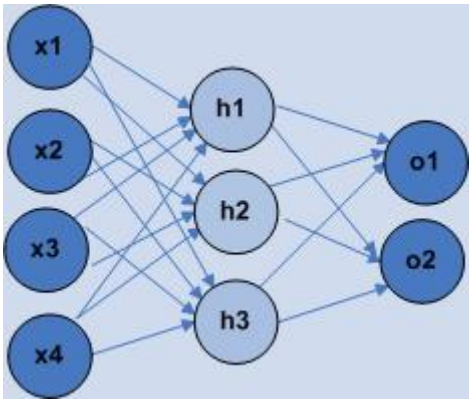


FIGURE 3.3: Three Layer Neural Network

that is to be evaluated during training. With each training point x_i and w_i network computes the resultant output y_j. If y (actual output)and y_j (predicted output) are different, the network weights are modified to reduce the calculated error. In this approach, weights are updated using learning rate p=0.7 and momentum=0.2. The sum of squared errors i.e. RMSE and MAPE are the performance measures that use a gradient-descent technique to minimize the error and to reach the local optima. When the calculated error during all training data is adequately small then the network has reached at local optima. Thus, the resultant ANN with backpropogation method has assured to generate a better prediction output after observing the input data in real time. The proposed approach has been shown in Algorithm 1.

Algorithm 1: FPNN with Backpropogation Learning
Initialize: network with random weight vector w_i
For all training examples from j= 1 to m, do
For i=1 to n do
Evaluate output $y_j = \sum_{i=1}^{n} x_i w_i$
Compare network output with correct output y with y_j
Error $e = y - y_j$
Use gradient descent to minimize the Error
Adapt weights in current layer
Repeat until the RMSE and MAPE error has been minimized
End For
End For

(iii) Random Forest: Random Forest approach can be used to generate thousands of tree. It joins the advantages of bagging and random selection methods [176]. Bagging techniques have been used to take the samples continually from various data sets with uniform probability distribution whereas random feature selection explores at each node for the finest split over a random subset of the features. The random forest categorizes a new object from an input vector by penetrating the input vector on every tree in the forest. Each tree is used to cast a unit vote at the input vector with classification and forest selects the classification which has the maximum votes over all the trees in the forest [177]. Each random tree is constructed with the following steps:

- For M number of cases in training set, sample randomly.
- For every node y fault predictors are randomly selected out of Y input variables and $y << Y$ where $y = \sqrt{Y}$.
- Each tree is developed to the large extent with no pruning.

(iv) *Naive Bayes Approach*: Naive Bayes Model has been used for fault prediction which assumes independence of attributes to each other, therefore it is named as Naive. It uses Bayesian theorem to calculate the probability of unknown instance Y and is classified as class T with possible outcomes. Where $T = \{T_1, T_2\}$, T is considered as $\{Success \mid Failure\}$ class and the probability of task failure or success is defined by the Eq.(3.5).

$$P\{T|Y\} = \frac{P(Y|T)P(T)}{P(Y)} \tag{3.5}$$

Since, the Naive assumes the conditional probabilities of the independent variables, therefore, $P\{T|Y\}$ can be decomposed into product of sums by using Eq.(3.6), where Y_j represents n attributes as: Y_1, Y_2.... Y_n which are conditionally independent on one another given T as output.

$$P(T|Y) = P(T)\prod_{j=1}^{n} P(Y_j|T) \tag{3.6}$$

The probability that T will take on any point k is defined in the Eq.(3.7). Here, the sum is considered as all the possible values t_j of T. The Eq.(3.7)

TABLE 3.1: Relation between Sensitivity and Specificity

Classified Class	**Task Failed**	**Task not Failed**
Task Failed	TP	FP
Task not Failed	FN	TN
	Sensitivity	Specificity

can be rewritten in Eq.(3.8) which is a basic equation of Naive Bayes classifier. This equation also describes to calculate the probability that T will take on any given value and can be estimated from training data with the distributions $P(T)$ and $P(Y_i|T)$.

$$P(T = t_k|Y_1...Y_n) = \frac{P(T = t_k)P(Y_1...Y_n|T = t_k)}{\sum_j P(T = t_j)P(Y_1...Y_n|T = t_j)} \tag{3.7}$$

$$P(T = t_k|Y_1...Y_n) = \frac{P(T = t_k)\prod_i P(Y_i|T = t_k)}{\sum_j P(T = t_j)\prod_i P(Y_i|T = t_j)} \tag{3.8}$$

Further, the performance of these approaches has been compared and evaluated using evaluation metrics discussed in the next section.

3.2.3 Evaluation Criteria for Validating Task Failure Prediction Approaches

To measure the performance of predicted task failure module, the results have been evaluated in Weka toolkit. The results of predicted task failures have been compared to select the best approach using evaluation metrics such as Sensitivity , Specificity, Recall, Precision, MAPE, RMSE, F-Measure and ROC . Furthermore, to validate the accuracy of actual task failure module, the predicted task failures have been compared with actual task failures using Standard Error of Mean on Pegasus and Amazon EC2 Cloud. The details of evaluation metrics are as follows:

(i) *Sensitivity and Specificity*: These metrics measure the correctness of the predicted model where Sensitivity specifies the percentage of actual faulty tasks (Task Failed) which are correctly classified whereas Specificity is the amount of non-faulty tasks (Task Success) which are correctly identified [178]. The relationship between these metrics is depicted in Table 3.1. These measures are calculated using the formulas as given below in Eq.(3.9).

$$Sensitivity = \frac{TP}{TP + FN}, Specificity = \frac{TN}{FP + TN} \tag{3.9}$$

- True Positives (TP): User tasks labelled as task failed and also evaluated as task failed.

- True Negatives (TN): User tasks labelled as not failed task but evaluated as not failed.

- False Positives (FP): User tasks labelled as a failed task but evaluated as not failed.

- False Negatives (FN): User tasks labelled as not failed task but evaluated as task failed.

(ii) *Recall and Precision*: Precision is defined as the ratio of correctly predicted failures to the number of all the recognized failures where Recall is defined as the ratio of correctly predicted failures to the number of true failures which are defined as follows in Eq.(3.10).

$$Recall = \frac{TP}{TP + FN}, Precision = \frac{TP}{TP + FP} \tag{3.10}$$

(iii) *MAPE and RMSE*: In the experimental results, MAPE and RMSE are evaluated to measure error in percentage. MAPE is used for evaluating the prediction accuracy as percentage [101]. Where y_j is the actual output, $\widehat{y}_j$ is the predicted output and m indicates total number of observations in the dataset and MAPE is defined in Eq.(3.11) and lower value of MAPE indicates a more precise prediction technique.

$$MAPE = \frac{1}{m} \sum_{j=1}^{m} \frac{|\widehat{y}_j - y_j|}{y_j} \tag{3.11}$$

The metric RMSE is described by the following formula in the Eq. (3.12) and smaller value of RMSE signifies a more effective prediction scheme.

$$RMSE = \sqrt{\frac{1}{m} \sum_{j=1}^{m} (\widehat{y}_j - y_j)^2} \tag{3.12}$$

(iv) *F-Measure and ROC*: The F-Measure values are computed as a harmonic mean of the precision and recall as shown in Eq.(3.13). The highest value of F-Measure would be considered as the best case whereas lowest value is evaluated as worst case.

$$F - Measure = \frac{2 * Precision * Recall}{Precision + Recall} \tag{3.13}$$

The accuracy of prediction approaches can also estimated by comparing their ROC curves in graphical approach. The area under curve method is used to access the ROC curve which is shown in Eq.(3.14) where tpr specifies the true positive rate and fpr indicates false positive rate [178]. The maximum value of area under curve signifies the best predictor.

$$AreaUnderCurve = \int_0^1 tpr * (fpr) dfpr \in 0, 1 \tag{3.14}$$

(v) *Standard Error of Mean (SEOM)*: The metric SEOM is evaluated by deviation of predicted task failures from the actual task failures, for high accuracy SEOM should be minimum which is expressed by the formula discussed in the Eq.(3.15). SD indicates standard deviation and m is the number of samples.

$$SEOM = \frac{SD}{\sqrt{m}} \tag{3.15}$$

3.2.4 Quantitative Analysis

The previous sections presented the design methodology of machine learning approaches to select and implement an effective approach for intelligent prediction of tasks. As per the experimental results (shown in Chapter 5), Naive Bayes is chosen as the best approach in this context. This approach validates the maximum accuracy and minimum error among other classification approaches for predicting task failures intelligently before the actual occurrence of these failures. Hence, this approach has been used to implement the failure prediction model. Next, an Autonomic VM Migration approach has been proposed to migrate the VMs in case of actual task failures which is elaborated in the subsequent section.

3.3 Proposed Autonomic VM Migration Approach

An Autonomic VM migration approach has been proposed to migrate the failed virtual machines to another host. After deploying multiple scientific workflows data in Cloud environment, if the tasks are predicted as failed using failure prediction model based on Naive Bayes approach due to overutilization of the resources, then, VM migration approach migrates the predicted faulty VMs using Inter-correlation coefficient based method.

3.3.1 Autonomic VM migration Approach using Single Parameter

Inter-Correlation Coefficient approach has been used to find the association between CPU utilization single parameter and hosts. For failed tasks, the correlation between CPU utilization and VMs has been calculated on the overloaded host. The maximum correlation between VMs and CPU utilization indicates that the task failure is due to the overloaded VM and that faulty VM machine needs to be migrated. The general flow chart of VM Migration using Inter-correlation coefficient method has been depicted in Figure 3.4.

At first, the proposed Failure Prediction Model checks the list of hosts where the tasks have been predicted as failed and gets the overloaded host. Then, proposed VM migration policy is used to select the faulty VM from the list of overloaded VMs that need to be migrated from the overloaded host. Initially, the correlation factor between $VM(i)$ and CPU utilization is evaluated, if the correlation factor is one, then $VM(i)$ is assumed to be overloaded. Further, the inter-correlation factor has been calculated between the overloaded machines $VM(i)$ and $VM(i+1)$, if the inter-correlation factor is one then there is a strong association between overloaded machines and these machines may be working together in a group and these machines are not migratable. If the inter-correlation factor between overloaded VMs is minimum , then, these machines can have weak association between them and these machines can be migratable.

The correlation coefficient has also been evaluated using CPU utilization parameter and other resource utilization parameters which are described in the next section.

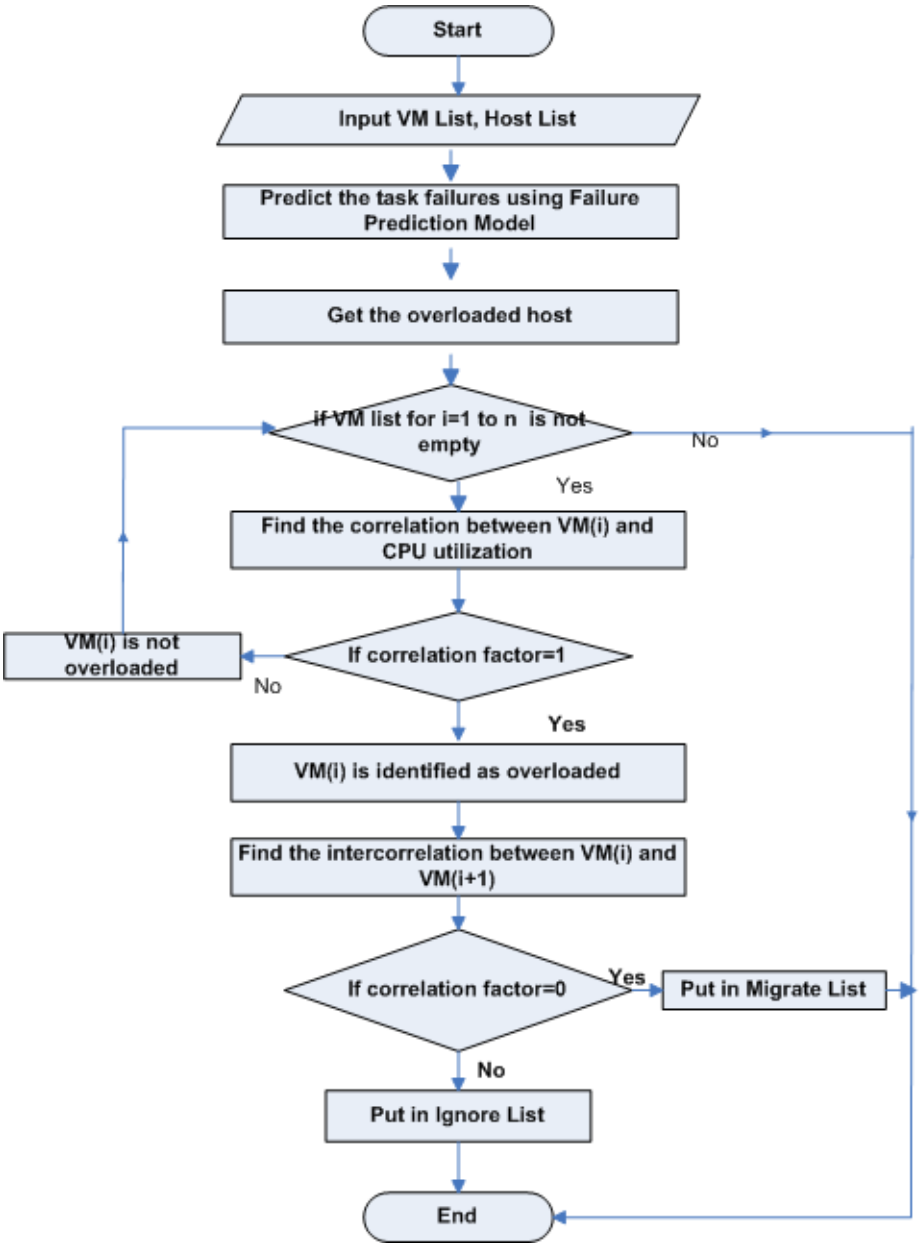


FIGURE 3.4: Flow Chart for Autonomic VM Migration

3.3.2 Autonomic VM migration Approach using Multiple Parameters

As the task has been predicted as failed or not failed using Failure Prediction Model through multiple resource utilization parameters, therefore, Inter-Correlation Coefficient approach has also been utilized to find the association between multiple resource utilization parameters (CPU, RAM, Bw, Disk I/O) and hosts. The proposed approach for fault tolerance during overutilized host has been discussed in Algorithm 2.

- If task failure is predicted due to overloading of the host, then proposed algorithm selects faulty VMs that have to be migrated from the overloaded host.
- This algorithm firstly finds the correlation between VM and resource utilization, the VM which has maximum correlation with resource utilization indicates that VM can be overloaded.
- Secondly, the proposed algorithm finds the inter-correlation factor between the overloaded VMs with other machines on the same host using multiple parameters. The overloaded VM can work with other machines in a group on the same host. Therefore, if the inter-correlation factor is one then put it into the 'ignore list' because these machines have strong association along with each other, otherwise, if the inter-correlation factor is zero that indicates these overloaded machines are independent of each other, then put that machine into the 'migrate list'.

Algorithm 2: Autonomic VM Migration for overutilized hosts

Input: Host List, VM list
Output: Migration List
Foreach Host in HostList(j),j= 1 to m do
If the tasks are predicted as failed using Naive Bayes and get the overloaded hosts
Then Foreach VM in VMList(i), i=1 to n do
find the correlation between VM(i) and Resource Utilization parameters
If correlation factor=1 then find inter- correlation of VM(i) with other VMs
If inter- correlation factor= 1 then put VM(i) into ignore list
Else put into migrate list
End If
End If
End For
End If
End For
Return Migration List of VM

After discussing the proposed algorithm for Autonomic VM Migration, the next section elaborates on the optimization of Cloud hosts using proposed approach.

TABLE 3.2: Inter-correlation between VMs on the same host

Virtual Machines	**VM1**	**VM2**	**VM3**	**VM4**
VM1	1	r (1,2)	r (1,3)	r (1,4)
VM2		1	r (2,3)	r (2,4)
VM3			1	r (3,4)
VM4				1

3.3.3 Optimization of Cloud Host

The overloaded or faulty VMs are shifted from the host but hosts are optimally utilized using CPU utilization parameter and multiple resource utilization parameters. Table 3.2 shows the inter-correlation of overloaded virtual machine with other machines. For example, there are four virtual machines $VM1$, $VM2$, $VM3$ and $VM4$ on the overloaded host, the correlation factor between same virtual machines is 1. The correlation factor between different VMs need to be evaluated using single and multiple resource utilization parameters. If Host 4 is predicted as overloaded, then VMs can be migrated from overloaded host using proposed approach.

Let $VM1$ have highest correlation factor with overall resource utilization which is calculated by the Eq.(3.16) where r is correlation coefficient, hence, $VM1$ is being considered as overloaded virtual machine from overloaded host. Then, inter-correlation factor r can be calculated by using above Eq.(3.17) between $VM1$ and other machines such as $VM2$,$VM3$ and $VM4$ on the same host.

Eq.(3.16)

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{3.16}$$

Then, the value can be computed using R^2_{1234} which is squared coefficient of correlation between predicted and practical variable. For example, $VM1$ is denoted as 1 supposed to be highly overloaded VM and finds the inter- correlation r with other machines $VM2$, $VM3$, $VM4$,here r_{12} indicates the correlation between VM1 and VM2 and r_{13} indicates the correlation between $VM1$ and $VM3$, similarly r_{14} indicates the correlation between $VM1$ and $VM4$,therefore R_2 has been calculated by using Eq.(3.17) correlation of $VM1$ with other machines such as $VM2$, $VM3$ and $VM4$ on the same host.

$$R^2_{1234} = \frac{r^2_{12} + r^2_{13} + r^2_{14} - 2(r_{12})(r_{13})(r_{14})(r_{234})}{1 - r^2_{234}} \tag{3.17}$$

Figure 3.5 shows the migration of virtual machine from overloaded host to another host by considering CPU utilization parameter. If $VM1$ has maximum correlation coefficient then inter-correlation between $VM1$ and other machines has been evaluated, if the inter-correlation coefficient is 0 then migrate the $VM1$ to another host which has currently CPU utilization lower than threshold value.

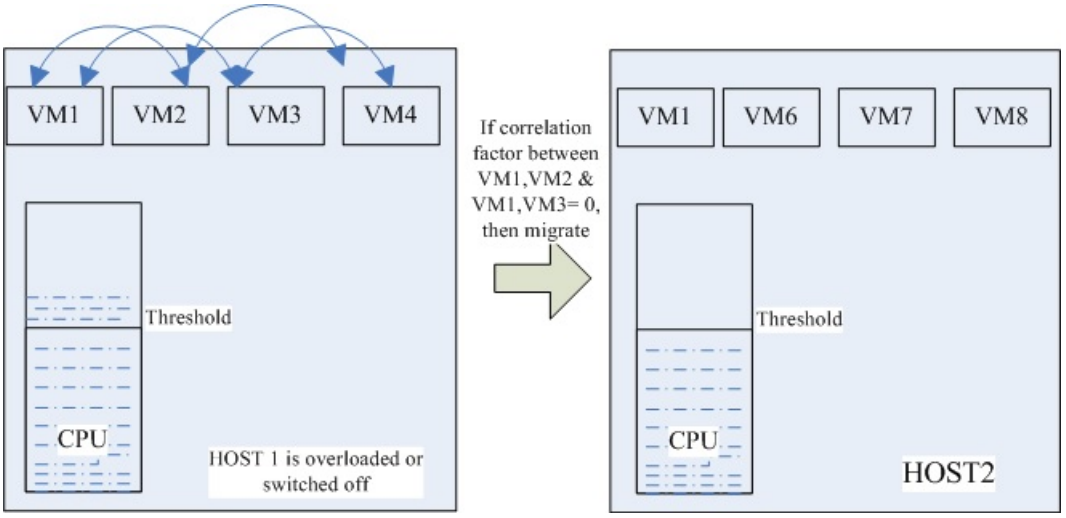


FIGURE 3.5: Optimization of Cloud Host using CPU Utilization

If $VM1$ is not strongly associated with other machines through multiple resource utilization parameters such as(CPU, RAM, Bw, Disk I/O), then it could not be migrated as shown in Figure 3.6. It illustrates that Host 4 is predicted as overutilized as the overall resource utilization of all the parameters is more than maximum threshold value, therefore, proposed VM migration approach migrates the virtual machine $VM1$ which is suitable for migration to another Host 3 which has the utilization below the maximum threshold value.

To validate the optimization of proposed autonomic VM migration approach, various evaluation parameters have been discussed in next section.

3.3.4 Evaluation Measures for Autonomic VM Migration Approach

These are the evaluation measures which have been used to measure the performance of proposed autonomic fault tolerant approach.

(i) *VM Execution Mean Time*: Reallocation Mean Time is average time which is denoted by $\bar{x}$ has been used to find the average time for reselecting or

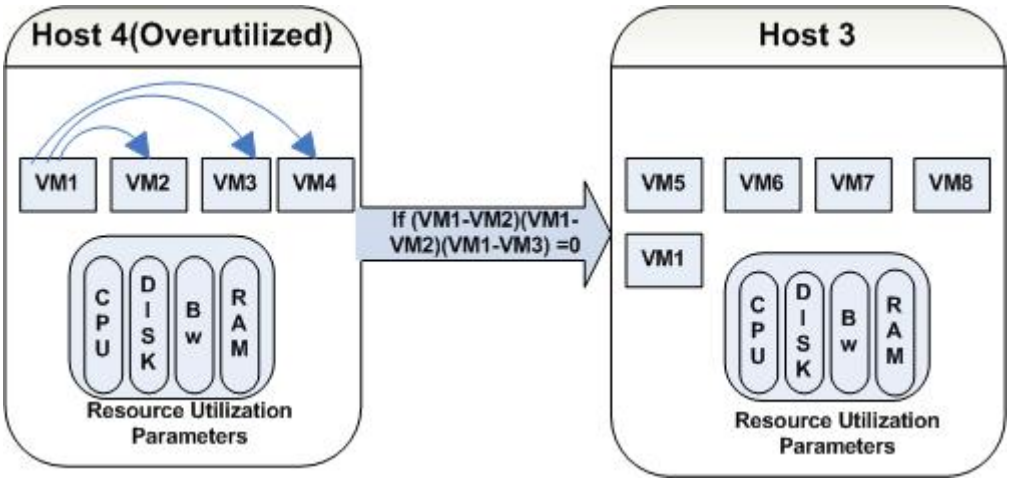


FIGURE 3.6: Optimization of Cloud Host using Multiple Parameters

migrating the VMs from one host to another as shown in Eq.(3.18). Here n is the total number of calculations.

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{3.18}$$

(ii) *VM Execution Standard Deviation Time*: Standard Deviation measures how much Reallocation time deviates from the mean is denoted by s as shown in Eq.(3.19).

$$s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2} \tag{3.19}$$

(iii) *Number of VM Migrations*: Number of VM migrations stipulates the count for how many times various VMs have migrated to other hosts. For the efficient VM migration approach there should be minimum number of VM migrations.

(iv) *SLAV*: The SLAV metric represents the CPU performance that has not been distributed to an application when requested that further results in performance degradation. For the efficient VM migration policy, SLAV should be minimum. It shows the agreement between the resource provider and consumers is violated as requested resource utilization is not given to the provider. The SLA violation Time per Active Host (SLVATAH) shows the percentage of time of active hosts that are overutilized and the Performance Degradation due to Migrations (PDDTM) describes the overall performance

degradation by VMs due to migrations. SLVATAH is defined by the Eq. (3.20) where n represents total number of hosts and T_{sj} indicates total time during host j has the utilization more than maximum threshold value that leads to further SLAV. T_{aj} describes total number of hosts j being currently in active state. PDDTM is defined in the Eq. (3.21) where m is the total number of virtual machines and P_{di} represents approximation of the performance degradation of VM_i due to migrations. P_{ci} is the total capacity requested by VM_i during runtime. Thus, the metric SLAV depends on the performance degradation due to overutilization of hosts and number of VM migrations which is described in the Eq. (3.22) [36].

$$SLAVTH = \frac{1}{n}\sum_{j=1}^{n}\frac{T_{sj}}{T_{aj}} \tag{3.20}$$

$$PDDTM = \frac{1}{m}\sum_{i=1}^{m}\frac{P_{di}}{P_{ci}} \tag{3.21}$$

$$SLAV = SLAVTH * PDDTM \tag{3.22}$$

3.4 Conclusion

This chapter discussed how task failures are handled by intelligent failure prediction models for scientific workflows. Classification based machine learning approaches such as Naive Bayes, ANN, LR and Random Forest have been used for implementing intelligent task failure model. The performance metrics have also been evaluated to select the most accurate machine learning approach. Further, the proposed VM migration approach is also detailed by incorporating failure prediction models with fault tolerant approach. Various performance metrics for fault tolerant approach have been evaluated to validate the performance of the proposed approach.

The next chapter presents the workflow scheduling approach for multiple scientific workflows and discusses in detail the usage of the proposed autonomic fault tolerant approach along with proposed workflow scheduling algorithm.

Chapter 4

Autonomic Fault Tolerant Scheduling Approach for Multiple Workflows

The previous chapter elaborated an intelligent failure prediction model and design of an autonomic fault tolerant approach for multiple scientific applications. As workflow scheduling is the key concept for the execution of performance motivated Cloud applications such as scientific workflows, for the execution of concurrent tasks in scientific workflows, Cloud provider has a need of efficient scheduling heuristics.

An autonomic fault tolerant scheduling approach has been formulated to provide the reliable and available Cloud services to the user. The approach has been used to schedule scientific applications in the form of workflows through autonomic fault tolerant behaviour. The performance of the proposed approach has been evaluated using execution time and makespan.

This chapter focuses on the design of a workflow scheduling algorithm for multiple scientific workflow applications. Firstly, Deadline Based Approach has been used to schedule multiple workflows, secondly, the workflows having maximum priority among multiple workflows has been scheduled first using proposed hybrid heuristics. The proposed autonomic fault tolerant technique has been used to migrate the VM automatically in case if task failurs occurr during scheduling.

4.1 Autonomic Fault Tolerant Scheduling

Autonomic Fault Tolerant Scheduling is now becoming mandatory for an execution of performance motivated Cloud applications such as scientific workflows. As the scientific applications entail large computing power which further requires large number of resources, hence, Clouds can be used to afford these resources whenever required. These applications are represented in the form of workflows so as to reduce the makespan and execution time. An efficient scheduling algorithms can have major impact on the performance of scientific workflow applications by utilizing Cloud services. Therefore, this section formulates an effort to focus on the research problem of designing a scheduling approach for multiple scientific workflow applications in fault tolerant manner. A hybrid heuristic has been proposed to schedule scientific workflows effectively and autonomic fault tolerant technique has been implemented using Virtual Machine (VM) migration approach that migrates the VMs automatically in case of task failure occurrence due to the overutilization of resources. Further, the proposed approach has been evaluated through scheduling factors using multiple scientific workflows.

4.1.1 Problem Statement and Solution to the Problem

This section discusses the problem statement for example workflow with existing scheduling heuristics and proposed hybrid approach for the solution of current problem along with VM migration approach.

(i) *Problem Statement*: Workflow tasks are the collection of instructions that can be executed on Cloud resources. Each Directed Acyclic Graph (DAG) is represented as a set of vertices has tasks T and edges E where $T = \{t_i, i = 1...n\}$ designate the set of tasks and E is a matrix defined on T that describes the operational preference rules. If $E_{i,i'} = 1$ means that t_i must be completed before $t_{i'}$ can start execution. Workflow example has been shown in Figure 4.1 that consists of a set of 18 tasks where T = $\{t_i, i = 1...18\}$ and E=$\{< t_R, t_0 >, < t_R, t_1 >, < t_R, t_2 >, < t_3, t_4 >, < t_3, t_5 >, < t_3, t_6 >, < t_3, t_7 >, < t_7, t_9 >, < t_7, t_{10} >, < t_7, t_{11} >, < t_7, t_{12} >, < t_{12}, t_{13} >, < t_{12}, t_{14} >, < t_{12}, t_{15} >, < t_{12}, t_{16} >, < t_{12}, t_{17} >\}$ and V is the set of virtual machines=$\{VM1, VM2, VM3\}$. Workflow scheduling efficiently maps the workflow tasks on Cloud resources to minimize the makespan and utilize the

resources efficiently as the existing algorithms do not consider the priority for the tasks which have the maximum number of child nodes. Therefore, scheduling heuristics such as FCFS, Min-Min, Max-Min, and MCT schedules the tasks in 8 execution cycles as detailed in Table 4.1. The output results for existing heuristic would be the same because same execution time has been considered for the entire tasks; hence, the existing algorithms would work similar to FCFS. For the first task schedule, existing schedulers schedule tasks 0, 1, and 2 on three VMs in parallel with X time to execute, and for second schedule, only task 3 can be scheduled; other tasks 4 and 5 could not be scheduled as its parent task has not completed its execution as yet. Similarly, for the third schedule, tasks 4, 5, and 6 would be allocated to three VMs; the fifth schedule and the seventh schedule behave akin to the same way as the third schedule. In the fourth schedule, task 7 and task 8 will be scheduled on two VMs and task 9 could not be assigned as its parent task does not complete the execution, and the eighth schedule assigns task 16 and task 17 to $VM1$ and $VM2$. Thus, in this way, resource's wastage and execution time would increase for the current example and the total execution time $8X$ will be required. Hence, there is a need to schedule the task effectively using efficient heuristics to avoid resource wastage and to reduce the makespan correspondingly.

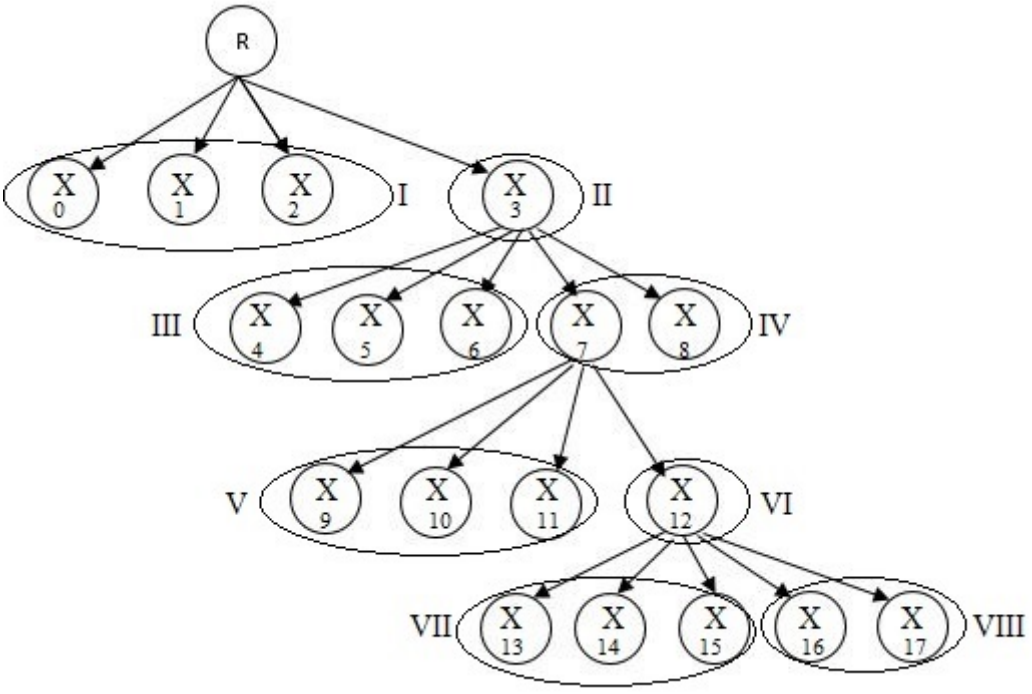


FIGURE 4.1: Workflow Example with Same Execution Time

TABLE 4.1: Scheduling Process of Existing Heuristics with 3 Virtual Machines

Scheduler Process	**VM1**	**VM2**	**VM3**	**Time Consumed**
Schedule 1	Task 0	Task 1	Task 2	X
Schedule 2	Task 3	– – –	– – –	X
Schedule 3	Task 4	Task 5	Task 6	X
Schedule 4	Task 7	Task 8	– – –	X
Schedule 5	Task 9	Task 10	Task 11	X
Schedule 6	Task 12	– – –	– – –	X
Schedule 7	Task 13	Task 14	Task 15	X
Schedule 8	Task 16	Task 17	– – –	X

(ii) *Hybrid Heuristic- Proposed Approach*: To optimally utilize the resources, an efficient scheduling approach has been proposed and named as hybrid heuristic that firstly schedules and executes that task which has the maximum number of child nodes; if two or more of the tasks have the same number of child nodes, then Min-Min heuristic would be used and if the two tasks have the same execution time, then FCFS will be considered. The working of hybrid scheduler has been indicated in Table 4.2 for the same example workflow of Figure 4.1. During the first schedule, task 3, task 0, and task 1 are allocated to $VM1$, $VM2$, and $VM3$ because task 3 has the maximum number of child nodes; where task 0 and task 1 have the same priority, then FCFS heuristic would be used to allocate the resources. Min-Min heuristic will work similar to FCFS because the same execution time is considered. Same process repeats for second, third, fourth, fifth, and sixth schedules, and the proposed heuristic would increase the scheduling efficiency of workflows through proper utilization of resources and minimize the execution time and makespan. Thus, the example workflow would be executed completely in $6X$ time, but during the execution, if the task failure occurs due to the overutilization of allocated VM, then VM migration approach also needs to be implemented along with the proposed workflow scheduling heuristic.

TABLE 4.2: Scheduling Process with Proposed Approach

Scheduler Process	**VM1**	**VM2**	**VM3**	**Time Consumed**
Schedule 1	Task 3	Task 0	Task 1	X
Schedule 2	Task 7	Task 2	Task 4	X
Schedule 3	Task 5	Task 6	Task 8	X
Schedule 4	Task 12	Task 9	Task 10	X
Schedule 5	Task 11	Task 13	Task 14	X
Schedule 6	Task 15	Task 16	Task 17	X

Hence, hybrid heuristic can be used to optimize the performance of Cloud based scientific applications. To implement the scheduling approach for multiple workflows in Cloud, design methodology used for the proposed approach has been defined in next section.

4.1.2 Proposed Methodology

The general flow of the proposed Autonomic Fault Tolerant Scheduling Algorithm has been described in Figure 4.2.

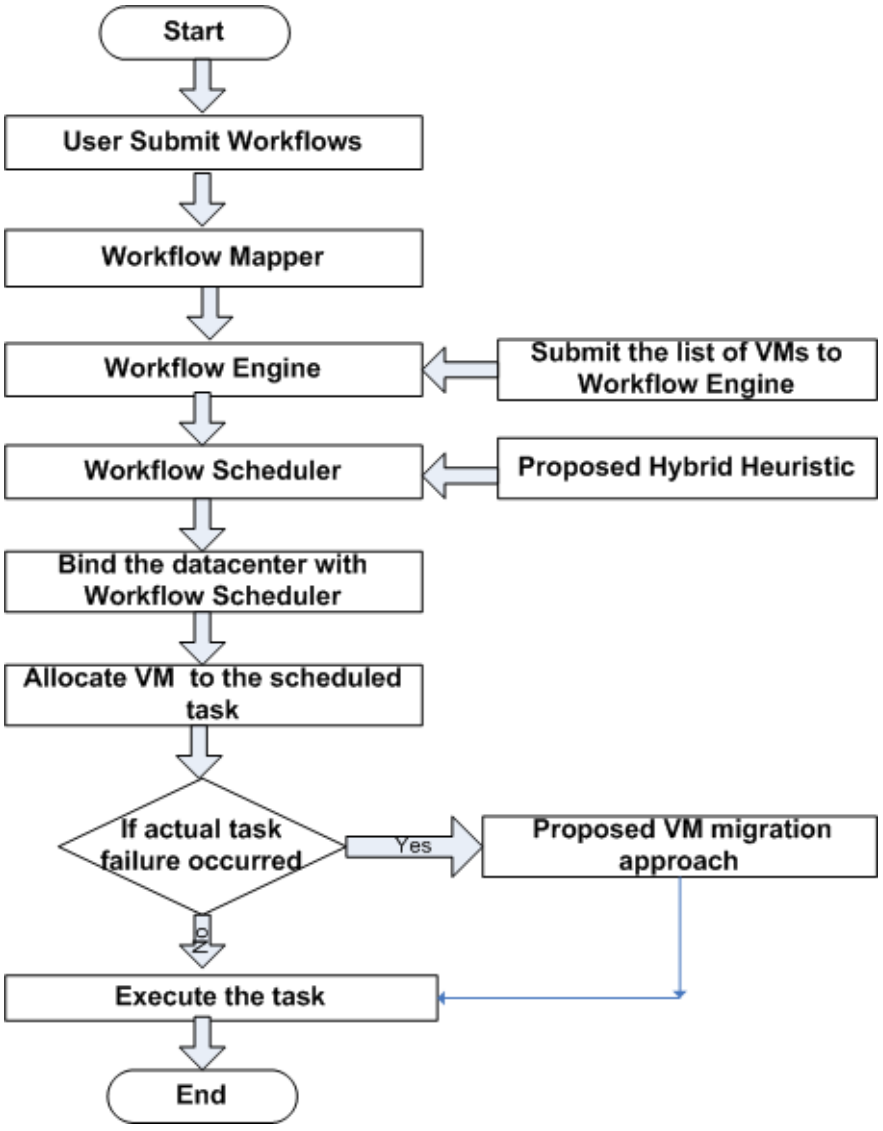


FIGURE 4.2: Flow Diagram

- Firstly, the Cloud user submits the multiple applications as workflows like Montage, Cybershake, Inspiral and SIPHT where Montage workflow contains

25 jobs and Cybershake, Sipht and Inspiral contain 30 jobs. Each job can further contain hundered to thousandths of tasks.

- A workflow mapper maps the abstract workflow into concrete workflow by importing the Directed Acyclic Graph (DAG) files. It also contains the related information like size of the workflow tasks, number of tasks in one job, and dependency between these tasks which are further formatted in the Extensible Markup Language (XML) file. By getting this information, workflow mapper defines a list of tasks to be executed on Cloud resources and a Cloud user creates the list of virtual machines and submits to the workflow engine for execution of these tasks on Cloud resources.

- A DAGMan workflow engine manages the data dependencies between the workflow jobs. As the job can only be executed when all of its parent jobs have successfully completed. So, workflow engine schedules only the jobs which are ready to execute the tasks and submits these jobs to the scheduler.

- Workflow scheduler schedules the jobs based on Earliest Deadline First (EDF) and proposed hybrid heuristic. As existing scheduling heuristics do not prioritize the workflow tasks based on hybrid heuristic which first schedules the tasks based on Max-Child, then Min-Min heuristic and FCFS heuristic so as to reduce the makespan and to further increase the resource utilization. One job can contain million of tasks, these tasks have been scheduled based on task scheduling algorithm.

- Workflow engine binds the datacenter with a workflow scheduler. Datacenter allocates the list of tasks to the virtual machines created by the user on Cloud host to accomplish the workflow execution.

- If the actual task failure occurs due to overutilized resource which has already been predicted through intelligent failure prediction model and proposed autonomic fault tolerant approach migrates the virtual machines to another host. But if no failure occurs then task would be executed on the allocated Cloud resource.

To implement the proposed design for scheduling, Cloud model is used along with various scheduling factors which serve as the performance metrics of workflow scheduling described in the next section.

4.1.3 Cloud System Model

Each Workflow is defined by using DAG which is the set of tasks $T = \{t_i; wherei = 1...n\}$, a task which has no precedents is called an entry task, and a task without any successors is an exit task. For the Cloud resources, $H = \{h_1, h_2, ..., h_l\}$ is the Cloud host list which has l total hosts and $V = \{v_1, v_2, ..., v_m\}$ defined as the collection of m Virtual Machines. To demonstrate the task scheduling problem evidently, in this section, the scheduling factors have been defined below.

(i) *Earliest Deadline First*: To schedule multiple workflows on Cloud resources, Earliest Deadline First(EDF) factor is used which is based on the deadline D_w= $\{D_{w1}, D_{w2}, ..., D_{wn}\}$ of multiple workflows where D_{w1} indicates workflow 1, D_{w2} as second workflow and D_{wn} represents n_{th} wokflow. Multiple workflows have been executed concurrently after assigning each workflows based on Deadline (D) to different virtual machines in the Cloud. The workflow which has EDF has been assigned first to Cloud resources then other workflows based on their deadline D have been allocated to another VMs parallely. In the proposed approach, deadline of each workflow has been shown in Table 4.3 which is statically defined by the Cloud user. For example, if the montage has earliest deadline of 10 hours, then other workflows such as Cybershake, Inspiral and SIPHT. Then, montage workflow would be allocated to VMs first, then other workflows such as Cybershake, Inspiral and SIPHT would be assigned to other free VMs on to the different or the same hosts from the list of l hosts. The main goal of the EDF to complete the execution of workflow on allocated virtual machine before the deadline which is defined by the Cloud user. Henceforth, Earliest Completion Time (ECT) of workflow wi on allocated v_j should be less than the deadline of that particular workflow.

$$ECT(wi, v_j) < D_{wi} \tag{4.1}$$

TABLE 4.3: Workflow Input Parameters

Workflow	**Number of Jobs**	**Period(hours)**
Montage	25	10
Cybershake	30	15
SIPHT	30	25
Inspiral	30	20

(ii) *Mean Execution Time*: Execution time is defined as how much time has been taken by Task t_i on the allocated virtual machine v_i, the mean of the estimated execution time (MET)is defined by the Eq:4.2.

$$\bar{ET} = \frac{\sum_{i=1}^{n} ET(t_i, v_i)}{n} \tag{4.2}$$

(iii) *Standard deviation of Mean Execution Time*: Standard Deviation is used to evaluate how much time deviates from the mean is denoted by s as shown in Eq.(4.3) where n is the total number of tasks.

$$s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(ET - \bar{ET})^2} \tag{4.3}$$

(iv) *Expected Completion Time*: Expected Completion Time time(ECT) is defined by the sum of Expected start time(EST) and Execution Time(ET) defined by the Eq.4.4

$$ECT(t_i, v_i) = EST(t_i, v_i) + ET(t_i, v_i) \tag{4.4}$$

(v) *Makespan*: Makespan(M) is calculated as the response time of a whole workflow, which is equal to the difference between the submission time of the entry task in the workflow and the output arrival time of the exit task in that workflow as defined in the Eq. 4.5

$$M = AFT(t_{exit}, v_i) - EST(t_{entry}, v_i) \tag{4.5}$$

As the Estimated Start Time for entry task(EST(t_{entry}, v_i) is assumed to be zero, hence the Eq. 4.5 can be written as shown in Eq. 4.6

$$M = AFT(t_{exit}, v_i) \tag{4.6}$$

After discussing the Cloud model and scheduling factors, the next section details the proposed workflow scheduling Algorithm.

4.1.4 Proposed Autonomic Fault Tolerant Scheduling for Multiple Workflows

Autonomic Fault Tolerant Scheduling approach for scientific workflows has been implemented by prioritizing the tasks of a workflow using hybrid heuristic. Then, broker can have better control on scheduling for independent tasks whereas hybrid heuristic merges the features of Max- Child, Min-Min and FCFS heuristic. As Workflow Scheduling has been utilized to schedule the tasks first which have maximum children but if there are more than two tasks having equal children then Min-Min heuristic would be used and if the tasks having same execution time then FCFS heuristic would be utilized. Further, during the execution of workflow tasks on VMs, task execution can be failed due to overutilization of resources such as Bandwidth, RAM, CPU, and Network I/O. Although, the Cloud provider intends to assure the reliability of the environment, then autonomic fault tolerant approach has been implemented along with workflow scheduling. (The autonomic fault tolerant approach has already been discussed in Chapter 3. The fault tolerant approach has been used to migrate the faulty machines to another host automatically.) To implement scientific workflow applications in Cloud, deadline based workflow scheduling has been utilized for concurrent execution of multiple workflows which is being discussed in next section.

4.1.4.1 Concurrent Execution of Multiple Workflows using Earliest Deadline First

For implementing deadline based scheduling, the daedline D has been given by the Cloud consumer. For concurrent scheduling of multiple workflows, the workflows has been prioritized by first using Algorithm 1. Multiple workflows are submitted to the Cloud along with the detail of deadline for each workflow defined by Cloud user. Each workflow has been sorted in increasing order of their deadline and the workflow which has Earliest Deadline First (EDF) would be scheduled first on the VMs and other workflows would be scheduled concurrently on other free resources based on their D.

Algorithm 1: Earliest Deadline First Algorithm for Multiple Workflows
PROCEDURE: Workflow Submit
Input: Workflow List, Deadline
Output: Sorted List of Workflows
Begin
For each workflow W_i where i=1 to n
Find the deadline D defined by user
Sort the Workflows in ascending order of D to find EDF
End For
Schedule the sorted workflows on different Virtual Machines (VMs)
Until worlkflow list is not empty
Call Algorithm 2 for poposed heuristic
Return the list of sorted workflows

4.1.4.2 Proposed Hybrid Heuristic for Workflow Scheduling

The Algorithm 2 discusses the proposed hybrid scheduling heuristic which utilizes the features of Max-Child , Min-Min and FCFS. The Min-Min heuristic schedules the set of independent tasks by computing ECT of each task on the Cloud resources. A task which has minimum ECT would be selected to schedule on to Cloud resources. It assigns the task to a resource which takes minimum time to complete. FCFS heuristic schedules the task which came first in the ready queue where Max-Child heuristic schedules the tasks having maximum child nodes, the child task can not complete its execution before the completion of parent task, hence the Max-Child is prioritized first, then Min-Min and FCFS. Cloud user will submit the workflow along with the information of task list, dependency list and list of Cloud resources which is VM list and Host List and Deadline for each workflow. The tasks have been scheduled in the ready queue according to the availability of resources. From ready task list, the task which has maximum siblings scheduled first and allocate the resource which takes minimum execution time for the selected task as discussed in Algorithm 3. If the two tasks having same number of siblings then Min-Min is utilized to schedule else FCFS is utilized to schedule. After scheduling the task on the resource, ready list is updated until unscheduled task list is not empty.

Algorithm 2: Hybrid Scheduling Heuristic for Workflows

PROCEDURE: Workflow Submit
Input: Task List, Task Dependency List, Resources(VM list, Host List)
Output: List of Scheduled Tasks
Begin
For all $t \epsilon T$ from the workflow
For all $v \epsilon V$ from list of resources
Set the Maximum priority to the task t using Max-Child heuristic
$N \leftarrow Max_Successors(t)$
If two or more tasks having same siblings
Use Min-Min Heursitic
$N \leftarrow$ MinECT(t,v)
Else Use FCFS heuristic
$N \leftarrow$ FCFS(t,v)
End If
Schedule Task N onto Cloud resources using Algorithm 3
Update the ready Task List
End For
End For
End

The next Algorithm 3 allocates the resources which takes minimum time. During the VM allocation to task, if the task fails on the resource, then proposed fault tolerant technique is used to implement scheduling, which further enhances the reliability of scientific applications.

Algorithm 3: Resource Allocation

Input: Scheduled Task Lists, Resource list (Number of Resources)

Begin i=0
While task list T is not empty do
Select v from the List of V which takes minimum time
Schedule the task t on to the resource and execute it
If the t_{Status} = Failure($v_{overutilized}$)
Use autonomic Fault Tolerant Technique
Else execute the task on the allocated v
End If
i=i+1
End While
End

After the detail description of the proposed scheduling algorithm, performance has been evaluated using multiple scientific workflow applications which is discussed below in next section.

4.1.5 Performance Evaluation

To evaluate the performance of proposed scheduling algorithm, multiple scientific workflow applications has been detailed below along with their node description and execution time characterstics:

(i) ***Case 1***- *Montage Scientific Workflow Application*: Montage is a Cloud-capable astronomical and high energy physics application that has been used to reproject the input images, resolve their backgrounds, and mosaic and compress them into a single image. It can provide the better results for fast compressed hybrid colored images [179]. The size of a Montage workflow depends upon the area of the sky covered by the output. In the example workflow, Montage Workflow with 25 tasks and 9 levels has been shown in Figure 4.3. The details of each level are shown in Table 4.4 and execution time characterstics are described in Table 4.5. For the first schedule, task 0, task 1, task 2, task 3 and task 4 are in queue and there are four virtual machines which can be allocated to these tasks. As the task 1 has maximum number of child nodes, hybrid heuristic first schedules task 1 to $VM0$, then task 0 to $VM1$, then task 2, task 3, task 4 have prioritized using min-min

heuristic, therefore, task 3 and task 4 would be allocated to $VM2$ and $VM3$. In next cycle, task 2, task 13, task 9, task 10 would be scheduled on four Virtual machines, because task 8, task 11 can not be scheduled as its parent task 2 is not completed its execution yet now. Similarly, for other tasks, this process is repeated again and again. By using Min-Min heuristic only, task 0, task 3, task 4 and task 2 will be scheduled on resources, in next schedule, task 1, task 13 and task 12 would be assigned, as task 1 has not completed its execution, hence, task 5, task 6, task 7, task 8, task 9, task 10, task 11 could not be assigned any resources. Therefore, one VM would be free and FCFS and MCT can also schedule task 1, but hybrid heuristic always performs enhanced or alike performance within best heuristic for scientific workflows which have more dependent nodes.

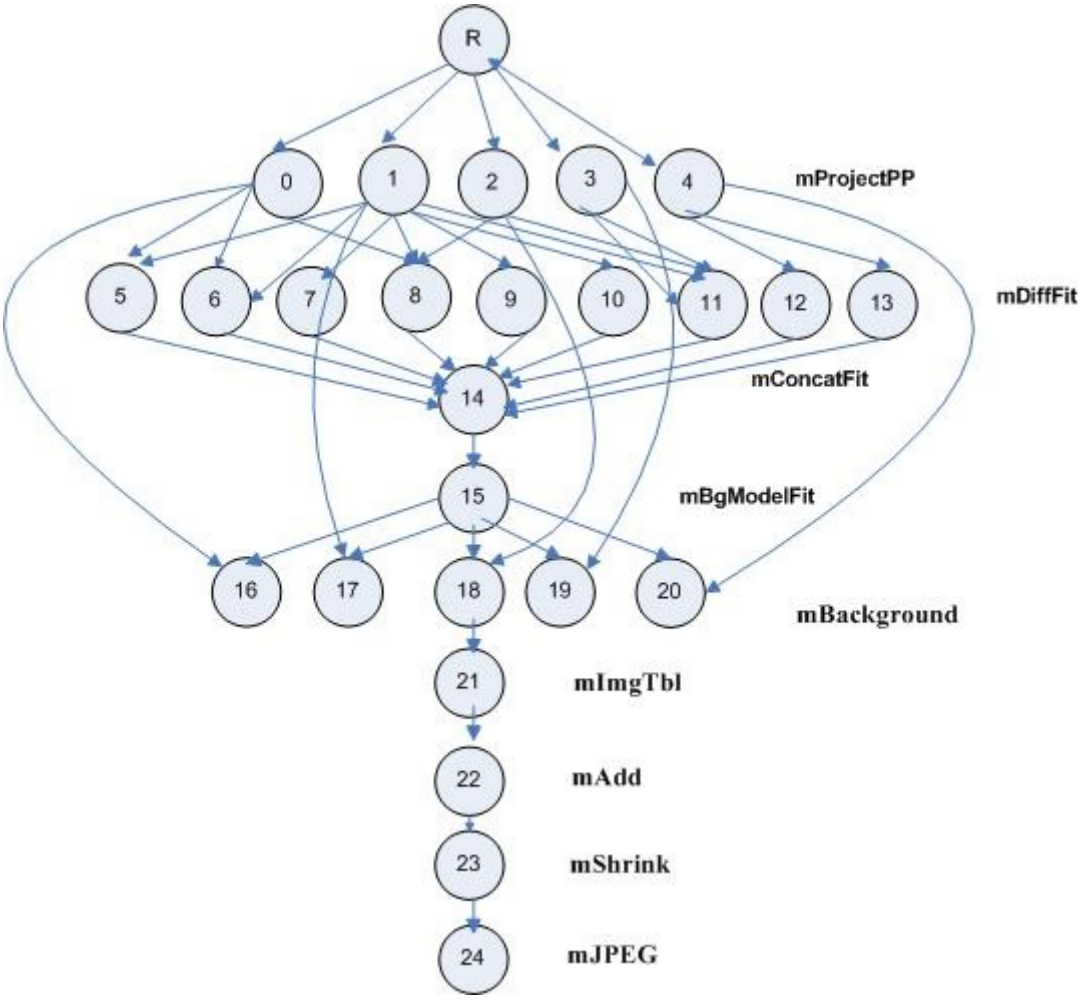


FIGURE 4.3: Montage Workflow with 25 tasks

(ii) ***Case 2***- *Cybershake Scientific Workflow Application*: Cybershake workflow has been developed by Southern California Earthquake Center (SECE)

TABLE 4.4: Montage Workflow Level-Description

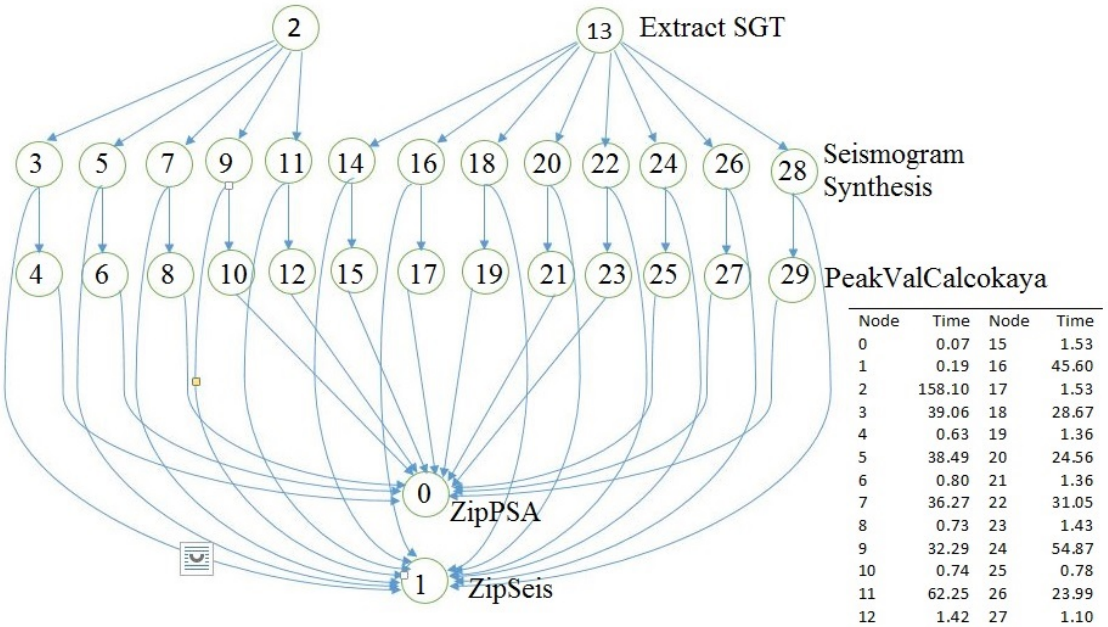
Level No	Transformation Name	Description
Level 1	mProjectPP	Fast re-projection algorithm used to re-project the image in a header file
Level 2	mDiffFit	Finds the difference between overlapping images
Level 3	mConcatFit	Combines the mDiffFit jobs into a single image file
Level 4	mBgModelFit	Models the sky background with plane fit parameters.
Level 5	mBackground	Resolves the background in a single image.
Level 6	mImgTbl	Gets the geometry information and stores in an image metadata table.
Level 7	mAdd	Adds a set of reprojected images to produce a mosaic.
Level 8	mShrink	Uses for image compression and enhancement.
Level 9	mJPEG	Creates a final mosaic in JPEG format.

TABLE 4.5: Execution Time Details for Montage Workflows

T.No.	ET	T.No.	ET	T.No.	ET	T.No.	ET	T.No.	ET
0	13.39	5	10.59	10	10.51	15	1.42	20	10.76
1	13.83	6	10.88	11	10.51	16	10.39	21	1.39
2	13.80	7	10.88	12	10.62	17	10.64	22	3.03
3	13.60	8	10.81	13	10.37	18	10.83	23	3.86
4	13.78	9	10.49	14	0.72	19	10.93	24	0.45

which is utilized to discover earthquake hazards by recognizing the earthquake raptures having moment magnitude. It also uses Probabilistic Seismic Hazard Analysis (PSHA) technique to illustrate the earthquake hazards and calculates the faults through SGTs (Strain Green Tensors) function, then it generates the ground motion of each rapture variations on the faults. Cybershake workflow has 5 levels as shown in Figure 4.4. The execution time of each job is also shown in the attached index along with cybershake workflow. In collaboration with the SGT data, estimated future faults rapture is also calculated along with the variations in these raptures. The details of each level have been presented in Table 4.6. Due to large number of files on these levels ExtractSGT and ZipPSA, they consume a lot of time to extract and compress the data respectively.

(iii) ***Case 3***- *SIPHT Scientific Workflow Application*: A small SIPHT workflow is an bioinformatics project that explores for small, untranslated RNAs (sRNAs) as described in Figure 4.5. SIPHT workflows have almost identical structure which are composed of smaller, independent workflows. The



Node	Time	Node	Time
0	0.07	15	1.53
1	0.19	16	45.60
2	158.10	17	1.53
3	39.06	18	28.67
4	0.63	19	1.36
5	38.49	20	24.56
6	0.80	21	1.36
7	36.27	22	31.05
8	0.73	23	1.43
9	32.29	24	54.87
10	0.74	25	0.78
11	62.25	26	23.99
12	1.42	27	1.10
13	96.91	28	26.46
14	47.44	29	0.85

FIGURE 4.4: Cybershake Workflow

TABLE 4.6: Cybershake Workflow Level-Description

Level No	**Transformation Name**	**Description**
Level 1	ExtractSGT	Extract the SGT files along with partitions of the data.
Level 2	Seismorgram Synthesis	Creates seismorgram synthesis for each rapture variation that is computed at level 1.
Level 3	PeakVal Calcokaya	Computes peak value of intensity for every job.
Level 4	ZipPSA	Accumulates the values of peak intensity with synthetic seismorgram respectively.
Level 5	ZipSeis	Compresses all the files in the archival form.

description of the workflow jobs has been detailed in Table 4.7.

(iv) ***Case 4***- *LIGO's Inspiral Scientific Workflow Application*: The LIGO Inspiral analysis workflow identifies the data from the binary neutron stars and black holes. This workflow is very complex and is composed of several sub-workflows is defined in Figure 4.6. The detailed description of each job has been discussed in Table 4.8

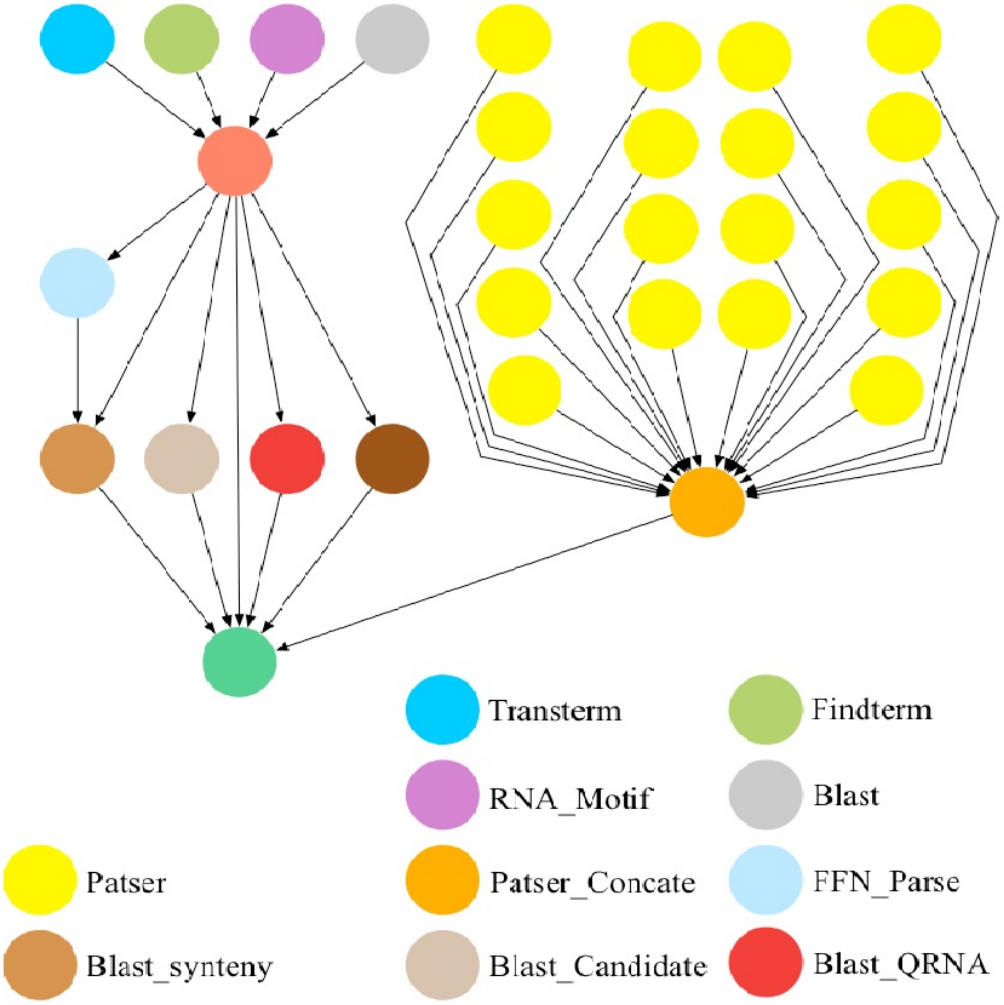


FIGURE 4.5: SIPHT Workflow[113]

TABLE 4.7: SIPHT Workflow Job Level-Description

Job No	Transformation Name	Description
Job 1	Patser	Scan sequences.
Job 2	Patserconcate	Concatenates Patser jobs
Job 3	Transterm	Searches for transcription terminators.
Job 4	Findterm	Searches for transcription terminators.
Job 5	RNAMotif	Searches for transcription terminators.
Job 6	Blast	compares different combinations of sequences.
Job 7	SRNA	Performs Prediction.
Job 8	FFNparse	Parses
Job 9	Blastsynteny	Inputs the data
Job 10	Blastcandidate	Searches
Job 11	BlastQRNA	Writes and read input data
Job 12	Blastparalogues	Searches
Job 13	SRNAannotate	Annonates the candidate SRNA

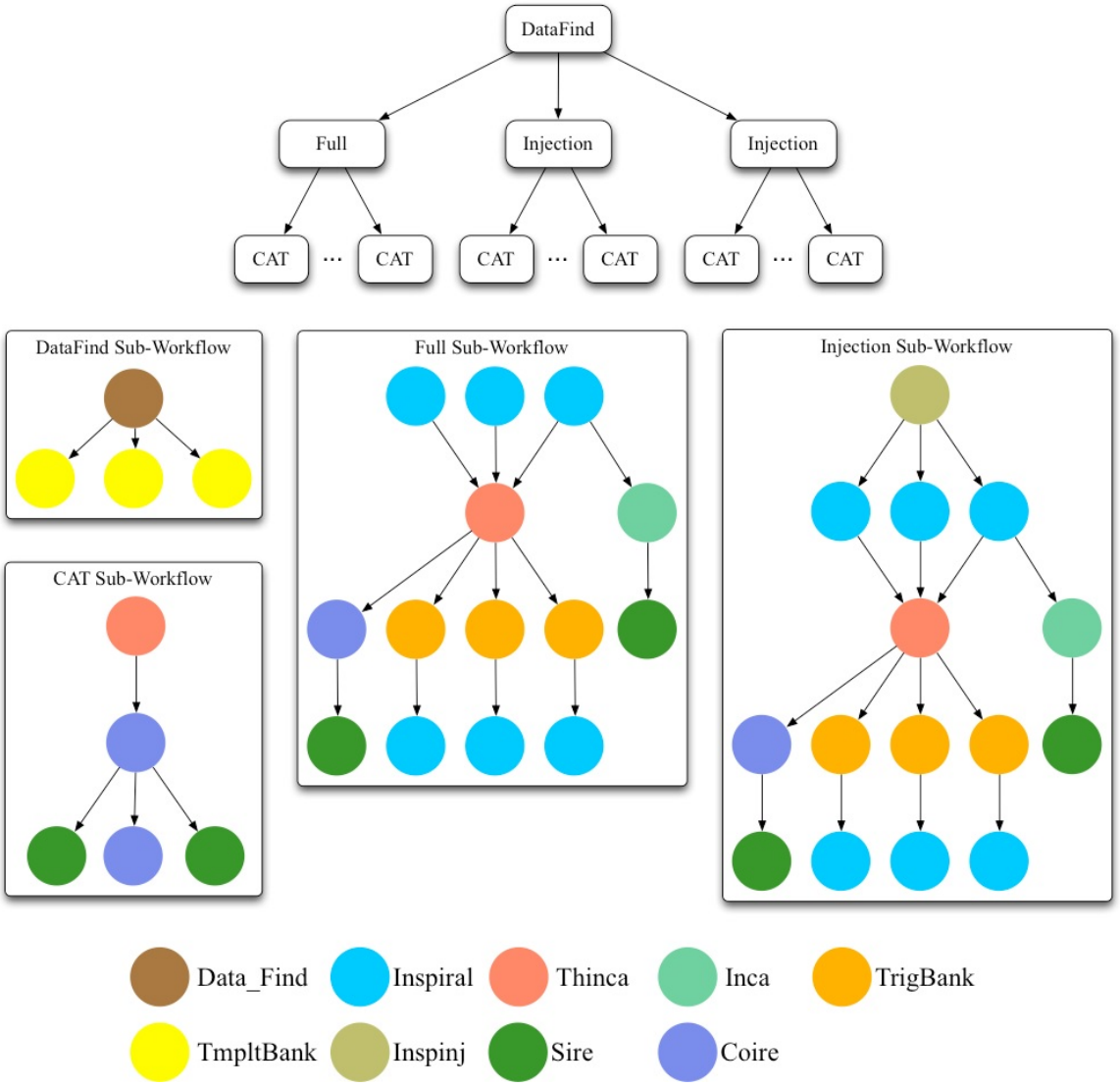


FIGURE 4.6: LIGO Inspiral Workflow[113]

TABLE 4.8: LIGO Workflow Job Level-Description

Job No	**Transformation Name**	**Description**
Job 1	TmpltBank	Identifies the continuous family of waveforms.
Job 2	Inspiral	Matches filtering code
Job 3	Thinca	Test for consistency.
Job 4	Inca	Searches for transcription terminators.
Job 5	DataFind	Searches data.
Job 6	Inspinj	low utilization
Job 7	TrigBank	Generates template banks out of the triggers.
Job 8	Sire	Parses
Job 9	Coire	Inputs large amount of data

4.2 Conclusion

This chapter discussed the problem statement for the need of hybrid scheduling heuristic. Further, this chapter endows with an effective approach of autonomic fault tolerant workflow scheduling through proposed hybrid heuristic along with the usage of proposed autonomic fault tolerant approach. The scheduling parameters have also been defined to validate the proposed approach. The performance evaluation for multiple workflows such as Montage, Cybershake, Inspiral and SIPHT have also been detailed with their job level description.

The next chapter discusses the experimental results obtained by validating the proposed approach. Firstly, the results of intelligent failure prediction model and autonomic fault tolerant technique have been detailed. Then, Autonomic Fault Tolerant Workflow Scheduling has been validated through evaluated performance metrics.

Chapter 5

Experimental Results and Discussion

The previous chapters discussed failure prediction model along with autonomic fault tolerant approach and also elucidated the proposed workflow scheduling algorithm. In order to prove the effectiveness of the proposed work in terms of various evaluated metrics, the experimental setup is required for simulation as well as on actual Cloud test bed.

This chapter is targeted towards the verification and validation of the proposed autonomic fault tolerant scheduling through the case study of multiple scientific workflow applications. This chapter highlights the implementation of the proposed work at three stages.

At first, an intelligent task failure prediction model has been evaluated by analysing the multiple workflow data using machine learning approaches. The efficient failure prediction model based on the maximum accuracy is further utilized for implementing an autonomic fault tolerant technique. Then, autonomic fault tolerance through VM migration approach has also been validated using single and multiple resource utilization parameters. Further, it summarizes the results obtained after implementing workflow scheduling algorithm along with proposed autonomic fault tolerant technique. The experimental results clearly validate the enhancements obtained in the performance of multiple scientific workflow applications in terms of makespan, mean execution time and standard deviation of mean execution time.

5.1 Experimental Results

The experimental results have been verified and categorized into the following sections.

- Intelligent Failure Prediction Model
- Autonomic VM Migration Approach
- Autonomic Fault Tolerant Workflow Scheduling Algorithm

5.2 Experimental Results: Intelligent Failure Prediction Model

The experimental results have been evaluated to measure the performance of the proposed intelligent task failure prediction model by implementing classification based machine learning approaches such as Naive Bayes, ANN, LR and Random Forest. Various simulators like WorkflowSim, CloudSim and Weka have been used for implementing predicted task failure module which are defined in Figure 5.1 whereas CloudTest bed has been used for verifying actual task failures. WorkflowSim [87] classes have been employed to execute scientific workflow applications like Montage, Cybershake, Inspiral and SIPHT. CloudSim [84] classes have been utilized for storing the log files and WEKA [114] is used to implement machine learning approaches using the data of scientific workflow applications. The evaluated results have been validated after running the scientific workflow applications on Cloud testbed using Pegasus and Amazon EC2 that provides a large selection of instance types such as CPU, memory, storage and networking capacity to deploy and execute scientific workflows. The detailed description of the experimental setup is discussed below :-

- Workflow Sim 1.0 has been used along with CloudSim that considers the scientific workflow applications and failure patterns with optimization techniques. It also chains simulation of both the static type and the dynamic type schedulers. Scientific workflow applications have been designed using XML schema and further the results have been analysed through Workflow Engine. It also provides the task clustering and fault tolerance. CloudSim

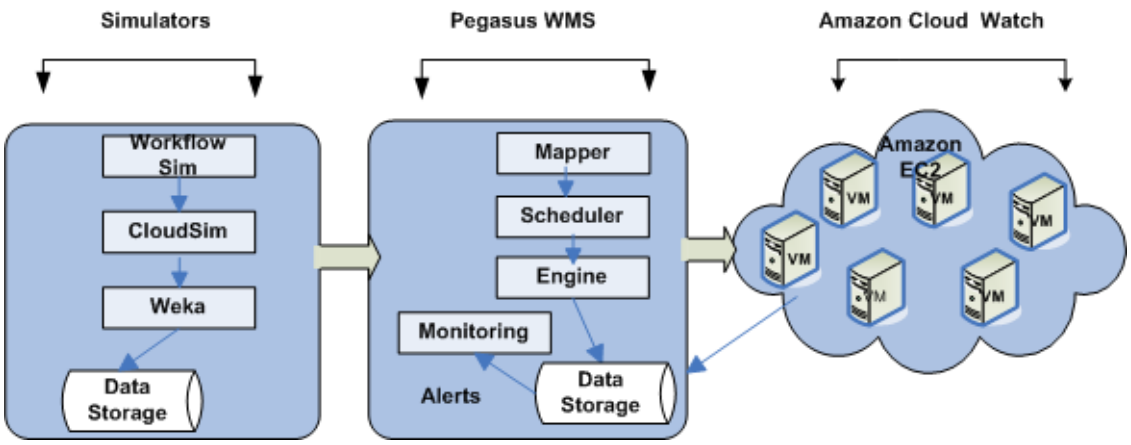


FIGURE 5.1: Testbed Design

and WorkflowSim have been used to analyse and collect the failure data of multiple scientific workflow applications.

- Cloud Sim 4.0 extends the features of GridSim that allows modeling and simulating the environment of Cloud using Datacenters, Hosts, Virtual Machines, and Cloudlets etc. To estimate the usefulness of the proposed model for predicted failures, CloudSim simulator has been used that supports modeling and simulating one or more VMs on a data center but it does not provide support for multi-threads. Hence, WorkflowSim has been used along with CloudSim.

- Weka is used to run data mining algorithms, in the current work, Weka has been utilized for data mining operations such as data extraction and for getting prediction based results. The data of predicted task failures for multiple workflows has been stored along with the values of their performance metrics.

- Pegasus -Workflow Management System (WMS) is used to validate the experimental results that map and execute the abstract workflows in a Cloud environment. Pegasus has a set of components such as the Pegasus Mapper, Workflow Scheduer and Execution Engine that are used to run and manage various workflow-based applications in different environment, including desktops, clusters, grids, and now Clouds. The workflow execution details are stored into a data base where the monitoring is done to analyse the workflow tasks and show the alerts. Pegasus bridges the scientific domain and the execution environment by automatically mapping high-level workflow descriptions onto Cloud resources. Pegasus mapper maps the abstract

workflows to concrete workflows. Then, the concrete workflow is further passed to scheduler for the execution on distributed resources. After the workflow finishes execution, this workflow can be deployed on its compatible Cloud platforms such as Amazon EC2, Nimbus, Open Nebula and Hadoop etc. The Future Grid account portal can be used to create an account on these Cloud Platforms. In the current work, Amazon EC2 is used as Cloud platform to deploy multiple scientific workflow applications.

- Amazon EC2 has been utilized for c3. xlarge instance type where c3 instances are the most recent generation of compute-optimized instances that provide highest performing processors at the lowest price. It is equipped with two quad core and 7.5 GB RAM, Intel Xeon E5-2680 v2, 1680 GB local storage and running Pegasus 4.2 with CentOS 6.0 on Virtual Box. Amazon Cloud Watch is utilized to accumulate the resource utilization metrics and log of task failure files.

After the detailed description about the requirement of these tools, the experimental results have been categorized into four steps shown below:

- Data Collection and Extraction
- Performance Validation
- Performance Measures using SEOM
- Comparison of Proposed Model with Existing Model

5.2.1 Data Collection and Extraction

Several attributes containing task failure information have been gathered using WorkflowSim and CloudSim classes. Principal Component Analysis (PCA) has been applied to reduce the dimensions for nine attributes that are required for implementing an intelligent failure prediction model. Each attribute along with its description is defined in Table 5.1. Resource utilization parameters have been evaluated using threshold value of CPU utilization, Bw Utilization, RAM and Disk utilization. The threshold value is selected which is based on the previous history of task failure due to over-utilization of VMs. If the utilization parameter has value more than threshold then the status would be classified as Task Failure otherwise No Failure. Task id indicates the task id that fails on the resource utilization and

TABLE 5.1: Attribute Description

S.No	Attribute Name	Description
1	Task id	Id of Task
2	VM id	Id of VM
3	Datacenter id	Id of datacenter
4	CPU utilization	Utilization of CPU (%)
5	Bw Utilization	Utilization of Bandwidth (%)
6	RAM utilization	Utilization of RAM (%)
7	Disk utilization	Utilization of Disk (%)
8	Depth	Level of task
9	Status	Failed or Not Failed

TABLE 5.2: Comparison of Performance Measures using Percentage Split

Metric	Naive Bayes	Random Forest	LR	ANN
Sensitivity	0.935	0.891	0.848	0.913
Specificity	0.849	0.417	0.127	0.42
Recall	0.935	0.891	0.848	0.913
Precision	0.94	0.876	0.754	0.921
RMSE	19	27	35.16	29.94
MAPE	29.63	72.0051	64.023	57.041
F-Measure	0.937	0.875	0.798	0.893
ROC	0.983	0.975	0.661	0.779

VM id signifies the VM number on which a task has been failed and similarly datacenter id identifies the datacenter where the task would fail or succeed. As there are multiple levels in the workflows, therefore, the depth specifies the level of tasks in workflows.

5.2.2 Performance Validation

Table 5.2 summarizes the results of the evaluated performance parameters using percentage split where 34 percentage data is used as test data and 66 percentage is used for training data. It shows the experimental results of various performance measures such as Sensitivity, Specificity , Recall, Precision, Root Mean Square Error (RMSE), Mean Absolute Percentage Error (MAPE), F-Measure and ROC. The experimental results show that Naive Bayes machine learning approach using proposed model performs best on the task failures predictions of scientific workflow data and ANN also performs better than Random Forest and LR. The results achieved are sketched out in Figure 5.2, 5.3, 5.4 and 5.5 using the following cases:

Case 1: Sensitivity and Specificity

It can be inferred from the Figure 5.2 that Naive Bayes has highest Specificity

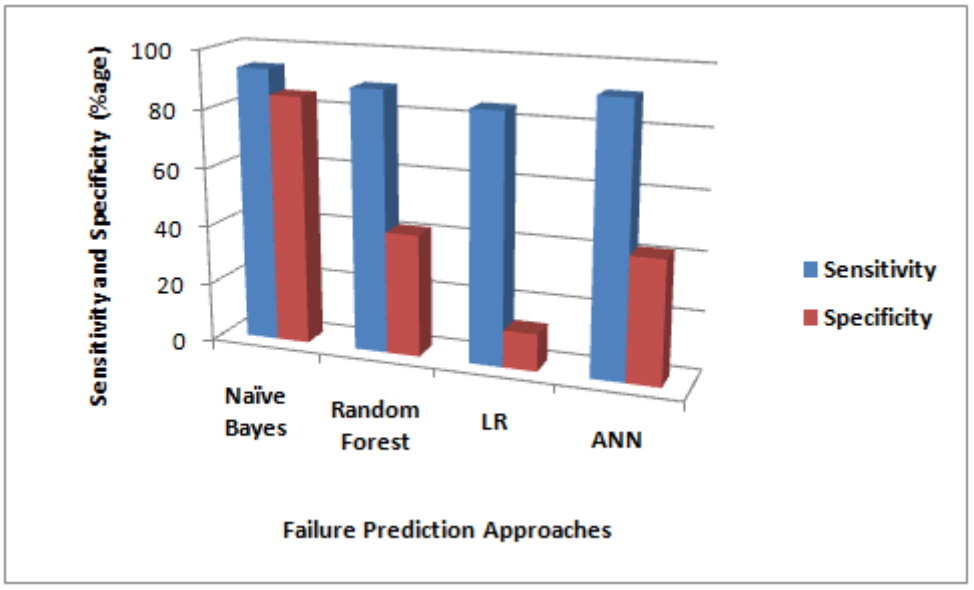


FIGURE 5.2: Sensitivity and Specificity

(84.9%) and Sensitivity (93.5%) where ANN also performs better results than LR and Random Forest with Sensitivity (91.3%) and Specificity (42%). LR has minimum Sensitivity (84.8%) and Specificity (12.7%) in comparison with all the approaches.

Case 2: Recall and Precision

It is apparent from Figure 5.3 that Recall and Precision is maximum of Naive Bayes (93% and 94%) and minimum value (84% and 75%) for LR. ANN has higher values of Recall and Precision (91.3% and 92.1%) than Random Forest and LR.

Case 3: MAPE and RMSE

Figure 5.4 shows the calculated RMSE and MAPE is minimum (19%) and (29.63%) with Naive Bayes where LR has maximum value of RMSE (35.16%). Random Forest has lower value of RMSE (27%) and higher value for MAPE (72.0051%) than ANN and LR. The ANN has lower value of MAPE (57.041%) than LR and Random Forest.

Case 4: F-Measure and ROC

Figure 5.5 indicates F-measure and ROC curve value is also high (0.937) and (0.983) with Naive Bayes approach whereas LR has minimum values. The F-Measure for ANN (0.893) is higher than Random Forest (0.875) and LR (0.798). The Random Forest has higher value of ROC (0.975) then ANN (0.779) and LR (0.661).

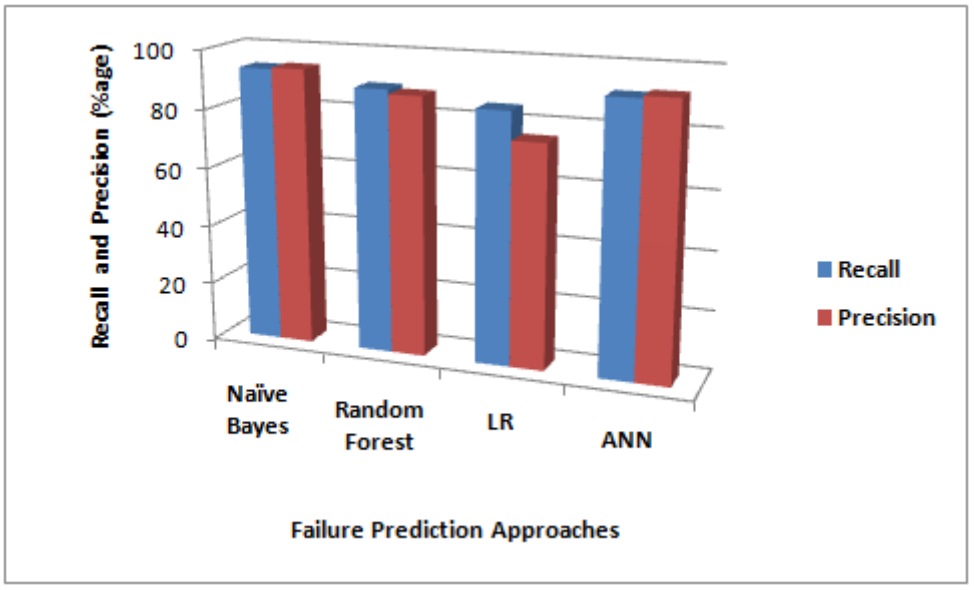


FIGURE 5.3: Recall and Precision

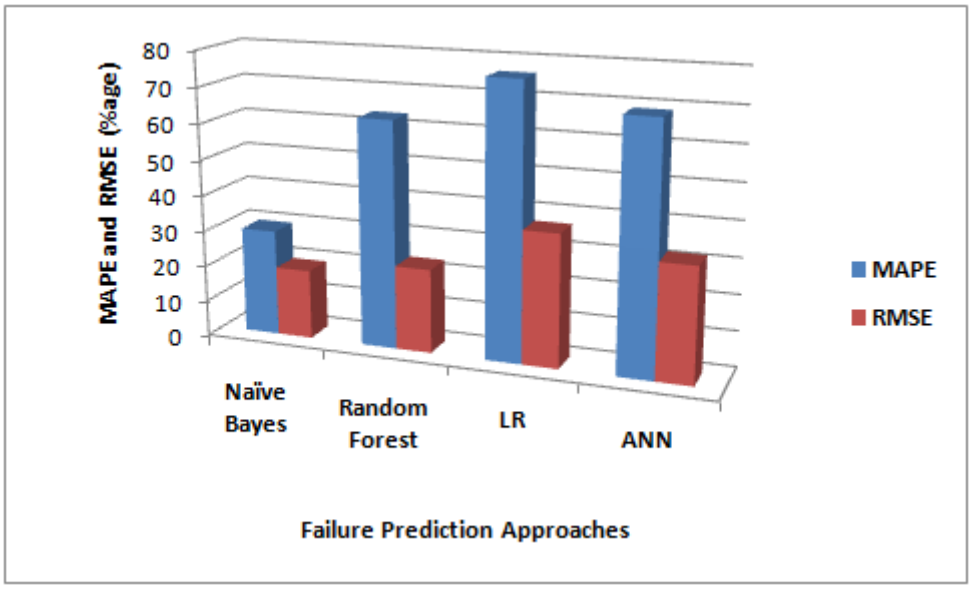


FIGURE 5.4: MAPE and RMSE

The experimental results validate the usage of the Naive Bayes approach for the proposed model of predicted failures is accurate enough to predict task failures with minimum values of %error in terms of MAPE, RMSE and maximum values for the metrics such as Sensitivity, Sensitivity, Recall, Precision, F-Measure and ROC values. Hence, Naive Bayes based models renounce higher prediction accuracy with percentage split as compared to others such as Random Forest, LR and ANN. Therefore, to achieve the actual accuracy with prediction, the task failure prediction for multiple scientific workflow type applications has been compared with actual failures which is discussed in the next section.

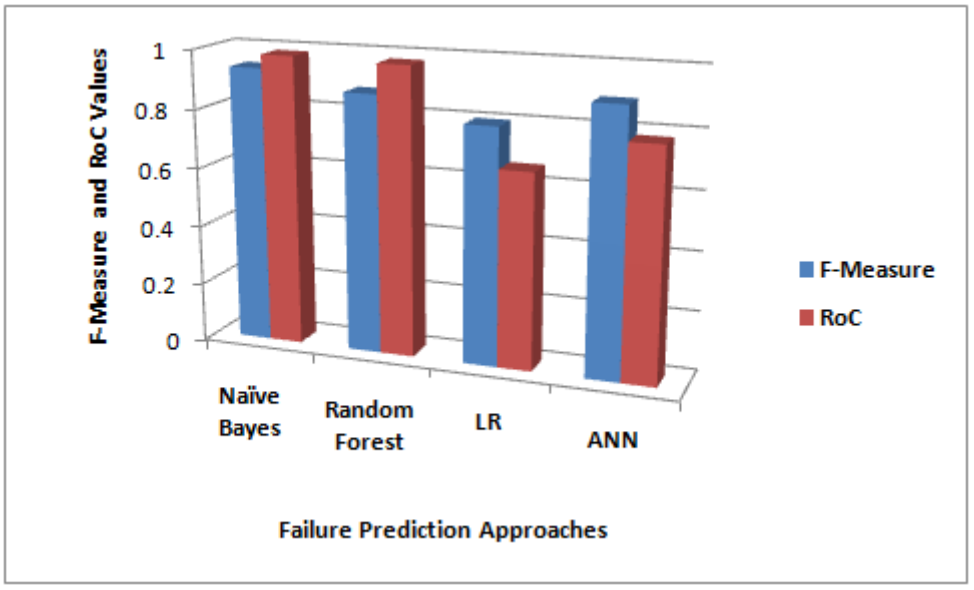
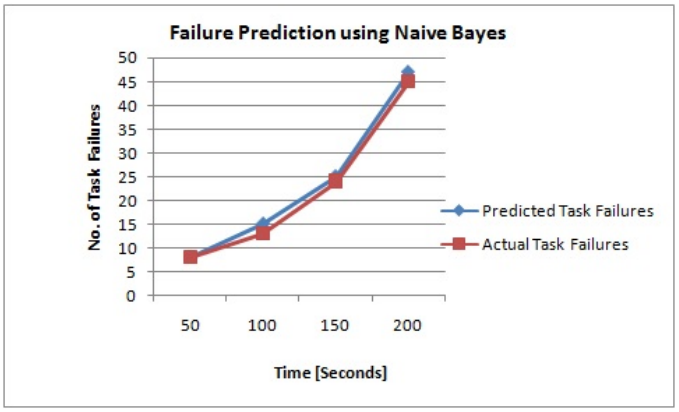


FIGURE 5.5: F-Measure and ROC

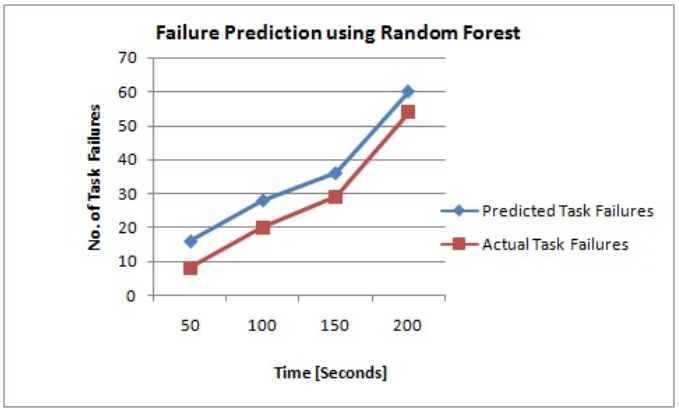
5.2.3 Performance Measures using SEOM

To evaluate the accuracy of prediction model, machine learning based approaches have been used to predict failures precisely for different size of scientific workflows. Then, the experimental results have been validated in Pegasus and Amazon EC2 for actual failures by focusing on Montage workflow data with 25 tasks , Cybershake with 30 tasks, Inspiral with 50 tasks and SIPHT with 100 tasks at different time intervals of 50 ,100, 150 and 200 seconds. Figure 5.6 compares the actual task failures and predicted task failures to different size of scientific workflow based on the evaluated performance metrics at different time intervals. Figure 5.6(a) depicts the results of failure predictions using Naive Bayes algorithm with more accuracy (94%) than Random Forest, LR and ANN for predicting task failures. The effect of failure predictions using Random Forest is also displayed in Figure 5.6(b). The accuracy of the LR is lower than Naive Bayes and Random Forest which is defined in Figure 5.6(c). The accuracy of ANN is also high for all the workflows as compared to LR and Random Forest as shown in Figure 5.6(d). Therefore, according to predictions as the workflow size grows, the number of correct predictions goes up for all the machine learning models.

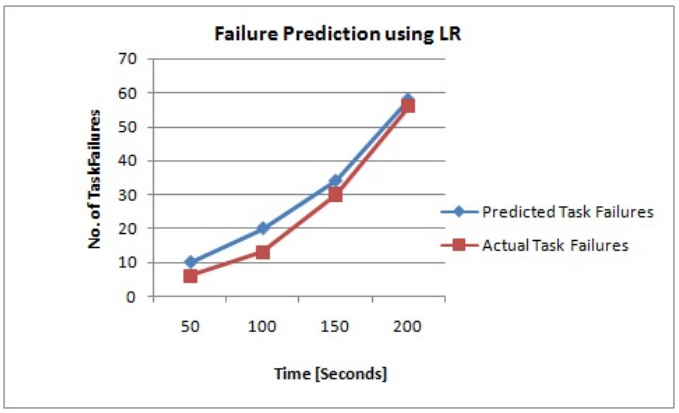
Figure 5.7 compares the SEOM of machine learning approaches and results of SEOM clearly validate that the SEOM is minimum (6%) for Naive Bayes with accuracy of 94% approximately and maximum value (33%) of SEOM for LR with minimum accuracy of (75%) approximately. Random Forest and ANN has lower



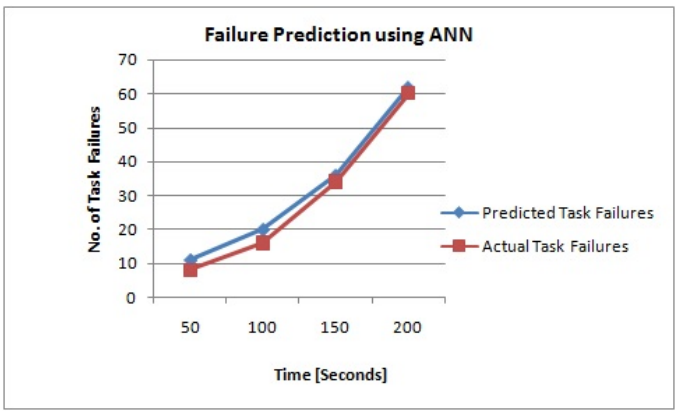
(a) Naive Bayes with Scientific Workflows



(b) Random Forest with Scientific Workflows



(c) LR with Scientific Workflows



(d) ANN with Scientific Workflows

FIGURE 5.6: Comparison of Actual Failures with Predicted Failures

SEOM (20% and 11%) than LR and more than Naive Bayes. Therefore, it can be inferred that the predicted model based on Naive Bayes might lead to construction of the optimum prediction model for developing intelligent failure prediction in the Cloud environment.

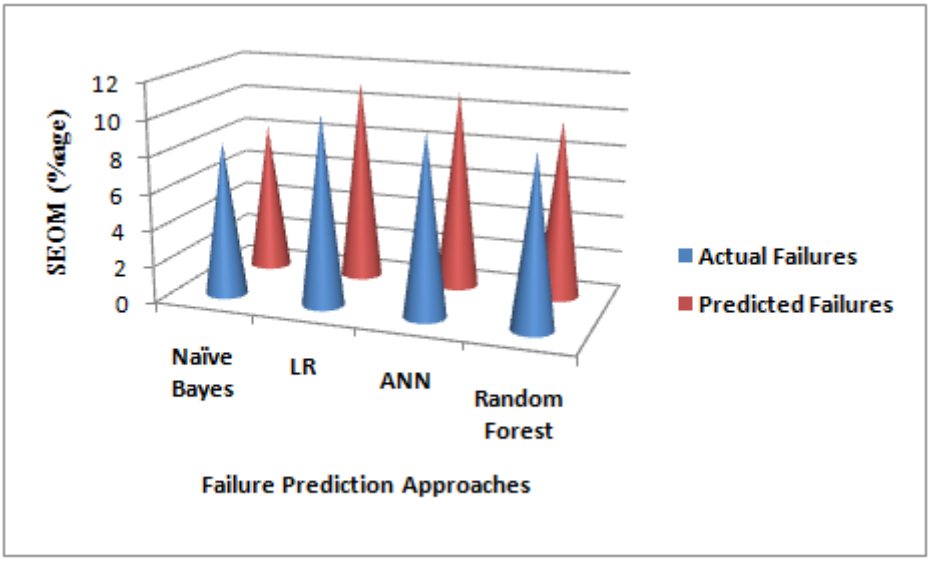


FIGURE 5.7: SEOM of Failure Prediction Approaches

5.2.4 Comparison of Proposed Model with Existing model

To evaluate the usefulness of our proposed model, the experimental results have been compared with existing ones [67] by implementing Broadband with 81 tasks, Epigenome with 320 tasks and Montage with 10,429 tasks using proposed failure prediction model. Figure 5.8 depicts that the proposed model using Naive Bayes has higher accuracy of (97%) with Epigenome workflows as these workflows are high CPU intensive than Broadband and Montage workflows where the existing model using Naive Bayes (NB1)[67] has the accuracy of (96%). The Broadband has also the maximum accuracy of (87%) with Naive Bayes Model, whereas LR has minimum accuracy of (79%) among other approaches and NB1 have the accuracy of (74%). The Montage workflows are I/O intensive which has the accuracy of (94%) with Naive Bayes and minimum accuracy (83%) with Random Forest where the NB1 has accuracy of (84%). Therefore, the average accuracy for all the workflows using Naive Bayes is maximum (93%) among other approaches ANN, LR and Random Forest whereas existing model have the average accuracy of (85%). Thus, it can be validated that the proposed model is more effective with Naive

Bayes approach for predicting task failures as well as resource failure by considering resource utilization for multiple scientific applications.

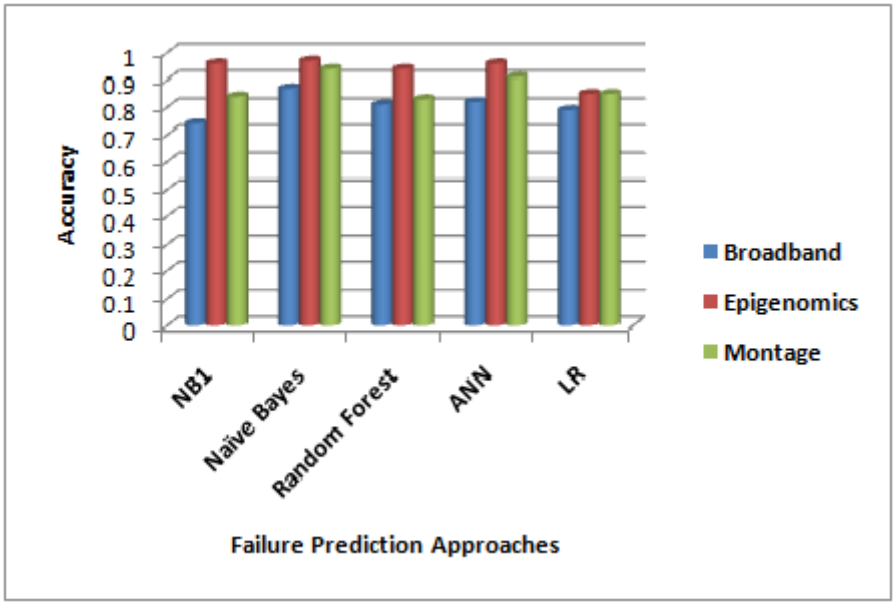


FIGURE 5.8: Comparison of Existing and Proposed Failure Prediction Model

This section discussed an effective approach to implement an intelligent failure prediction model using scientific workflows data in Cloud environment. The analysis of the results in Pegasus and Amazon EC2 have proved the accuracy of task failure prediction models and verified that the maximum accuracy of 94% was achieved with Naive Bayes by comparing actual task failures and predicted task failures. Finally, the failure prediction model using Naive Bayes has shown the effectiveness with average accuracy of (93%) as compared to existing model for Epigenome, Broadband and Montage workflows.

After validating the maximum accuracy for task failure prediction using Naive Bayes, it has been utilized for implementing an Autonomic Fault Tolerant Technique that has been elaborated in the subsequent section.

5.3 Experimental Results: Autonomic VM Migration Approach

To analyse the performance of the proposed technique on a large-scale virtualized datacenter is tremendously difficult. Therefore, to evaluate the effectiveness

of the proposed technique, it has been implemented and tested using Cloudsim 4.0 which supports modeling and simulating one or more VMs on a datacenter [41][33]. The datacenter has been simulated through 100 heterogeneous nodes where the first half of the nodes are HP ProLiant ML110 G4 servers and other half of the nodes consist of HP ProLiant ML110 G5 servers. The experimental results have been evaluated by using different threshold value of resource utilization parameters. Then, the experimental results have been compared between Maximum Correlation (MC) coefficient approach [110] and proposed approach (MC1) through the proposed task failure prediction model. The evaluated results using single resource utilization parameter of (CPU) and multiple resource utilization parameters of (CPU, Bw, RAM, DiskI/O) confirm that the proposed Autonomic Fault Tolerant Technique is better in terms of execution mean time, standard deviation time, number of VM migrations and SLAV.

5.3.1 Result Analysis using Single Parameter

Several metrics have been used to assess the performance and efficiency of the proposed technique. The selection of performance parameters has to be done in such a manner that it must be able to find the effectiveness of the technique in terms of time and also be able to reduce the number of migrations. The descriptive parameters have been used like mean, standard deviation that help to understand how much average reallocation time it takes to handle the task failures. Table 5.3 shows the comparative analysis of proposed inter-correlation coefficient approach using CPU resource utilization parameter with existing multiple correlation coefficient approach (MC). Further, these results have been detailed in the Figures 5.9, 5.10, 5.11 and 5.12.

(i) *VM Reallocation Mean Time*: Figure 5.9 describes the average time taken by VM for migration or reallocation. VM reallocation implies to move the overloaded VM to another host by finding the correlation of CPU utilization and inter-correlation of overloaded VM with other VMs on the same host. The execution time for 0.1 threshold value is higher in the case of existing approach 0.00089 s. while for the proposed approach it is 0.00011 s. For other threshold values of CPU utilization such as 0.2, 0.4, 0.5, 0.7, 0.9 and 1.1, it can be inferred that the proposed approach is performing better than existing approach.

TABLE 5.3: Comparative Analysis of Proposed Approach with Existing

Th.Value	**Approach**	**SLAV**	**No.of Migrations**	**Mean Execution Time**	**Mean Standard Deviation Time**
0.1	Existing(MC)	0.00015691	101	0.00089	0.00089
0.2	Existing(MC)	0.000159068	114	0.00011	0.00133
0.3	Existing(MC)	0.00022887	121	0.00005	0.00089
0.5	Existing(MC)	0.000590043	128	0	0
0.7	Existing(MC)	0.000837352	287	0.00016	0.0016
0.9	Existing(MC)	0.00077191	780	0.0006	0.00299
1.1	Existing(MC)	0.000671692	1199	0.00157	0.0047
0.1	Proposed(MC1)	0.000113004	92	0.00011	0.00129
0.2	Proposed(MC1)	0.000113004	98	0.00006	0.00094
0.3	Proposed(MC1)	0.000201784	108	0	0
0.5	Proposed(MC1)	0.000556532	130	0	0
0.7	Proposed(MC1)	0.00075296	242	0.00006	0.00094
0.9	Proposed(MC1)	0.000638794	626	0.00043	0.00144
1.1	Proposed(MC1)	0.000514103	1037	0.00108	0.00397

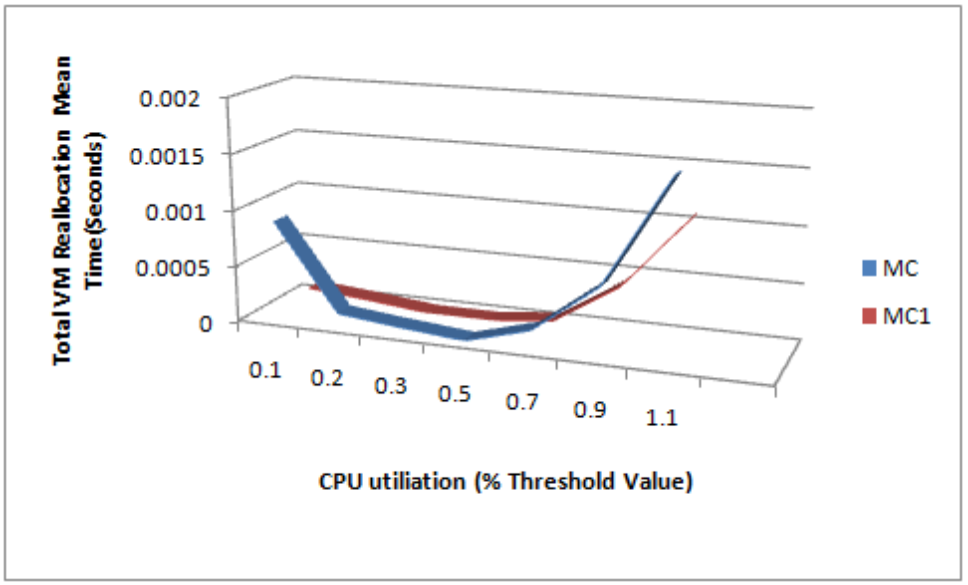


FIGURE 5.9: Execution Time VM Reallocation Mean

(ii) *Total Standard Deviation VM Reallocation Mean Time*: Standard deviation evaluates the distribution of data around both sides of the mean. As the difference between minimum and maximum value increases, the standard deviation would be high. Figure 5.10 shows that the standard deviation for threshold value 0.1 is higher for proposed approach (0.00129 s) whereas for existing approach is 0.00089 s of the mean. But for other threshold values, the standard deviation is minimum for the proposed approach, which validates the proposed approach in terms of consistency.

(iii) *Number of VM migrations*: VM Migration migrates the VMs into a saved

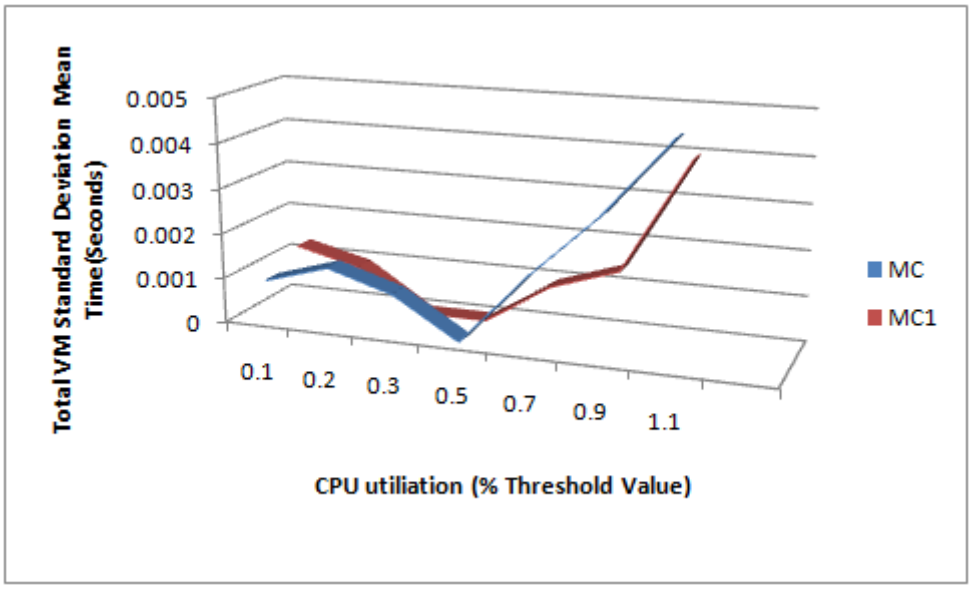


FIGURE 5.10: Execution Time VM Reallocation Standard Deviation

state during the migration from faulty or overloaded host to other working host. Figure 5.11 demonstrates that the count of total VM Migrations is less in proposed approach for all the threshold values. The difference between number of migrations for the proposed and existing approach increases as threshold value increases. For the threshold value 1.1 where the proposed approach has migrations 1037 as compared to existing which has 1199. However, it can be useful to optimize the performance during live migration of VMs.

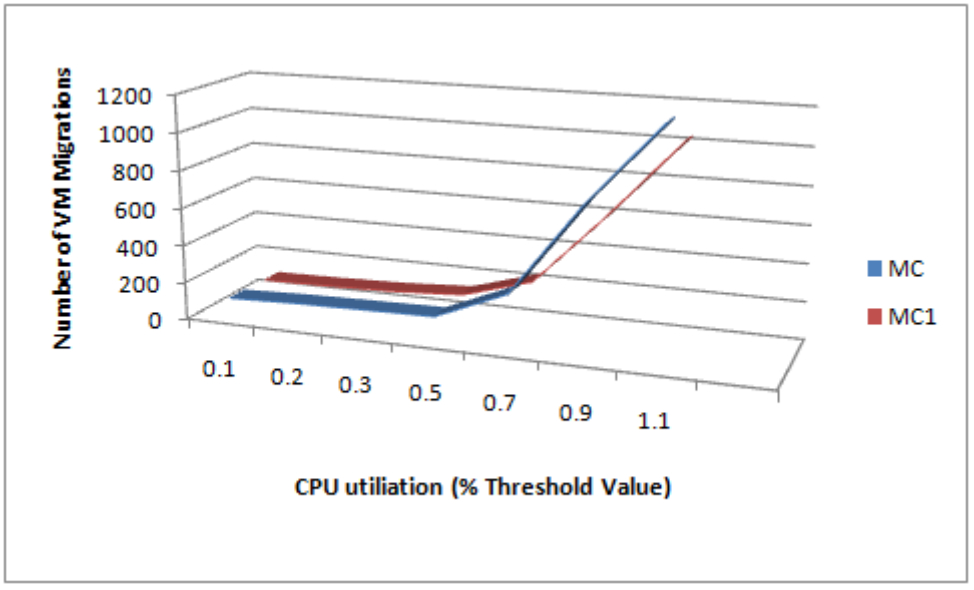


FIGURE 5.11: Number of VM Migrations

(iv) *SLA Violations*: SLA Violations should be minimum for the efficient VM migration approach. Figure 5.12 shows that the SLA violations reduce in proposed approach in comparison to the existing ones. For the threshold value 1.1, SLAV are 0.000514103% through proposed approach and for existing approach, the SLAV are 0.000671692%. Thus, the results verified that the value of SLAV is minimum for all the threshold cases in the proposed approach.

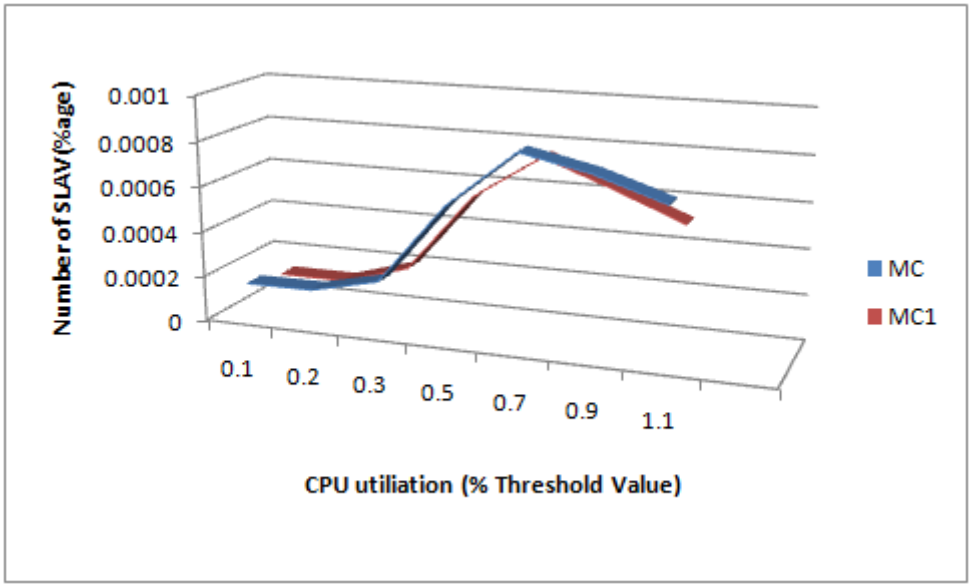


FIGURE 5.12: SLA Violations

5.3.2 Result Analysis using Multiple Parameters

To evaluate the performance of each VM migration metric, the metrics for multiple resource utilization parameters such as CPU, RAM, Bandwidth and Disk I/O have been validated through different threshold values such as 0.2, 0.4, 0.6 and 0.8 as detailed below:-

(i) *Total VM Reallocation Mean Time*: Figure 5.13 shows how much time it takes on an average for reallocating and migarting VMs. It can be inferred that the average time has improved due to the proposed approach as it is used for reallocating the machines based on finding the inter-correlation factor using multiple resource utilization parameters. For the maximum threshold value 1.0, the mean of execution time for the existing approach is 0.05093728 seconds whereas in case of the proposed approach, the value of the mean is 0.02531359 seconds which is appreciably smaller that the

existing ones. Thus, the results of mean execution time corroborated that the proposed approach is better for reducing the mean execution time.

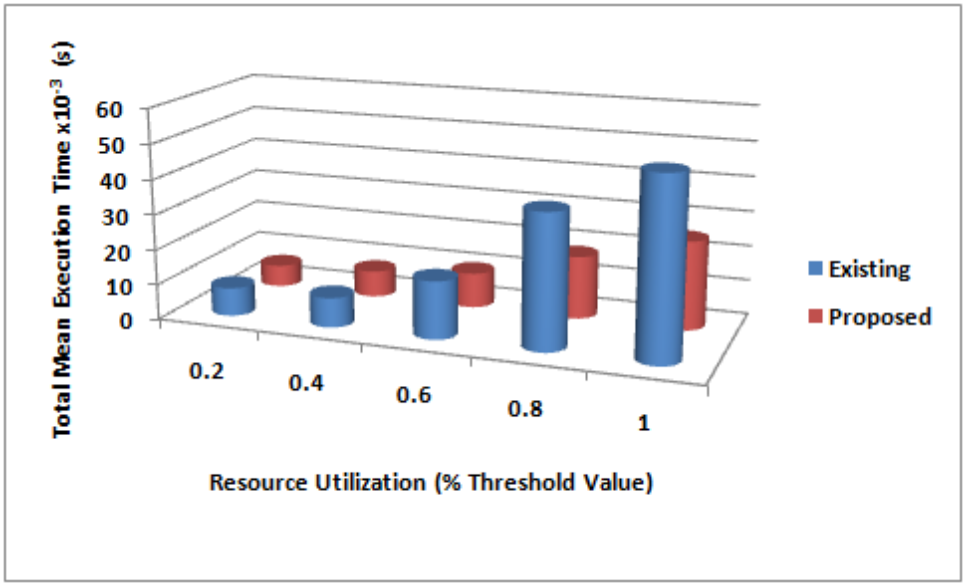


FIGURE 5.13: Total Mean Execution Time

(ii) *Total Standard Deviation Mean Time*: For all the threshold values, the standard deviation is minimum for the proposed algorithm as shown in Figure 5.14, which signifies that the proposed approach performs better during reallocation of VMs.

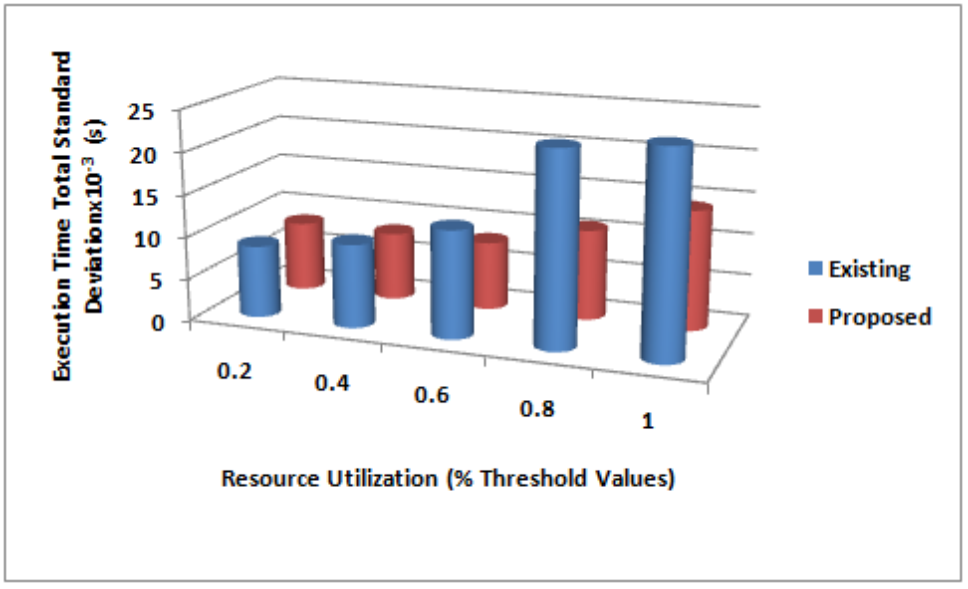


FIGURE 5.14: Total Standard Deviation Mean Execution Time

(iii) *SLA Violations*: Figure 5.15 shows that the SLA violations are significantly decreasing in the proposed approach as compared to the existing ones. There are 0.000157 % violations in the case of maximum correlation coefficient approach for threshold value 0.2 whereas for proposed approach there are 0.000113 violations. The SLA violations are appreciably increasing as threshold value raised and for the threshold value 0.8, it gives the best results for proposed approach with SLAV (0.000672) and for existing, the value of SLAV is 0.000514.

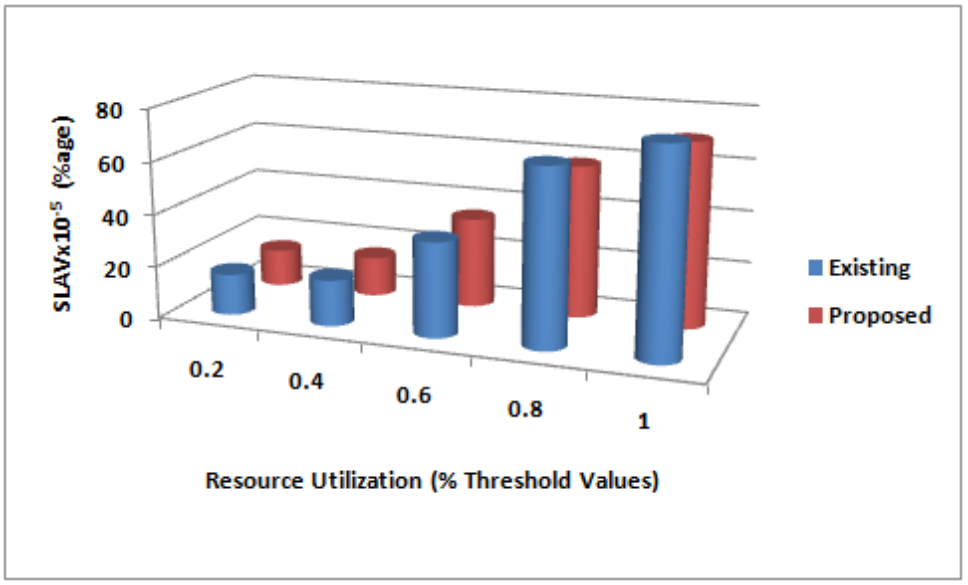


FIGURE 5.15: SLA Violations

(iv) *Number of VM migrations*: Figure 5.16 depicts that there are minimum number of migrations in the proposed VM migration approach. For the threshold value of 0.2, the total number of VM migrations for the existing correlation coefficient approach is 515 whereas in the proposed approach the count is 427. Similarly, for all other threshold values, the number of migrations has reduced as the threshold value is increased by 0.2. For threshold value 1.0, there are maximum number of (5193) VM migrations in the existing approach while for the proposed approach, the number of migrations are 3860. Hence, the results depicted through the graph, demonstrate that the proposed approach is effective during overutilization of hosts

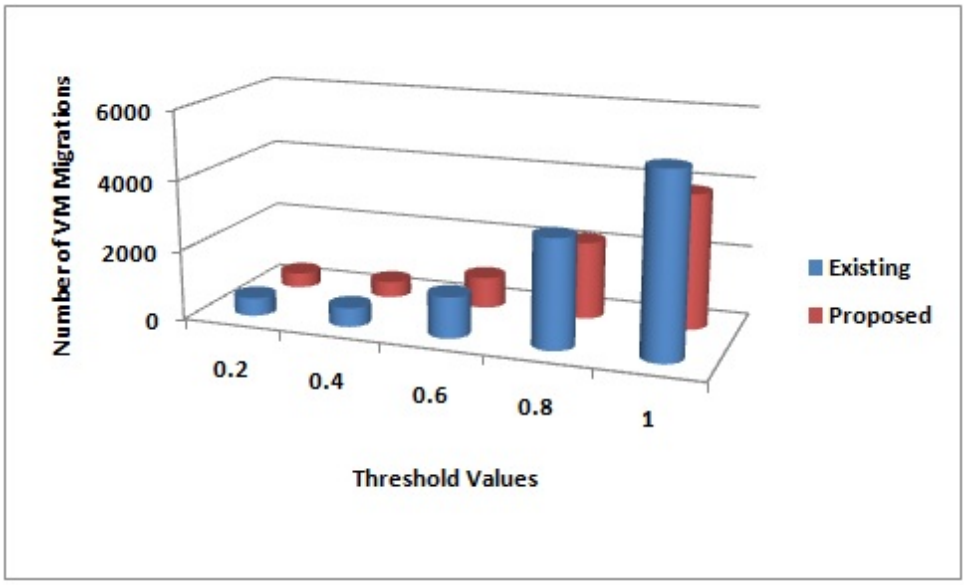


FIGURE 5.16: Number of VM Migrations

5.4 Experimental Results: Autonomic Fault Tolerant Workflow Scheduling

The proposed Autonomic Fault Tolerant Task Scheduling algorithm has been implemented and tested using WorkflowSim 1.0 along with Cloudsim 3.0.2 due to following reasons [84][87].

- It supports modeling and simulating one or more VMs on a data center.
- Multiple users can submit workflow tasks for execution concurrently.
- To evaluate various workflow scheduling techniques with various parameters.
- WorkflowSim1.0 considers the scientific workflow applications and failure patterns with optimization techniques.
- Workflow tasks can be heterogeneous and can be either CPU or I/O intensive.

The comparative analysis of the proposed algorithm has been done with the existing heuristics such as Min-Min and Max-min, MCT. But if any overloading or task failure has been detected, the proposed VM migration approach has been compared with multiple correlation coefficient approach [110](existing). For validating the experimental results, the data center has been simulated with 20 heterogeneous nodes where 10 nodes are HP ProLiant ML110 G4 servers, and the other 10 consist

of HP ProLiant ML110 G5 servers. The results have been also evaluated and compared by implementing VM migration with scheduling heuristics such as Particle Swarm Optimization (PSO), Genetic Algorithm (GA), Min-Min, Max-Min, MCT and proposed hybrid heuristic.

5.4.1 Performance Evaluation

Various evaluation metrics such as Makespan, Execution Mean Time and Standard Deviation of Mean Time have been used to evaluate the performance and efficiency of the proposed algorithm which are shown below:

(i) ***Case 1***- *Makespan* : The performance of the proposed scheduling algorithm has been analysed with Montage and other scientific workflow applications such as Cybershake, Inspiral, and SIPHT. Cybershake workflow is used to identify earthquake hazards with five levels, whereas Inspiral workflow has been utilized for identifying gravitational waves produced by several actions with six levels. SIPHT Workflow is employed for the bioinformatics project to generate replicas of bacteria by exploring the small RNA genes and Montage is used for astronomical images from mosaics. Furthermore, to validate the experimental results, 25, 30, and 100 jobs are considered where each job can contain hundred to thousand of tasks. It is apparent from Figure 5.17 that the proposed heuristic has minimum makespan of 91.62 s for montage workflow with 25 jobs, whereas Min-Min heuristic has the maximum makespan of 106.17 s. Correspondingly, for Cybershake workflow with 30 jobs, the highest makespan of 513.52 s is attained with Min-Min heuristic and lowest makespan of 320.78 s is obtained with the hybrid heuristic. SIPHT workflow and Inspiral workflow also achieve lowest makespan value of 2794.39 and 4411.96 s with hybrid heuristic as compared to that with other heuristics. Figure 5.18 also validates the usefulness of proposed heuristic by increasing the size of scientific workflow to 100 jobs. It is apparently visible from the experimental results that SIPHT, Montage, and Inspiral workflows have maximum makespan with MCT heuristic, whereas Cybershake workflow has highest makespan using Min-Min heuristic. Although for all the scientific workflows the proposed hybrid heuristic has minimum makespan, the experimental results clearly demonstrate that the hybrid heuristic is the best heuristic for all the example workflows.

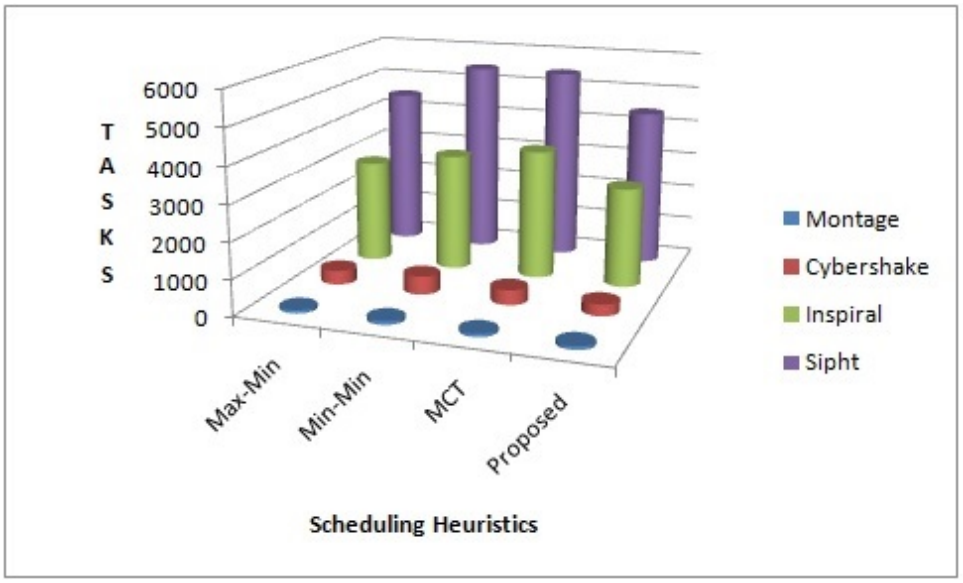


FIGURE 5.17: Makespan of Scientific Workflows with 25 and 30 jobs

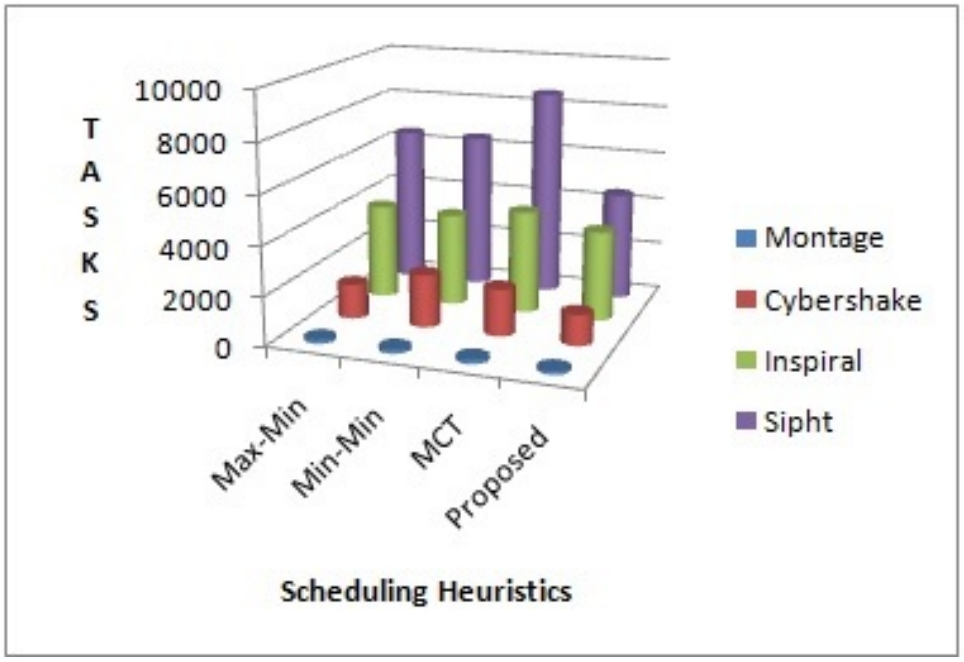


FIGURE 5.18: Makespan of Scientific Workflows with 100 Jobs

(ii) ***Case 2***- *Total Mean Execution Time*: It can be seen from the Figure 5.19 that the mean execution time has been improved in the proposed approach as it is working to reselect or reallocate the machines based on inter-correlation factor with multiple parameters which is affecting the VM migration decision making process. VM reallocation implies to move the overloaded VM to

another host and the total mean time is the average time to perform reallocation of overloaded VMs to another host. For threshold value 0.2, the proposed VM migration approach has an execution time of 0.00255749 s which is minimum, whereas MC (existing) has an execution time of 0.00272822 s that is maximum. For other threshold values 0.4, 0.8, and 1, the proposed approach reduces the mean execution time also. Furthermore, for threshold value 0.6, MC has the highest execution time of mean among other threshold values which is 0.03808362 s,whereas the proposed approach has minimum execution time of 0.003327526 s. Since the random workload class is used to generate the resource utilization values, the mean of the execution time can be maximum or minimum at any threshold value randomly. For all the cases, MC1 enhanced the performance of MC, whereas the proposed approach has demonstrated the best results with both the existing policies MC and MC1. Therefore, the proposed VM migration works superior to reselect or reallocate the machines based on inter-correlation factor with multiple parameters.

FIGURE 5.19: Execution Time Mean

(iii) ***Case 3***- *Standard Deviation of Execution Time*: Figure 5.20 describes the effectiveness of the proposed VM migration approach using multiple parameters. For 0.2 threshold value, MC1 has minimum standard deviation time of 0.005226518 s, whereas MC has maximum standard deviation time as 0.006101932 s and the proposed approach has a time of 0.005792689 s which is further minimized for all other threshold values such as 0.4, 0.6, 0.8, and

1. Hence, it can be validated that the proposed approach is effective also to reduce the execution time as well as standard deviation of mean time.

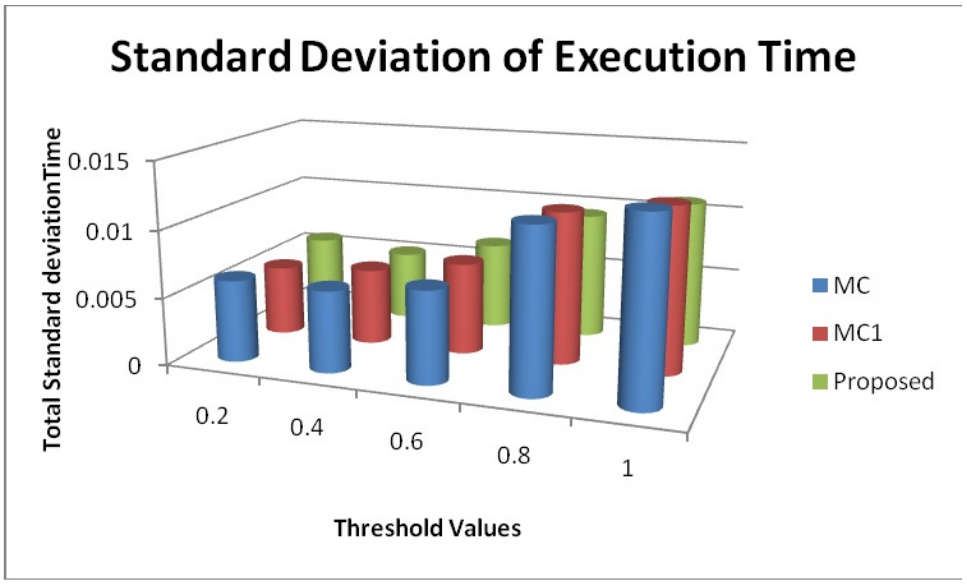


FIGURE 5.20: Execution Time Standard Deviation

5.4.2 Comparative Analysis of Autonomic Fault Tolerant Scheduling with Existing Approaches

For the comparative analysis, very few approaches are found which have utilized the concept of fault tolerant scheduling in Cloud; none of them has incorporated the implementation of workflow scheduling approaches along with VM migration. As PSO [60] has used to optimize the cost for workflows and GA [67] for optimizing the execution time, but both of the heuristics do not consider the fault tolerance aspect. Therefore, the comparative analysis of proposed approach has been done after implementing the existing scheduling heuristics with VM migration approach in Cloud environment. The experimental results in Figure 5.21 evidently confirm that the proposed scheduling approach is efficient for large-scale workflows also, such as Cybershake and Epigenome with 1000 jobs. These results have been validated using the existing heuristics such as PSO , GA , Min-Min, Max-Min , MCT and proposed heuristics. It is apparent from the experimental results for Epigenome workflow that the PSO performs better than Max-Min and MCT, whereas GA also enhances the performance over Max-Min, Min-Min, MCT, and PSO. Similarly, for Cybershake workflow, PSO performs better than Max-Min, Min-Min, MCT, and GA. Furthermore, the hybrid heuristic combines the features

of all other heuristics along with VM migration approaches, thus the proposed heuristic outperforms all the existing heuristics by reducing the average makespan for large-scale scientific workflows such as Cybershake and Epigenome.

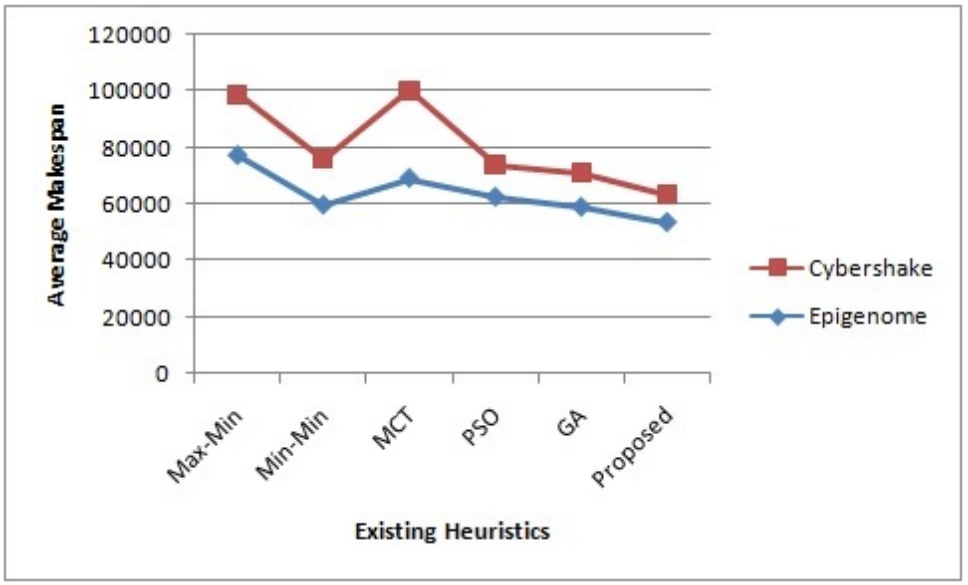


FIGURE 5.21: Comparison of Proposed Approach with Existing Approach

5.5 Conclusion

This chapter thus verified the maximum accuracy of Naive Bayes by implementing proposed model on Cloud infrastructure. Then, Naive Bayes model has been utilized to predict the task failures. An autonomic fault tolerant technique migrates the faulty virtual machines on other working hosts autonomically by appreciably reducing mean execution time, standard deviation of mean execution time and SLAV. Further, the experimental results for proposed scheduling algorithm for multiple workflows along with autonomic fault tolerant scheduling have been validated on CloudSim and WorkflowSim toolkits by reducing average makespan.

The next chapter presents the concluding remarks for all the chapters along with their future directions.

Chapter 6

Conclusion and Future Directions

Autonomic Workflow Scheduling is a desired aspect for any large scale distributed system and is even more important in dynamic infrastructures such as Clouds. The complexity of the workflow scheduling further increases due to the presence of multiple scientific applications having the demand of reliable, cheap and efficient scheduling on heterogenous resources. Therefore, existing Cloud Computing systems ought to be autonomic, largely, to handle the challenges of fault-tolerance and workflow scheduling effectively.

This thesis persistently addresses the autonomic fault handling for efficient workflow scheduling of multiple workflows in Clouds. The research work carried out the implementation of intelligent model to predict task failures using machine learning approaches, which is based on the failure analysis of multiple scientific workflow applications. It intended to propose and design an autonomic fault tolerant technique through VM migration and further, workflow scheduling algorithm is proposed in conjunction with an autonomic fault tolerant technique.

At last, in this chapter, the concluding remarks on this research work have been given by describing the advancement made towards the objectives of the research in terms of autonomic fault tolerant scheduling for multiple scientific workflow applications. This chapter specifies the outcome of each chapter and later highlights the contributions of the proposed work. Furthermore, it also confers the future directions for the research community.

6.1 Conclusion

The objective of this research work has been to design and implement an autonomic fault tolerant workflow scheduling in Cloud Computing environment. It has been addressed by the proposed workflow scheduling algorithm for scientific applications through Autonomic Fault Tolerant technique. The key findings of the research work has been discussed below:

The thesis work starts from the introductory section of Cloud Computing in **Chapter 1** that overviews the Cloud Computing along with related technologies. Further, the basics of autonomic computing, workflows and fault tolerant techniques have been illustrated that end up in the open research issues along with thesis contributions. The chapter finally comes up with research motivation.

Chapter 2 provides the state of the art of workflow scheduling algorithms, fault tolerant techniques and autonomic concepts. From the existing survey, it has been identified that very few workflow scheduling algorithms are fault tolerant. None of the discussed approaches have implemented fault tolerant scheduling for scientific workflows in an autonomic way. In addition, these scientific workflows are highly complex and heterogenous and most of them need to incorporate with self-healing techniques of autonomic computing. However, in order to cope up with the workflow scheduling, these workflows are required to be scheduled through autonomic fault tolerance techniques. Based on the findings of the literature survey, the research problem has been defined in the concluding remarks.

In **Chapter 3**, task failure prediction model is proposed based on the machine learning approaches by analysing the data of scientific workflow applications. The proposed model is further employed to select more accurate and efficient machine learning approach for prediction by implementing in Cloud environment. Further, the experimental results verified that the Naive Bayes approach has maximum accuracy as compared to the existing model for Epigenome, Broadband and Montage workflows. Henceforth, Naive Bayes has been further utilized to heal the task failures intelligently. Then, fault tolerant approach has been identified through proposed VM migration policy using task failure prediction on allocated VM. The proposed policy evaluates the correlation between utilization parameters and virtual machines and also finds the inter-correlation between VMs. Then, VM is

migrated which is suitable for migration to other working hosts as to heal the task failures automatically. The proposed technique is further utilized for workflow scheduling which is discussed in **Chapter 4**.

The previous scheduling heuristics do not combine the features of Max-Child, Min-Min and FCFS heuristic along with deadline based scheduling to schedule multiple workflows on Cloud environment. Henceforth, to address this issue in **Chapter 4**, scheduling approach for multiple workflows has been proposed that leverages a deadline based scheduling along with the proposed hybrid heuristic. The proposed algorithm utilized the autonomic fault tolerant technique to migrate VMs during the occurrence of task failure. Further, various performance evaluation parameters have been defined to measure the effectiveness of the proposed autonomic fault tolerant scheduling approach by describing the cases of the multiple scientific applications. It has also been validated by comparing with the existing heuristics for large scale workflows such as Epigenome and Cybershake with 1000 jobs.

Further, the experimental results have also been implemented in **Chapter 5** by categorizing into three sections. In the first section of this chapter, proposed failure prediction model has been implemented using ANN, LR, Random Forest and Naive Bayes. The experimental results have validated the maximum accuracy of 94% and minimum error with Naive Bayes. Henceforth, in the second section, Naive bayes has been employed for task failure prediction and Autonomic Fault Tolerant technique has been implemented by migrating faulty VMs to other hosts using inter-correlation coefficient approach. The experimental results using CPU utilization parameter and multiple resource utilization parameters have been evaluated that minimized the execution mean time, number of VM migrations and SLA violations etc. Further, third section has verified the proposed autonomic fault workflow scheduling by comparing with existing scheduling algorithm for the large scale multiple workflows. The results are included in this chapter and the following key contributions of the thesis are listed below:-

- Literature review of the workflow scheduling algorithms, fault tolerant and failure prediction techniques along with autonomic self-healing concepts .

- Intelligent failure prediction model that predicts the task failures for scientific workflows in Cloud environment.

- Autonomic Fault Tolerant technique that takes into account predicted task failures with VM migration policy to proactively migrate the VMs to another active host.
- Workflow scheduling algorithm that schedules multiple workflows through Earliest Deadline First, hybrid heuristics and utilizes the proposed autonomic fault tolerant approach.

6.2 Future Directions

This thesis enhances the understanding of workflow scheduling and fault tolerance in Cloud Computing environment and advances the state-of-the-art through its contributions. Moreover, the contributions of this thesis have led to the new research areas in Cloud Computing that are required to be addressed through further research. Therefore, this section briefly depicts some of these challenges within the scope of the thesis. The following future directions have been suggested for the research community:

- The incorporation of Naive Bayes model with other fault tolerant techniques would be able to enhance the reliability and availability of Cloud services and to improve the performance of existing proactive fault-tolerant approaches.
- Failure prediction model can also be useful for resource provisioning and to accomplish time and cost based optimization and further it can increase the scheduling efficiency of different size workflow applications.
- The intelligent prediction model would also be valuable for predicting and analysing security threats which can further enhance the performance of various expert system applications such as project management, risk assessment and engineering etc.

- Adaptive checkpointing can be used to migrate the VMs from the last checkpoint by incorporating other failures such as libraries are not installed, network failures and system running out of bound etc.

- The proposed fault tolerant scheduling addresses the task failures due to overutilization of resources and scheduling issues in Cloud Computing. For future aspects, more QoS parameters including scalability, availability and reliability can be considered and their impact on the overall performance can be examined in Cloud environment.

- Various workflow optimization and clustering techniques can be used to optimize the performance of workflow applications. It can be tested under real testbed like Pegasus using Hadoop , for other scientific applications like Broadband and Flooding also.

Bibliography

[1] P. Dasgupta, LeBlanc R.J., and W.F Jr., Appelbe. The clouds distributed operating system: functional description, implementation details and related work. *8th International Conference on Distributed Computing Systems*, pages 2–9, 1988. URL https://www.researchgate.net/publication/23700633.

[2] P. Dasgupta, R. Chen, S. Menon, M. Pearson, R. Ananthanarayanan, U. Ramachandran, M. Ahamad, R. J. LeBlanc, W. Applebe, J. M. Bernabeu-Auban, P. Hutto, M. Khalidi, and C. J. Wileknloh. The design and implementation of the clouds distributed operating system. *USENIX Computing Systems Journal*, 3:11–46, 1988. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.55.7747.

[3] Krauter Klaus, Rajkumar Buyya, and Muthucumaru Maheswaran. A taxonomy and survey of grid resource management systems for distributed computing. *Software: Practice and Experience*, 32(2):135–164, December 2002. URL http://www.cloudbus.org/papers/gridtaxonomy.pdf.

[4] Chetty, Madhu, and Rajkumar Buyya. Weaving computational grids: How analogous are they with electrical grids? *Computing in Science andexit Engineering*, 4(4):61–71, December 2002. URL http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1014981.

[5] Sadashiv Naidila and SM Dilip Kumar. Cluster, grid and cloud computing: A detailed comparison. *Computer Science and Education (ICCSE), 2011 6th International Conference, IEEE*, pages 477–482, August 2011. URL http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6028683.

[6] Foster Ian. The anatomy of the grid: Enabling scalable virtual organizations. *International journal of high performance computing applications*, 15(3):200–222, 2001. URL http://www.computer.org/csdl/proceedings/ccgrid/2001/1010/00/10100006.pdf.

[7] Ionut Anghel, Tudor Cioara, Ioan Salomie, Mihaela Dinsoreanu, and Anca Rarau. A policy driven self-healing algorithm for context aware system. *IEEE international conference*, pages 229–236, 2009. URL http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=5284754.

[8] Foster Ian and Carl Kesselman. The grid: Blueprint for a new computing infrastructure. *Morgan Kaufmann: San Fransisco,Book Chapter, second edition CA*, 1999. URL http://www.amazon.com/The-Grid-Second-Edition-Infrastructure/dp/1558609334.

[9] Kaur P.D. and Chana I. Unfolding the distributed computing paradigms. *Proceedings of the IEEE International Conference on Advances in Computer Engineering*, pages 339–342, 2010. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5532807.

[10] Ian Foster, Carl Kesselman, Jeffrey M. Nick, and Steven Tuecke. Grid services for distributed system integration. *Computer Journal*, 35(6):37–46, 2002. URL http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1009167.

[11] Nist. 2010. URL http://www.nist.gov/itl/cloud/.

[12] Rajkumar Buyya, Rodrigo N. Calheiros, and Xiaorong Li. Autonomic cloud computing: Open challenges and architectural elements. *International Conference*, pages 1–8, 2011. URL http://www.cloudbus.org/papers/AutonomicCloudKeynote2012.pdf.

[13] Beloglazov A. and R. Buyya. Managing overloaded hosts for dynamic consolidation of virtual machines in cloud data centers under quality of service constraints. *IEEE Transactions on Parallel and Distributed Systems*, pages 1366–1379, August 2012. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6269025&tag=1.

[14] Q. Zhang, L. Cheng, and R. Boutaba. Cloud computing: State-of-the-art and research challenges. *Journal of Internet Services and Application, Springer*,

pages 7–18, 2010. URL http://cloud.pubs.dbs.uni-leipzig.de/sites/cloud.pubs.dbs.uni-leipzig.de/files/fulltext.pdf.

[15] T. Dillon, C. Wu, and E. Chang. Cloud computing: Issues and challenges. *24th IEEE Intl Conference onAdvanced Information Networking and Applications (AINA)*, pages 27–33, 2010. URL http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=5474674.

[16] Cloud deployment model. 2014. URL http://www.centre4cloud.nl/nl/kennis-ontwikkeling/definition-cloud-computing/deployment-models/.

[17] D. C. Wyld. Moving to the cloud: An introduction to cloud computing in government. *Cloud Computing Report*, 2009. URL http://www.businessofgovernment.org/report/moving-cloud-introduction-cloud-computing-government.

[18] HU Leigang and XIAO Mingqing. The future of automatic test system (ats) brought by cloud computing. *IEEE International Conference, China*, pages 412–414, 2009. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5314089&tag=1.

[19] K. Chard, S. Caton, O. Rana, and K. Bubendorfer. Social cloud: Cloud computing in social networks. *IEEE 3rd International Conference on Cloud Computing (CLOUD)*, pages 99–106, 2010. URL http://www.im.uni-karlsruhe.de/Upload/Publications/aface0bb-d437-49dc-bf6d-a943034c9870.pdf.

[20] Zhao W., P. Melliar, and L. E. Moser. Fault tolerance middleware for cloud computing. *Proceedings of the 2010 IEEE 3rd International Conference on Cloud Computing*, page 67 74, 2010. URL http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=5558010.

[21] A.G. Ganek and T.A. Korbi. The dawning of the autonomic computing era. *IBM Systems Journal*, 42, 2003. URL http://wiki.daimi.au.dk/toppen/_files/ganek2003ibm.pdf.

[22] M. C. Huebscher and J. A. McCann. A survey of autonomic computing-degrees, models, and applications. *ACM Computing Surveys*, 40:1–31, 2008. URL http://spiral.imperial.ac.uk/bitstream/10044/1/5738/1/autonomic-computing.pdf.

[23] Shakti Mishra, D.S. Kushwaha, and A.K.Misra. Hybrid load balancing in auto-configurable trusted clusters. *Journal of Computer Science and Engineering*, 2:16–25, 2010.

[24] Engin pek, OnurMutlu, Josef Martnez, and RichCaruana. Self-optimizing memory controllers: A reinforcement learning approach. *International Symposium on Computer Architecture*, 2008. URL http://research.microsoft.com/en-us/people/ipek/isca08.pdf.

[25] Harald Psaier and Schahram Dustdar. A survey on self-healing systems: approaches and systems. *Computing Journal*, 91:4373, 2010. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.189.4855&rep=rep1&type=pdf.

[26] Jia Yu and Rajkumar Buyya. Scheduling scientific workflow applications with deadline and budget constraints using genetic algorithms. *Scientific Programming*, 14(3):217–230, 2006. URL http://www.cloudbus.org/papers/Workflow_JSP_2005.pdf.

[27] C.L. Philip Chen and Chun-Yang Zhang. Data-intensive applications, challenges, techniques and technologies: A survey on big data. *Information Sciences Journal*, 275:314–347, 2014. URL http://www.sciencedirect.com/science/article/pii/S0020025514000346.

[28] Jia Yu and Rajkumar Buyya. A budget constrained scheduling of workflow applications on utility grids using genetic algorithms. *International Conference*, pages 1–10, 2006.

[29] Jia Yu and Rajkumar Buyya. A taxonomy of workflow management systems for grid computing. *SIGMOD Record*, 34(3):1–33, 2005. URL http://arxiv.org/ftp/cs/papers/0503/0503025.pdf.

[30] S. Hwang and C. Kesselman. Grid workflow: A flexible failure handling framework for the grid. *12th IEEE International Symposium on High Performance Distributed Computing (HPDC03), Seattle, Washington,USA*, pages 126–137, June 2003. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1210023.

[31] Michael Armbrust, Armando Fox, and Rean Griffith. Above the clouds: A berkeley view of cloud computing. *Technical Report,Electrical Engineering and Computer Sciences University of California at Berkeley,*

2009., 2009. URL http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.pdf.

[32] Y. Tao, H. Jin, and X. Shi. Grid workflow scheduling based on reliability cost. *Proceedings of the 2nd international conference on Scalable information systems*, 2007. URL http://eudl.eu/pdf/10.4108/infoscale.2007.895.

[33] Mehmet Yildiz, Jemal Abawajy, Tuncay Ercan, and Andrew Bernoth. A layered security approach for cloud computing infrastructure. *Global Technology Services, IBM Australia, Melbourne, Australia*, 2009. URL http://www.docstoc.com/docs/111345052/A-Layered-Security-Approach-for-Cloud-Computing-Infrastructure.

[34] P.D. Manuel and S. Thamarai Selvi. Trust management system for grid and cloud resources. *First International Conference on Advanced Computing, ,ICAC*, pages 176–181, 2009. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5378187&tag=1.

[35] P. Varalakshmi and S. Thamarai Selvi. Thwarting ddos attacks in grid using information divergence. *Future Generation Computer Systems*, 29: 429–441, 2013. URL http://www.sciencedirect.com/science/article/pii/S0167739X11002093.

[36] Rajkumar Buyya, Rajiv Ranjan, and Rodrigo N. Calheiros. Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services. *International Conference, The University of Melbourne, Australia*, 2009. URL http://arxiv.org/ftp/arxiv/papers/1003/1003.3920.pdf.

[37] Rajkumar Buyyaa, Chee Shin Yeoa, Srikumar Venugopala, James Broberg, and Ivona Brandic. Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems*, 25:599–616, June 2009. URL http://www.sciencedirect.com/science/article/pii/S0167739X08001957.

[38] Jia Yu and Rajkumar Buyya. A taxonomy of workflow management systems for grid computing. *Journal of Grid Computing*, pages 171–200, September 2005. URL http://arxiv.org/ftp/cs/papers/0503/0503025.pdf.

[39] Deelman E. Grids and clouds: Making workflow applications work in heterogeneous distributed environments. *nternational Journal of*

High Performance Computing Applications Online First, pages 1–15, December 2009. URL http://pegasus.isi.edu/publications/2009/Deelman-GridAndClouds.pdf.

[40] Meng Xu, Lizhen Cui, Haiyang Wang, and Yanbing Bi. A multiple qos constrained scheduling strategy of multiple workflows for cloud computing. *IEEE International Symposium on Parallel and Distributed Processing, 2009.*, pages 629–634, 2009. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5207867.

[41] Luqun Li. An optimistic differentiated service job scheduling system for cloud computing service users and providers. *Third International Conference on Multimedia and Ubiquitous, Normal University, Shanghai*, pages 295–299, 2009. URL http://s3.amazonaws.com/.../20026_1_jobscheduling.pdf.

[42] Daniel de Oliveira, Kary A. C.S.Ocaa, Fernanda Baio, and Marta Mattoso. A provenance-based adaptive scheduling heuristic for parallel scientific workflows in clouds. *Journal of Grid Computing*, 10:521–552, 2012. URL http://link.springer.com/article/10.1007%2Fs10723-012-9227-2#page-1.

[43] Shajulin Benedict. Issues and performance analysis tools for hpc cloud applications: a survey. *Journal of Computing*, 95(11):89–108, 2013. URL http://link.springer.com/article/10.1007%2Fs00607-012-0213-0.

[44] P. Varalakshmi, Thamarai Selvi, Javed Ashraf S., and K. A., Karthick. B-tree based trust model for resource selection in grid. *proceeding of Signal Processing Communications and Networking*, pages 222–227, 2007. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4156617&tag=1.

[45] B. Bethwaite, D. Abramson, F. Bohnert, S. Garic, C.Enticott, and T. Peachey. Mixing the grid and clouds: High throughput science using the nimrod tool family. *Cloud Computing: Principles, Systems and Applications, Antonopoulos and Gillam (eds)*, 2010. URL http://www.hawaii.edu/tlpw/sites/www.hawaii.edu.tlpw/files/workshops/dercap09AbramsonGrid_vs_Cloud.pdf.

[46] Shakti Mishra, D.S. Kushwaha, and A.K.Misra. An efficient job scheduling technique in trusted clusters for load balancing. *First International Conference on Cloud Computing, GRIDs and Virtualization, Cloud Computing*, pages 26–31, 2010.

[47] Chunye Gong, Jie Liu, Oiang Zhang, Haitao chen, and Zhengh Gong. The characteristics of cloud computing. *Parallel Processing workshops*, pages 275–279, 2010. URL http://www.postdm.post.ir/_ITCenter/Documents/TheCharacteristicsofCloudComputing_20140722_154207.pdf.

[48] Piotr Chrz, Astowski-Wachtel, Boualem Benatallah, Rachid Hamadi, Milton ODell, and Adi Susanto. A top-down petri net-based approach for dynamic workflow modeling. *LNCS*, 2678:336–353, 2003. URL http://citeseerx.ist.psu.edu/viewdoc/downloaddoi=10.1.1.111.9294rep=rep1type=pdf.

[49] Irene Vanderfeesten, Hajo A. Reijers, Wil M.P., and van der Aalst. Product based workflow support: A recommendation service for dynamic workflow execution. *LNCS*, 5074:571–574, 2008. URL http://link.springer.com/chapter/10.1007%2F978-3-540-69534-9_42.

[50] Josefina Guerrero Garca, Christophe Lemaigre, Juan Manuel, and Gonzlez Calleros. Model-driven approach to design user interfaces for workflow information systems. *Journal of Universal Computer Science*, pages 3160–3173, 2008. URL http://www.jucs.org/jucs_14_19/model_driven_approach_to.

[51] Dr. Mark Klein and Prof. Chrysanthos Dellarocas. A knowledge-based approach to handling exceptions in workflow systems. *Journal of Computer Supported Collaborative Work, Special Issue on Adaptive Workflow Systems*, pages 1–11, 2000. URL http://cci.mit.edu/klein/papers/cscw-99.pdf.

[52] Lin Chen, Minglu Li Jian, and Cao Yi Wang. An eca rule-based workflow design tool for shanghai grid. *Services Computing (SCC05).*, 2005. URL http://ieeexplore.ieee.org/xpls/abs_all.jsparnumber=1531271tag=1.

[53] Macro Comuzzi and Irene T.P. Vanderfeesten. Product based workflow design for monitoring of collaborating buisness processes. *CAiSE*,

pages 154–168, 2011. URL http://link.springer.com/chapter/10.1007%2F978-3-642-21640-4_13#page-1.

[54] Therani Madhusudan. An agent-based approach for coordinating product design workflows. *International journal of science direct Computers in Industry*, pages 235–239, 2005. URL http://dl.acm.org/citation.cfm?id=1672888.

[55] Hai Zhong, Kun Tao, and Xuejie Zhang. An approach to optimized resource scheduling algorithm for open-source cloud systems. *The Fifth Annual ChinaGrid Conference*, pages 124–129, 2010. URL http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=5563015.

[56] Y. Yang, K. Liu, J. Chen, X. Liu, D. Yuan, and H. Jin. An algorithm in swindew-c for scheduling transaction-intensive cost-constrained cloud workflows. *Proc. of 4th IEEE International Conference on e-Science*, pages 374–375, December 2008. URL http://www.ict.swin.edu.au/personal/xliu/papers/C%28eScience%29&_eScience08%28Ke%29.pdf.

[57] Mrs.S.Selvarani and Dr.G.Sudha Sadhasivam. Improved cost-based algorithm for task scheduling in cloud computing. *IEEE International Conference*, May 2010. URL http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=5705847.

[58] K. Liu, Y. Yang, J. Chen, X. Liu, D. Yuan, and H. Jin. A compromised-time- cost scheduling algorithm in swindew-c for instance-intensive cost-constrained workflows on cloud computing platform. *International Journal of High Performance Computing Applications*, 24:445–456, May 2010. URL http://www.ict.swin.edu.au/personal/xliu/papers/C%28eScience%29&_eScience08%28Ke%29.pdf.

[59] Suraj Pandey, LinlinWu, Siddeswara Mayura Guru, and Rajkumar Buyya. A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments. *Proceedings IEEE International Conference*, pages 400–407, May 2010. URL http://www.buyya.com/papers/AINA2010-PSOSched-Workflow.pdf.

[60] Maria Alejandra Rodriguez and Rajkumar Buyya. Deadline based resource provisioning and scheduling algorithm for scientific workflows on clouds.

IEEE Transactions On Cloud Computing, (2):222–235, May 2014. URL http://www.computer.org/csdl/trans/cc/preprint/06782394.pdf.

[61] Zhangjun Wu, Xiao Liu, Zhiwei Ni, Dong Yuan, and Yun Yang. A market-oriented hierarchical scheduling strategy in cloud workflow systems. *Journal of Supercomputing*, pages 222–235, 2011. URL http://faculty.ecnu.edu.cn/picture/article/2443/32/9a/e69148ae413b841a37228a6df5be/bd11cc93-7e68-4c46-8d45-e2ac334532d7.pdf.

[62] Henan Zhao and Rizos Sakellariou. An experimental investigation into the rank function of the heterogeneous earliest finish time scheduling algorithm. *Euro-Par 2003 Parallel Processing, ser. Lecture Notes in Computer Science*, pages 189–194, 2003. URL http://www.cs.man.ac.uk/~rizos/papers/europar03.pdf.

[63] Cui Lin and Shiyong Lu. Scheduling scientific workflows elastically for cloud computing. *IEEE 4th International Conference on Cloud Computing*, 2011. URL http://www.academia.edu/2055107/SchedulingScientificWorkflowsElasticallyforCloudComputing.

[64] Xiaofeng Wang, Chee Shin Ye, Rajkumar Buyya, and Jinshu Su. Optimizing the makespan and reliability for workflow applications with reputation and a look-ahead genetic algorithm. *Future Generation Computer Systems*, (27): 11241134, 2011.

[65] S. Binato. Grasp for job shop scheduling. *Essays and surveys on metaheuristics*, pages 59–79, 2001. URL http://link.springer.com/article/10.1007%2FBF01096763.

[66] J. Blythe, S. Jain, and E. Deelman. Task scheduling strategies for workflow-based applications in grids. *CCGrid, IEEE*, pages 759–767, 2005. URL http://dl.acm.org/citation.cfm?id=1169581.

[67] D. P. Spooner, J. Cao, S. A. Jarvis, L. He, and G. R. Nudd. Performance-aware workflow management for grid computing. 2005. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.101.9279.

[68] R. Prodan and T. Fahringer. Dynamic scheduling of scientific workflow applications on the grid using a modular optimisation tool. *A Case Study, the 20th Symposium of Applied Computing*, pages 687–694, 2005. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.63.2693.

[69] P.Varalakshmi, Aravindh Ramaswamy, Aswath Balasubramanian, and Palaniappan Vijaykumar. An optimal workflow based scheduling and resource allocation in cloud. *International Conference*, pages 411–420, 2011. URL http://link.springer.com/chapter/10.1007/978-3-642-22709-7_41#page1.

[70] Saeed Parsa and Reza Entezari-Maleki. Rasa: A new task scheduling algorithm in grid environment. *World Applied Sciences Journal 7 Special Issue of Computer, IT*, pages 152–160, 2009. URL http://idosi.org/wasj/wasj7(candit)/20.pdf,origin=publication_detail.

[71] N. Metropolis et al. Equations of state calculations by fast computing machines. *Journal of Chemistry and Physics*, (21):1087–1091, 1953. URL http://www.stat.cmu.edu/~acthomas/724/Metropolis.pdf.

[72] A. YarKhan and J. J. Dongarra. Experiments with scheduling using simulated annealing in a grid environment. *International Workshop on Grid Computing*, pages 232–242, 2002. URL http://link.springer.com/chapter/10.1007/3-540-36133-2_21#page-1.

[73] Laurie Young, Stephen McGough, Steven Newhouse, and John Darlington. Scheduling architecture and algorithms within the iceni grid middleware. *UK e-Science All Hands Meeting*, pages 5–12, 2003. URL http://www.nesc.ac.uk/events/ahm2003/AHMCD/pdf/005.pdf.

[74] Monir Abdullah and Mohamed Othman. Simulated annealing approach to cost-based multi- quality of service job scheduling in cloud computing enviroment. *American Journal of Applied Sciences*, 11:872–877, 2014. URL http://thescipub.com/PDF/ajassp.2014.872.877.pdf.

[75] Weiwei Chen and Ewa Deelman. Integration of workflow partitioning and resource provisioning. *12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pages 764–768, 1953. URL http://www.stat.cmu.edu/acthomas/724/Metropolis.pdf.

[76] Ian Taylor. Pegasus workflow management system. URL http://pegasus.isi.edu/cloud/.

[77] . URL http://code.google.com/p/hamake/wiki/HamakeManual.

[78] Yahoos workflow engines for hadoop. . URL http://yahoo.github.com/oozie/.

[79] Cascading workflow. . URL http://www.cascading.org/.

[80] Trial version of orangescape. . URL https://www.orangescape.com.

[81] Workflow engines. . URL ftp://ftp.cordis.europa.eu/pub/transport/docs/summaries/integrated_octopus_report.pdf.

[82] W.M.P. van der Aalst and A.H.M. ter Hofstede. Yawl: Yet another workflow language. pages 1–44. URL http://www.yawlfoundation.org/documents/yawlrevtech.pdf.

[83] R. Buyya and M. Murshed. Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing. *Concurrency and Computation: Practice and Experience*, 14:1175–1220, 2002. URL http://arxiv.org/ftp/cs/papers/0203/0203019.pdf.

[84] Calheiros RN, Ranjan R, Beloglazov A, Rose CAFD, and Buyya R. Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience*, 41:23–25, 2011. URL http://www.buyya.com/papers/CloudSim2010.pdf.

[85] B. Wickremasinghe, R. Calheiros, and R. Buyya. Cloudanalyst: A cloudsim-based visual modeller for analysing cloud computing environments and applications. *Proceedings of the 24th IEEE International Conference on Advanced Information Networking and applications*, pages 446–452, 2010.

[86] Rodrigo. N., M .A Calheiros, S. Netto, C. A. F. De Rose, and R. Buyya. Emusim: An integrated emulation and simulation environment for modeling, evaluation, and validation of performance of cloud computing applications. *Software: Practice and Experience*, pages 1–18, 2012. URL http://www.buyya.com/papers/EMUSIM-SPE.pdf.

[87] W. Chen and E. Deelman. Workfowsim: A toolkit for simulating scientific workflows in distributed environments. *E-Science, 2012 IEEE 8th International Conference*, pages 1–8, 2012. URL https://www.ci.uchicago.edu/escience2012/pdf/WorkflowSim-AToolkitforSimulating_Scientific_Workflows_in_Distibuted_Environments.pdf.

[88] Chtepen M, Dhoedt B, Cleays F, and Vanrolleghem. Evaluation of replication and rescheduling heuristics for gird systems with varying resource availability. *Proceedings of 18th international conference on parallel and distributed computing systems, Anaheim*, pages 622–627, 2006.

[89] Rajkumar Buyya, Chee Shin Yeo, Srikumar Venugopal, James Broberg, and Ivona Brandic. Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems*, 25:599–616, 2009. URL http://www.sciencedirect.com/science/article/pii/S0167739X08001957.

[90] R.and Prodan R. Duan and T. Fahringer. Data mining-based fault prediction and detection on the grid. *Proceedings of the 15th IEEE International Symposium on High Performance Distributed Computing*, pages 305–308, 2006. URL http://www.computer.org/csdl/proceedings/hpdc/2006/0307/00/01652162-abs.html.

[91] Jitsumoto H., T. Endo, Matsuoka, and S. ABARIS. An adaptable fault detection/recovery component framework for mpi. *Proceedings of the IEEE International Parallel and Distributed Processing Symposium,IEEE Computer Society Press*, pages 1–8, 2007. URL file:///C:/Users/Administrator/Downloads/0046351ed37e68b3b8000000.pdf.

[92] S. Fu and C.-Z. Xu. Exploring event correlation for failure prediction in coalitions of clusters. *Proceedings of the 2007 ACM/IEEE conference on Supercomputing*, 41:1–12, 2007. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.122.8727&rep=rep1&type=pdf.

[93] S. Fu and C.-Z. Xu. Quantifying event correlations for proactive failure management in networked computing systems. *Journal of Parallel and Distributed Computing*, 70:1100– 1109, 2010. URL http://www.sciencedirect.com/science/article/pii/S0743731510001218.

[94] E. Sindrilaru and Costan A. Cristea. Fault tolerance and recovery in grid workflow management systems. *International Conference on Complex Intelligent and Software Intensive Systems*, pages 475–480. URL http://dl.acm.org/citation.cfm?id=1796176.1796525.

[95] Z. Yu, C. Wang, and W. Shi. Flaw: Failure-aware workflow scheduling in high performance computing systems. *Journal of Cluster Computing*,

13:421 –434, 2010. URL http://www.cs.wayne.edu/~weisong/papers/yu07-flaw.pdf.

[96] Q. Guan, Zhang Z., and F. Song. A failure detection and prediction mechanism for enhancing dependability of data centers. *International Journal of Computer Theory and Engineering*, 4:726–730, 2012. URL http://www.techrepublic.com/resource-library.

[97] R. Jhawar, V. Piuri, and M.D. Santambrogio. A comprehensive conceptual system-level approach to fault tolerance in cloud computing. *EEE Int. Syst. Conf.*, pages 1–5, 2012. URL http://spdp.di.unimi.it/papers/JPS.SysCon2012.pdf.

[98] Q. Guan, Z. Zhang, and F. Song. Ensemble of bayesian predictors for autonomic failure management in cloud computing. *IEEE international conference*, pages 1–6, 2011. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6006036.

[99] Deepak Poola, Saurabh Kumar Garg, Rajkumar Buyya, Yun Yang, and Kotagiri Ramamohanara. Robust scheduling of scientific workflows with deadline and budget constraints in clouds. *2014 IEEE 28th International Conference on Advanced Information Networking and Applications*, pages 858–865, 2014. URL http://buyya.com/papers/Cloud-WorkflowSched-AINA2014.pdf.

[100] C. Catal. Software fault prediction: A literature review and current trends. *Expert Systems With Applications*, 38(11), 2011. URL http://www.sciencedirect.com/science/article/pii/S0957417410011681.

[101] R. Malhotra and A. Jain. Fault prediction using statistical and machine learning methods for improving software quality. *Journal of Information Processing Systems*, 8:241–262, 2012. URL http://ocean.kisti.re.kr/downfile/volume/kips/E1JBB0/2012/v8n2/E1JBB0_2012_v8n2_241.pdf.

[102] N. Ohlsson, M. Zhao, and M. Helander. Application of multivariate analysis for soft ware fault prediction. *Software Quality Journal*, 7:51–66, 1998. URL http://link.springer.com/article/10.1023/B:SQJO.0000042059.16470.f0.

[103] Y. Suresh, L. Kumar, and Rath. Ku. S. Statistical and machine learning methods for software fault prediction using ck metric suite:a comparative analysis. *ISRN Sofware Engineering*, pages 1–16, 2014.

[104] A. Islam, J. Keunga, K. Lee, and Liu. A. empirical prediction models for adaptive resource provisioning in the cloud. *Future Generation Computer Systems*, 28:155–162, 2012. URL http://www.sciencedirect.com/science/article/pii/S0167739X11001129.

[105] G. Kousiouris, A. Menychtasa, D. Kyriazis, S. Gogouvitis, and T. Varvarigou. Dynamic behavioral-based estimation of resource provisioning based on high-level application terms in cloud platforms. *Future Generation Computer Systems*, 32:27–40. URL http://www.sciencedirect.com/science/article/pii/S0167739X12001057.

[106] M.Armbrust, A.Fox, and R. Griffit. A view of cloud computing. *Communications of the ACM*, 53:50–58, 2010. URL http://mars.ing.unimo.it/didattica/ingss/Lec_SaS/CloudView.pdf.

[107] Zaipeng Xie, Hongyu Sun, and Kewal Saluja. A survey of software fault tolerance techniques. *International Conference*, pages 1–10, 2006. URL http://www.pld.ttu.ee/IAF0030/Paper_4.pdf.

[108] Geoffroy Vallee and Anand Tikotekar Stephen L. Scott Kulathep Charoenpornwattana, Christian Engelmann. A framework for proactive fault tolerance. *International Conference*, 2008. URL http://www.christian-engelmann.info/publications/vallee08framework.ppt.pdf.

[109] Kassian Plankensteiner and Thomas Fahringer Radu Prodan. New fault tolerance heuristic for scientific workflows in highly distributed environments based on resubmission impact. *Fifth IEEE International Conference on e-Science*, 2009. URL http://libra.msra.cn/Publication/50852578.

[110] Anton Beloglazov and Rajkumar Buyya. Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers. *Journal of concurrency Computation: Practices and Experiences*, 24:1397–1420, 2012. URL http://beloglazov.info/papers/2012-ccpe-vm-consolidation-algorithms.pdf.

[111] Abdi H. Multiple correlation coefficients. *Encyclopedia of Measurement and Statistics*, pages 648–651, 2007.

[112] Jing Tai Piao and Jun Yan. A network-aware virtual machine placement and migration approach in cloud computing. *Ninth International Conference on Grid and Cloud Computing IEEE*, pages 87–92, November 2010. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5662521&tag=1.

[113] Gideon Juve, Ann Chervenak, Ewa Deelman, Shishir Bharathi, Gaurang Mehta, and Karan Vahi. Characterizing and profiling scientific workflows. *Future Generation Computer Systems*, 29:682–692, 2013. URL http://www.sciencedirect.com/science/article/pii/S0167739X12001732.

[114] Hall M., Frank E., Holmes G., Pfahringer B., Reutemann P., and Witten I.H. The weka data mining software: an update. *SIGKDD Explorations*, (11): 10–19, 2009. URL http://www.cms.waikato.ac.nz/~ml/publications/2009/weka_update.pdf.

[115] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield. Live migration of virtual machines. *Proceedings of the 2nd Symposium on Networked Systems Design and Implementation (NSDIS).*, 2: 273–286, 2005. URL http://dl.acm.org/citation.cfm?id=1251223.

[116] Glass M, Lukasiewycz M, Reimann F, Haubelt C, and Teich J. Symbolic reliability analysis of self-healing networked embedded systems,. *Proceedings of the 27th international conference on computer safety, reliability, and security. Springer*, pages 139–152, 2008. URL https://www12.informatik.uni-erlangen.de/publications/pub2008/GLRHT08b.pdf.

[117] Glass M, Lukasiewycz M, Streichert T, Haubelt C, and Teich J. Reliability-aware system synthesis, design, automation and test. *Europe Conference and Exhibition*, pages 1–6, 2006. URL https://www12.informatik.uni-erlangen.de/publications/pub2007/GLSHT07b.pdf.

[118] Koch D, Streichert T, Dittrich S, Strengert C, Haubelt C, and Teich. An operating system infrastructure for fault-tolerant reconfigurable networks. *Proceedings of the 19th international conference on architecture of computing systems, Germany. Springer*, pages 202–212, 2006. URL http://link.springer.com/chapter/10.1007.

[119] Cheng SW, Garlan D, and Schmerl B. Architecture-based self-adaptation in the presence of multiple objectives. *SEAMS 06: ACM*, pages 2–8, 2006. URL http://www.irisa.fr/lande/lande/icse-proceedings/seams/p2.pdf.

[120] E.M. Dashofy, A.V.D. Hoek, and R.N. Taylor. Towards architecture based self-healing systems. *Proceedings of the First Workshop on Self-Healing Systems*, pages 21–26, 2002. URL http://www.ics.uci.edu/~andre/papers/C23.pdf.

[121] Jason Nichols. Autonomic workflow execution in the grid. *IEEE transactions on systems, man, and cyberneticspart c: applications and review*, 36, 2006. URL http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=1629200.

[122] I. Georgiadis, J. Magee, and J. Kramer. Self-organizing software architectures for distributed systems. *Proceedings of the First Workshop on Self-Healing Systems*, pages 33–38, 2002. URL http://delivery.acm.org/10.1145/590000/582135/p33-georgiadis.pdf.

[123] J. Aldrich, V. Sazawal, C. Chambers, and D. Nokin. Architecture centric programming for adaptive systems. *Proceedings of the First Workshop on Self-Healing Systems*, 2002. URL http://archjava.fluid.cs.cmu.edu/papers/woss02.pdf.

[124] C. Dabrowski and K.L. Mills. Understanding self-healing in service discovery systems. *Proceedings of the First Workshop on Self-Healing Systems*, pages 15–20, 2002. URL http://www.itl.nist.gov/div897/ctg/adl/adl_files/Dabowski&Mills-WOSS2002final.pdf.

[125] M.M. Rakic, N.R. Mehta, and N. Medvidovic. Architectural style requirements for self-healing systems. *Proceedings of the First Workshop on Self-Healing Systems*, pages 49–54, 2002. URL http://sunset.usc.edu/~mehta/Prism.pdf.

[126] Holderfield V and Huhns M. A foundational analysis of software robustness using redundant agent collaboration. *Lecture notes in computer science*, page 355369, 2003. URL http://link.springer.com/chapter/10.1007/3-540-36559-1_26#page-1.

[127] Corsava S and Getov V. Intelligent architecture for automatic resource allocation in computer clusters. *Proceedings of the 17th international symposium*

on parallel and distributed processing. IEEE Computer Society, pages 1–8, 2003. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1213369&tag=1.

[128] Huhns MN, Holderfield VT, and Gutierrez RLZ. Robust software via agent-based redundancy. *Proceedings of the second international joint conference on autonomous agents and multiagent systems. ACM*, pages 118–119, 2003. URL http://www.cse.sc.edu/~huhns/confpapers/AAMAS2003.pdf.

[129] Tesauro G, Chess DM, Walsh WE, Das R, Segal A, Whalley I, Kephart JO, and White SR. A multi-agent systems approach to autonomic computing. *Proceedings of the third international joint conference on autonomous agents and multiagent systems. IEEE Computer Society*, pages 464–471, 2004. URL http://dl.acm.org/citation.cfm?id=1018780.

[130] Raheel Ahmad, Yung-Chuan Lee, Shahram Rahimi, and Bidyut Gupta. A multi-agent based approach for particle swarm optimization. *Integration of Knowledge Intensive Multi-Agent Systems, IEEE*, pages 267–271, 2007. URL http://opensiuc.lib.siu.edu/cgi/viewcontent.cgi?article=1016&context=cs_pubs.

[131] Rizvan Erola, Cenk Sahina, Adil Baykasoglub, and Vahit Kaplanogluc. A multi-agent based approach to dynamic scheduling of machines and automated guided vehicles in manufacturing systems. *Journal of Applied Soft Computing*, pages 1720–1732, 2012. URL http://www.sciencedirect.com/science/article/pii/S1568494612000543.

[132] Jeongmin Park, Hyunsang Youn, and Eunseok Lee. A multi-agent based context aware self-healing system. *Intelligent Data Engineering and Automated Learning-IDEAL 2005. Springer Berlin Heidelberg*, pages 515–523, 2005. URL http://link.springer.com/chapter/10.1007%2F11508069_67#page-1.

[133] Modafferi S and Conforti E. Methods for enabling recovery actions in ws-bpel. *Lect Notes in Computer Science*, http://link.springer.com/chapter/10.1007:210–219, 2006.

[134] Modafferi S, Mussi E, and Pernici B. Sh-bpel self-healing plug-in for ws-bpel engine. *Proceedings of the 1st workshop on middleware for service oriented*

computing MW4SOC 2006, pages 48–53, 2006. URL http://dl.acm.org/citation.cfm?id=1169099.

[135] Xiangping Bu, Jia Rao, and Cheng-Zhong Xu. Coordinated self-configuration of virtual machines and appliances using a model-free learning approach. *International Conference*, pages 1–2, 2012. URL http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6216363.

[136] Baresi L and Guinea S. Dynamo and self-healing bpel compositions. *29th International conference on software engineering-companion, ICSE 2007*, pages 69–70, 2007. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4222685.

[137] Baresi L, Guinea S, and Pasquale L. Self-healing bpel processes with dynamo and the jboss rule engine. *International workshop on Engineering of software services for pervasive environments. ACM*, pages 11–20, 2007. URL http://dl.acm.org/citation.cfm?id=1294906.

[138] Moo-Mena F, Garcilazo-Ortiz J, Basto-Daz L, Curi-Quintal F, and Alonzo-Canul F. Defining a self-healing qos-based infrastructure for web services applications. *Proceedings of the 2008 11th IEEE international conference on computational science and engineering-workshops. IEEE Computer Society*, pages 215–220, 2008. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4625064&tag=1.

[139] Halima RB, Drira K, and Jmaiel. A qos-oriented reconfigurable middleware for self-healing web services. *Proceedings of the 2008 IEEE international conference on web services. IEEE Computer Society*, pages 104–111, 2008. URL http://homepages.laas.fr/khalil/wiki/uploads/THESES/PapersRiadh/ICWS08.pdf.

[140] Mahmoud Hossein Zadeh and Mir Ali Seyyedi. A self-healing architecture for web services based on failure prediction and a multi agent system. *IEEE international conference*, pages 48–52, 2011. URL http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6041420.

[141] Dai Y, Yang L, and Zhang B. Qos-driven self-healing web service composition based on performance prediction. *Journal of computer science and technology*, 24:250261, 2009. URL http://link.springer.com/article/10.1007%2Fs11390-009-9221-8#page-1.

[142] Marko Kankaanniemi. Self-optimization in autonomic systems. *International Conference of Innovations in Information technology (IIT).*, pages 1–6, 2011. URL http://www.cs.helsinki.fi/u/niklande/opetus/SemK07/paper/kankaanniemi.pdf.

[143] Benoit Gaudin, Emil I. Vassev, Michael G. Hinchey, and Patrick Nixon. A control theory based approach for self-healing of un-handled runtime exceptions. *ICAC*, 2011. URL http://dl.acm.org/citation.cfm?id=1998633.

[144] Qianxiang Wang. Towards a rule model for self-adaptive software. *ACM SIGSOFT Software Engineering Notes*, 30, 2005. URL http://dl.acm.org/citation.cfm?id=1039198.

[145] Mohamed Boukhebouze and Youssef Amghar. Towards self-healing execution of business processes based on rules. *ICEIS 2009*, pages 510–512, 2009. URL http://www.researchgate.net/publication/220711746_Towards_Self-healing_Execution_of_Business_Processes_Based_on_Rules.

[146] Wlodzimierz Funika and Piotr Piegiel. A role-based approach to self-healing in autonomous monitoring systems. *PPAM 2009*, pages 125–134, 2009. URL http://link.springer.com/chapter/10.1007%2F978-3-642-14403-5_14$#$page-1.

[147] Michael E. Shin and Daniel Cooke. Connector-based self-healing mechanism for components of a reliable system. *DEAS, St. Louis Missouri*, pages 1–6, 2005. URL https://www.netlab.tkk.fi/opetus/s384030/k06/papers/ConnectorBasedSeflHealingMechanism.pdf.

[148] Jia Rao, Xiangping Bu, and George Yin. Vconf:a reinforcement learning approach to virtual machines auto- configuration. *ICAC09: Barcelona, Spain*, pages 137–146, 2009. URL http://cs.uccs.edu/~jrao/papers/ICAC09.pdf.

[149] Stefania Montani and Cosimo Anglano. Achieving self-healing in service delivery software systems by means of case-based reasoning. *journal of Springer Science,Business Media Applied Intelligence*, pages 139–152, 2008. URL http://link.springer.com/article/10.1007%2Fs10489-007-0047-1#page-1.

[150] R.H.Bordani. A self healing approach to designing and deploying complex, distributed and concurrent software systems. *International journal of springer*, 44:3–13, 2007.

[151] Teemu Kemppainen. Biology-inspired self-healing system design. *Seminar on self-healing information systems, springer*, pages 1–6, 2007. URL http://www.cs.helsinki.fi/u/niklande/opetus/SemK07/paper/kemppainen.pdf.

[152] C. Catal and Diri. A systematic review of software fault prediction studies. *Expert Systems with Applications*, 36:7346–7354, 2009. URL http://www.sciencedirect.com/science/article/pii/S0957417408007215.

[153] Haproxy tool. 2012. URL http://www.haproxy.org/.

[154] S. Sidiroglou, O. Laadan, C. Perez, N. Viennot, J. Nieh, and A. D. Keromytis. Assure: Automatic software self-healing using rescue points. *Proceedings of the 14th International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 37–48, 2009. URL http://people.csail.mit.edu/stelios/papers/assure_asplos.pdf.

[155] Gang Chen, Hai Jin, Deqing Zou, Bing Bing Zhou, Weizhong Qiang, and Gang Hu. Shelp: Automatic selfhealing for multiple application instances in a virtual machine environment. *IEEE International Conference on Cluster Computing*, pages 97–106, 2010. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5600316&tag=1.

[156] Hadoop map reduce framework. 2012. URL HadoopMapReduceTutorial. http://hadoop.apache.org/core/docs/current/mapredtutorial.html.

[157] Amazon ec2. 2012. URL AmazonElasticComputeCloud(EC2)http://www.amazon.com/ec2/.

[158] Antonina Litvinova, Christian Engelmann, and Stephen L. Scott. A proactive fault tolerance framework for high performance computing. 2009. URL http://www.christian-engelmann.info/publications/litvinova10proactive.ppt.pdf.

[159] Zheng Q and Veeravalli B. On the design of communication-aware fault-tolerant scheduling algorithms for precedence constrained tasks in grid

computing systems with dedicated communication devices. *Journal of Parallel Distributed Computing*, 69(3):282–294, 2009. URL http://www.sciencedirect.com/science/article/pii/S0743731508002001.

[160] Yang Zhang, Anirban Mandal, Charles Koelbel, and Keith Cooper. Combined fault tolerance and scheduling techniques for workflow applications on computational grids. *9th IEEE/ACM international symposium on clustering and grid*, pages 244–251, 2010. URL http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=5071878.

[161] Deepak Poola, Kotagiri Ramamohanarao, and Rajkumar Buyya. Fault-tolerant workflow scheduling using spot instances on clouds. *Procedia Computer Science*, 29(3):523–533, 2014. URL http://www.buyya.com/papers/Cloud-FTWorkflowSched-ICCS2014.pdf.

[162] Elvin Sindrilaru, Alexandru Costan, and Valentin Cristea. Fault tolerance and recovery in grid workflow management systems. *International Conference on Complex, Intelligent and Software Intensive Systems*, pages 1–6, 2010. URL http://airccse.org/journal/ijcses/papers/0211cses07.pdf.

[163] Zhifeng Yu and Weisong Shi. A planner-guided scheduling strategy for multiple workflow applications. *International Conference on Parallel Processing*, pages 1–8, 2008. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4626773.

[164] Tim Dornemann, Ernst Juhnke, and Bernd Freisleben. Security,fault tolerance and modeling of grid workflows in bpel4grid. *Lecture notes*, 2010. URL https://bi.offis.de/bisgrid/bi.offis.de/bisgrid/tiki-download_file636d.pdf?fileId=331.

[165] Soonwook Hwang and Carl Kesselman. A flexible framework for fault tolerance in the grid. *Journal of Grid Computing*, pages 251–271, 2003. URL http://link.springer.com/article/10.1023%2FB%3AGRID.0000035187.54694.75#page-1.

[166] Chtepen M, Dhoedt B, Cleays F, and Vanrolleghem. Evaluation of replication and rescheduling heuristics for gird systems with varying resource availability. *Proceedings of 18th international conference on parallel and distributed computing systems*, 29:622627, 2006. URL http://www.researchgate.net/publication/228966331.

[167] Weiwei Chen and Ewa Deelman. Fault tolerant clustering in scientific workflows. pages 1–8, 2012. URL http://pegasus.isi.edu/publications/2012/WChen-FTC-Services12.pdf.

[168] Mustafizur Rahman. Autonomic workflow management system for grid computing. *Ph.d.thesis,The University of Melbourne, Australia*, 2010. URL http://www.cloudbus.org/students/RahmanPhDThesis2010.pdf.

[169] M. Xie, Y.S. Dai, and K.L. Poh. Computing system reliability: Models and analysis. *Kluwer :Academic Publishers*, pages 1–17. URL http://www.netlib.org/utk/people/JackDongarra/PAPERS/Cloud-Shaun-Jack.pdf.

[170] B. Varghese, G. McKee, and V. Alexandrov. Intelligent agents for fault tolerance: From multi-agent simulation to cluster-based implementation. *24th international Conference IEEE*, pages 985–990, 2010. URL http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=5480939.

[171] A. Abramovici, W. Althouse, R. Drever, Y. Grsel, S. Kawamura, F. Raab, D. Shoemaker, L. Sievers, R. Spero, R. Weiss S. Whitcomb K. Thorne, R. Vogt, and M. Zucker. Ligo: The laser interferometer gravitational-wave observatory. *Science*, 256:325–333, 1992. URL http://iopscience.iop.org/0034-4885/72/7/076901/.

[172] J. Livny, H. Teonadi, M. Livny, and M. K. Waldor. High-throughput, kingdom-wide prediction and annotation of bacterial non-coding rnas. *PLoS ONE*, 3:3197, 2008. URL http://www.ncbi.nlm.nih.gov/pubmed/18787707.

[173] G. Juve and E. Deelman. Scientific workflows in the cloud. *Cloud Chapter*, pages 56–74, 2010. URL http://isi.edu/~gideon/publications/JuveG-CloudChapter.pdf.

[174] G. et al. Juve. Data sharing options for scientific workflows on amazon ec2. *ACM/IEEE International Conference on HPC, Networking, Storage*

and Analysis, pages 1–9, 2010. URL http://delivery.acm.org/10.1145/1890000/1884693/75590050.pdf.

[175] D. Hosmer and S. Lemeshow. Applied logistic regression. *John Wiley and Sons*, 1989. URL http://onlinelibrary.wiley.com/book/10.1002/9781118548387.

[176] L. Breiman. Random forests. *Machine Learning*, 45:1–33, 2001. URL http://oz.berkeley.edu/~breiman/randomforest2001.pdf.

[177] L. Guo, Y. Ma, B. Cukic, and H. Singh. Robust prediction of fault-proneness by random forests. *Proceedings of the 15th International Symposium on Software Reliability Engineering*, pages 417–428, 2004. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1383136.

[178] F. Salfner, M. Lenk, and M. Malek. A survey of online failure prediction methods. *ACM Computing Surveys*, pages 1–42, 2010. URL https://informatik.hu-berlin.de/Members/salfner/publications/salfner10survey.pdf.

[179] Sanjay Silakari Kamlesh Gupta. Novel approach for fast compressed hybrid color image cryptosystem. *Advances in Engineering Software, Elsevier*, 49:29–42, 2012. URL http://www.sciencedirect.com/science/article/pii/S0965997812000464.

List of Publications

International Journals (indexed by SCI)

1. Anju Bala, Inderveer Chana (2015). Intelligent Failure Prediction Models for Scientific Workflows, Expert Systems with Applications, Elsevier Publications, Volume 42, Issue 3, pp. 980-989, 2015, DOI: 10.1016/j.eswa.2014.09.014, Impact Factor: 1.965.

2. Anju Bala, Inderveer Chana (2015). Autonomic Fault Tolerant Scheduling Approach for Scientific Workflows in Cloud Computing, Concurrent Engineering: Research and Applications, Sage Publications, pp 1-13, 2015, DOI: 10.1177/1063293X14567783, Impact Factor: 0.531.

International Journals (Non-SCI with Impact Factor)

1. Anju Bala,Inderveer Chana (2012). Fault Tolerance-Challenges, Techniques and Implementation in Cloud Computing, International Journal of Computer Science Issues, Volume 9, Issue 1, pp. 288-293, 2012, Impact Factor:0.242, Citations:21.

2. Anju Bala,Inderveer Chana (2012). Design and Deployment of Workflows in Cloud Environment, International Journal of Computer Applications, Volume 51, Issue 11, pp.9-15, 2012, Impact Factor:0.8, Citations:02.

3. Anju Bala,Inderveer Chana (2011). A Survey of Various Workflow Scheduling Algorithms in Cloud Environment, International Journal of Computer Applications, Proceedings on 2nd National Conference on Information and Communication Technology, Volume NCICT4, pp. 26-30, 2011, Impact Factor:0.8, Citations:31.

International Conferences

1. Anju Bala, Inderveer Chana (2013). VM Migration Approach for Autonomic Fault Tolerance in Cloud Computing, International Conference of Grid and Cloud Applications GCA13, USA, Las Vegas, pp. 3-10, 2013, Indexed by Scopus, Citations:04

2. Anju Bala, Inderveer Chana (2015). Multi-level Priority Based Task Scheduling Algorithm for Workflows in Cloud Environment, ICECECE 2015 : International Conference on Electrical, Computer, Electronics and Communication Engineering, New Delhi, India, February, 7-8, 2015-*(Accepted)*, Indexed by Scopus

Papers Communicated

1. Anju Bala, Inderveer Chana (2015).Prediction Based Proactive Load Balancing Approach through VM migration, Knowledge and Information Systems, Springer, Impact Factor:2.67, **(SCI indexed)**- *Communicated.*

2. Anju Bala, Inderveer Chana (2015). Autonomic Cloud Computing: Existing Techniques and Future Directions, Concurrency and Computation: Practice and Experience, Wiley, Impact factor: 0.784, **(SCI indexed)**- *Communicated.*

3. Anju Bala, Inderveer Chana (2015). Prediction Based Proactive Fault Tolerant Approach , Computing and Informatics, Impact Factor:0.319, **(SCI indexed)**- *Communicated.*