Tomas Bata University in Zlín
Faculty of Applied Informatics

Doctoral Thesis Summary

Fault Tolerance for Big Data Scientific Workflows in Cloud Computing Environments

Spolehlivost a odolnost vůči poruchám cloudových systémů pro automatické řízení a koordinaci procesů zpracování rozsáhlých a heterogenních vědecko technických dat

Author:	**Ammar Nassan Alhaj Ali**
Degree programme:	P3903 Engineering Informatics
Degree Course:	3902V037E Engineering Informatics
Supervisor:	Prof. Said Krayem

Zlin, June 2021

Published by **Tomas Bata University in Zlín** in the Edition **Doctoral Thesis Summary.**

The publication was issued in the year 2021.

Klíčová slova: *odolnost vůči poruchám, spolehlivost, rozsáhlá a heterogenní data, workflow, cloud.*

Keywords**:** *Fault Tolerance, reliability, Big Data, workflow, cloud.*

Full text of the scientific publication is available in the Library of TBU in Zlín.

ISBN 978-80-……

Abstract

Past few years, Big Data and cloud computing have become buzzwords in IT region, and we have been seeing that data are generated in massive amounts and at an increasing rate in all domains. The reliability and efficiency of distributed systems have always been a major concern of the service providers and users. Therefore, fault tolerance is among the most essential issues in distributed clouds to deliver reliable services to customers.

In Big Data domain, scientific workflows are increasingly used for Big Data analysis, processing, and management. With movement the world to Big Data, single-site processing becomes unsuitable and Big Data scientific workflows can no longer be accommodated within a single computing system, and ensuring a level of reliability for a scientific workflow execution is a complex task that will tend to increase the cost.

Replication of tasks increases redundancy and thereby the reliability, which is achieved by parallel execution of a task on multiple virtual machines simultaneously to guarantee a viable result, which leads to a high cost.

This doctoral Thesis presents a fault-tolerant model with two approaches that optimize the reliability and execution cost of Big Data scientific workflows on cloud computing environments and ensure a predefined level of reliability by replicating tasks.

Finally, the model was implemented using WorkflowSim, it is extension of the CloudSim simulator framework that is used for modelling and simulation of cloud computing infrastructures and services.

Abstrakt

V posledních několika letech se v oblasti IT staly hesly Big Data a cloud computing a my jsme svědky toho, že data jsou generována resp. zpracovávána v obrovských objemech a stále rychleji ve všech oblastech. Spolehlivost a bezpečnost distribuovaných systémů byla vždy hlavním zájmem poskytovatelů služeb i uživatelů. Proto patří odolnost proti výpadkům a chybám mezi klíčové požadavky na provoz cloudových systému, podmiňující spolehlivost a použitelnost služeb pro zákazníky.

V oblasti velkých dat se pro analýzu, zpracování a správu velkých dat stále častěji používají metody vědeckotechnické analýzy (matematické a statistické metody, aplikace umělé inteligence apod). S přechodem uživatelských aplikací ke zpracování velkých dat je použití distribuovaných cloudových řešení stále častěji jediným ekonomicky přijatelným řešením.

Jednou z metod, které cloudový systém nabízí je replikace úloh, která zvyšuje redundanci, a tím i spolehlivost paralelním prováděním úlohy na více virtuálních strojích současně. Tak lze zaručit přijatelné řešení, avšak za cenu vysokých nákladů.

Tato disertační práce představuje model odolný proti poruchám se dvěma přístupy, které optimalizují spolehlivost a náklady na provádění vědeckých pracovních postupů s velkými objemy dat v prostředí cloudového systému a zajišťují předem definovanou úroveň spolehlivosti replikací úloh.

Navržený model byl implementován pomocí WorkflowSim, což je rozšíření simulátorového rámce CloudSim, který se používá pro modelování a simulaci infrastruktur a služeb cloudovho systému.

CONTENTS

1. INTRODUCTION

Cloud computing systems have become mature sufficient to manage and handle a huge volume of heterogeneous data that is rapidly changing. However, failures are unavoidable in cloud computing systems as they are composed of a large number of hardware resources (e.g., CPU, storage, and network).

Scientific workflows allow users easily to express multi-step computational tasks, for example, retrieve data from a database, reformat the data, and run an analysis. A scientific workflow usually describes the dependencies between the tasks. In most cases, the workflow is described as a directed acyclic graph (DAG), where the nodes are tasks and the edges denote data dependencies between tasks [1].

Scientific workflows demand massive resources from diverse computing infrastructures to process a massive amount of Big Data. Automatic provisioning of such Big Data applications on the cloud platform is challenging since current resource management and scheduling approaches may not be able to scale well, especially under highly dynamic conditions [2].

Fault tolerance to failures is of major importance when running on cloud computing systems, where a predefined level of reliability is required for long-running applications and services, to ensure that level of reliability, the cloud should be uncommonly fault tolerant.

One of the best ways for increasing the reliability is by replication tasks. Task replication is a proficient technique in case of a task running on an unreliable execution environment. The goal of the replication is to ensure that at least one replica is always able to complete the computation in case the others fail [3].

On the other hand, the cost of reliability improvements are paid by a reduction in failure, this issue is not quite so simple for many failures, nevertheless, there is never an endless budget for improving the reliability and some consideration of cost is inevitable [4].

AWS, Microsoft Azure, and Google Cloud Platforms provide many services. One thing is constant over all companies: the cloud cost is a headache to predict and control. A fresh Forbes article included an interesting statistic that 30 percent of cloud spend is wasted! This waste is due to using duplicate services; give up services and reckless buying [5].

According to a survey by Spiceworks company (April 2018), the reliability and cost are at the top as extremely important factors when evaluating cloud-based IT services. (Figure 1.1) shows a priority of IT decision-makers when they buy services cloud.

This Thesis primarily addresses to optimize the reliability and execution cost of Big Data scientific workflows on cloud computing environments by a model with fault tolerance is offered. We propose two approaches to optimize the reliability and

cost of scientific workflows on cloud computing environments. In Addition, the thesis reviews former researches in the same field, and evaluate them by results comparison.

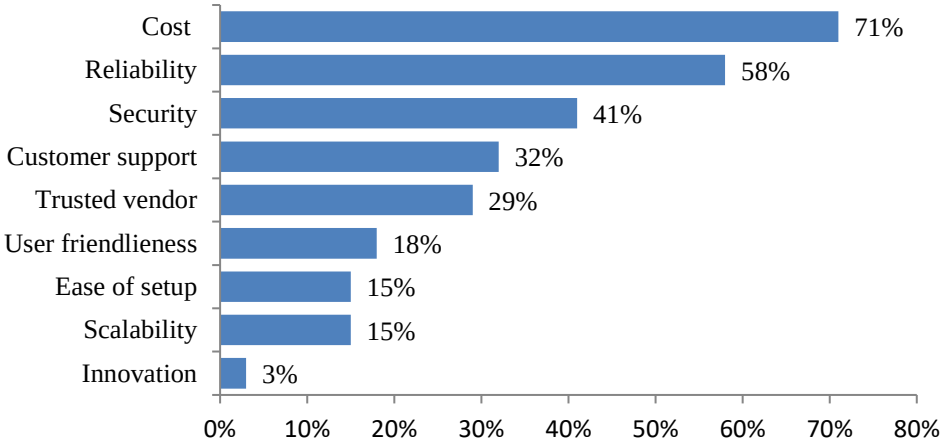


Fig. 1.1: Most important factors of cloud-based IT services 1

The rest of the thesis is organized as follows:

- Chapter 2 presents a state of the art of algorithms that have been proposed to improve the reliability and cost of performance in distributed environments.
- Chapter 3 presents the goals and benefits of the thesis.
- Chapter 4 proposes a fault tolerance model for scheduling and processing scientific workflows on computing cloud environments
- Chapter 5 presents the first approach which we proposed for scheduling scientific workflows on computing cloud environments using the genetic algorithm.
- Chapter 6 presents the second approach that guarantees predefined level of reliability for scheduling scientific workflows with minimum execution cost by the greedy algorithm.
- Chapter 7 provides our achievements in this Thesis.
- Chapter 8 concludes this thesis with a summary of contributions and the perspectives brought by our solutions.

[1] *https://www.spiceworks.com/ Diving into IT Cloud Services*

2. STATE OF THE ART

Although the cloud computing systems themselves promise high reliability, ensuring a high quality of service is still one of the challenging and critical research problems, and it has gained increasing attention recently [6].

The cloud computing systems are often made up of heterogeneous resources and ensuring reliability is a complex task, therefore fault tolerance mechanism operates as a backbone of distributed systems and has an important role in the reliability of enterprise distributed applications [7].

Many scheduling techniques and algorithms have been proposed to improve the reliability of performance in distributed environments, and ensuring a predefined level of reliability under various constraints such as task deadline or execution cost, and improving the economic aspect of scheduling in distributed systems.

Fault-Tolerant Scheduling Algorithm FTSA [8] is proposed which aims to tolerate multiple processor failures. FTSA is based on an active replication scheme to mask failures, in this approach we don't need for detecting and handling failures. Multiple copies of each task are mapped on different processors, which are run in parallel to tolerate a fixed number of failures. It assumes that some processors are reserved only for realizing fault tolerance, i.e., the reserved processors are not used for the original scheduling, and a static number of replicas are used of each task on processors.

In FTSA, the processor is selected for replication which has a minimum finish time. We can see that in FTSA we can increase reliability but sometimes we cannot satisfy the required reliability.

Other attempts for designing fault-tolerant systems by the use of replication have been made, the MaxRe algorithm (Max Reliability) [9] focused to satisfy the user's reliability requirement with minimum resources, and the number of replicas for each task should be as few as possible. The MaxRe algorithm transfer the reliability requirement of the workflow to the sub-reliability requirement of each task, and iteratively select available replicas and VMs with the maximum reliability value for each task to minimize the number of replicas, and thereby to reduce execution cost, until the sub-reliability of the task is satisfied.

The RR Algorithm (Reliability Requirement) [10] uses the same approach of MaxRe algorithm in selection VMs with the maximum reliability and transfers the reliability requirement of the workflow to the sub-reliability requirement, whereas the sub-reliability requirement of the entry task is still calculated same equation, but the sub-reliability requirement of other tasks is calculated by a different way.

Optimizing the makespan and reliability for workflow applications by genetic algorithm [11] has been proposed. The reliability-driven RD reputation can be used to effectively evaluate the reliability of a resource in distributed systems. And it

proposed the genetic algorithm which utilizes the RD reputation to optimize both the makespan and the reliability of a workflow.

A fault tolerant framework with deadline guarantee FTDG [12] has been proposed to achieve high fault tolerance and low response time in a Big Data stream computing environment, and obtain the conditions to meet the high reliability and low response time.

Another approach to enhance reliability on heterogeneous computing systems by the use of replication is proposed in [13], in this approach, the main objective is to propose a replication-based algorithm that maximizes the system reliability while considering the communication between tasks.

In [14], a model with dynamic fault tolerance is presented, which ensures the required reliability is met by replicating tasks. By dynamically adapting to changing attributes of the system and resources, it also ensures that the optimal numbers of replicas are used. The model ensures a minimized use of resources by not using more replicas than needed, and by minimizing the number of resources needed. This was achieved by placing replicas on the most reliable resources first and foremost.

In [15], the authors proposed the quantitative fault-tolerance with minimum execution cost QFEC and QFEC+ algorithms for a workflow. QFEC is implemented by iteratively choosing available replicas and VMs with the minimum execution time (Makespan) for each task until its sub-reliability requirement is satisfied. On another side, QFEC+ is implemented by filtering out partial QFEC opted replicas and VMs for each task with less redundancy (remove some replicas) while still satisfying its sub-reliability requirement.

3. THESIS GOAL

The following items are proposed as aims of the thesis:

1. Exploring and a comprehensive survey of the state-of-the-art algorithms of fault tolerance for Big Data scientific workflows in cloud computing environments.
2. Critical overview and evaluate former researches by comparison in the experiments.
3. Identification of the important objectives for scheduling scientific workflows on cloud computing according to the latest researches.
4. Creation of a model with two fault-tolerant approaches for scheduling scientific workflows on the cloud, the first approach uses a genetic algorithm and the second one uses the greedy algorithm.
5. Evaluating the model on different sizes and types of scientific workflows to validate the effectiveness of the proposed methods.
6. Deep analysis of the results, summarizing the results, benefits, and drawbacks of the proposed approaches, formalizing the recommendations for future development in the related researches.

The methods to fulfil the proposed aims of the thesis include:

Analysis:

- Identification of the state-of-the-art in improving the reliability of performance in distributed environments.
- Overview of the wide range of algorithms and techniques, with a focus on the most popular and used methods.

Implementation:

- Using WorkflowSim, which is an extension of the CloudSim framework that is completely written in Java, for modelling and simulation of cloud computing environments.
- The selected state-of-the-art algorithms will be coded in java alongside our proposed algorithms in CloudSim framework.
- Using scientific workflows that are taken from diverse domains such as astronomy, earthquake science, and biology, and are similar to real workflows.
- Simulation results will be handled and examined in the Excel spreadsheet.

Testing:

- Testing each individual component of the code to see if these components are working properly, then testing them as a collective group to see if there are potential errors and malfunctions.

Evaluation:

- All the simulation results will be collected in an appropriate manner for analysing them in the Excel spreadsheet.
- The proposed methods will be assessed by their influence on improving multi objectives of scheduling scientific workflows on cloud.
- Based on the analysis of the simulation results, the pros and cons of the proposed methods will be determined.
- The general recommendations and suggestions for good practice in fault-tolerant scheduling of scientific workflows will be formulated for use in relevant researches.

4. FAULT TOLERANCE MODEL

4.1 System model.

The system architecture is presented in (Figure 4.1). The workflow engine acts as a middle layer between the Big Data management systems and the cloud. Workflow submitted into the engine which schedules the workflow tasks, provide fault tolerance mechanism, and allocate resources in a manner for achieving a trade-off between reliability and cost or fulfilling required reliability with minimum cost.

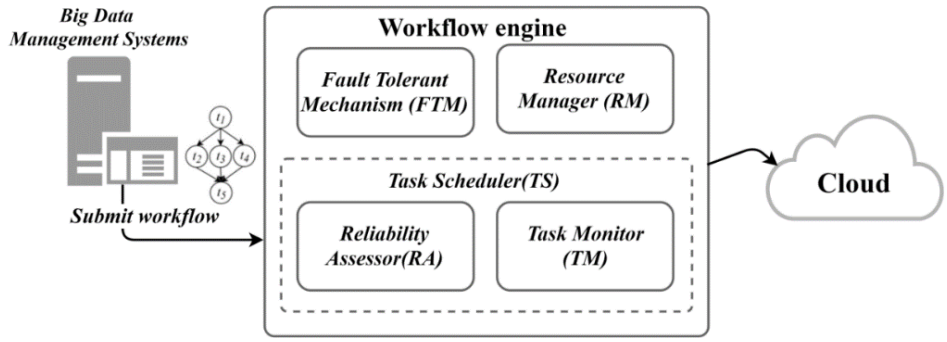


Fig. 4.1: System architecture.

Fault-Tolerant Mechanism (FTM): the cloud is prone to failures and an efficient fault-tolerant strategy is critical for increasing the reliability of unreliable execution environment. In this Thesis, we use replication mechanism as a fault tolerant strategy and offer two approaches for that, first one depends on genetic algorithm and the second one depends on greedy algorithm for optimizing the reliability and execution cost.

The Resource Manager (RM): acts as a broker for the available computing resources in the cloud. This module allocates the appropriate resource (virtual machine) for every task as chosen by the task scheduler.

The Task Scheduler (TS): can schedule workflow tasks to different resources. According to the scheduling information, the task scheduler employs a scheduling algorithm to find a suitable resource for every task.

The Task Monitor (TM): detect if a task t_i on virtual machine v_k successfully completes or fails before completion, and it sends a report about virtual machine v_k to the reliability assessor RA.

Reliability Assessor (RA): can maintain and recalculate the failure rate λ for each resource, which can be used to schedule the next workflow.

4.2 Workflow model

A scientific workflow with dependent tasks is modelled by a directed acyclic graph (DAG). Let consider that V represents a set of heterogeneous virtual machines

(VMs) on cloud, each individual virtual machine is denoted by V_k ,$V= \{V_1,V_2,\ldots, V_k\}$[9][10][11][13][15][16][17][18][19]. We also presume that communication can be overlapped with computation, which means data can be transmitted from one virtual machine to another while a task is being executed on the recipient virtual machine [9][10][15].

In this thesis a DAG is defined as $G= (T, W, D_{in}, D_{out} , E, C)$.

- T is the set of scientific workflow tasks, each individual task is denoted by t_i, $T= \{t_1,t_2,\ldots,t_i\}$. Each node $t_i \in T$ is a task with different execution times on different VMs. $pred(t_i)$ is the set of immediate predecessor tasks of t_i, while $succ(t_i)$ is the set of immediate successor tasks of t_i. Tasks without predecessor tasks are denoted by t_{entry}, and tasks with no successor tasks are denoted by t_{exit} [15][16][17][19].
- $W = |T| \times |V|$ represents matrix, where $w_{i,k} \in W$ denotes the execution time of task t_i running on VM v_k.
- D_{in} is the set of input datasets for all workflow. Each task t_i has input datasets is denoted by $d\_in_{i,j}= \{d\_in_{i,1}, d\_in_{i,2},\ldots, d\_in_{i,j}\}$.
- D_{out} is the set of output datasets for all workflow. Each task t_i has output datasets is denoted by $d\_out_{i,j}= \{d\_out_{i,1}, d\_out_{i,2},\ldots, d\_out_{i,j}\}$.
- E represents the set of directed edges among tasks in the workflow. Each individual an edge $e_{p,c} \in E$ means that a part or all of the output data of task t_p is the input data of tasks t_c.
- C represents the set of the communication time of data between tasks. $c_{p,c} \in C$ represents the communication time of data between task t_p and t_c.

4.3 Reliability model

Faults can be classified into three major types (according to duration) as transient, intermittent and permanent [20][21]. In this Thesis considers the transient failure of VMs.The mean time between failures (*MTBF*) for a VM is the average time between successive failures for that VM [22].

The *MTBF* can be calculated as:

$$MTBF = \frac{Total\ time}{Number\ of\ failures} \quad \text{(E.1)}$$

The failure rate λ calculate:

$$\lambda = \frac{1}{MTBF} \quad \text{(E.2)}$$

The models using the Poisson distribution to model the probability of failure assume constant failure rates. And the reliability for VM can be calculated as [9][10][13][15][18][19][22][23][24]

$$R(t) = e^{-\lambda t} \tag{E.3}$$

The probability of fail during a time interval of length t for VM is

$$F(t) = 1 - e^{-\lambda t} \tag{E.4}$$

The reliability of task t_i executed on VM v_k in its execution time is denoted by

$$R(t_i,v_k) = e^{-\lambda_k w_{i,k}} \tag{E.5}$$

Where, λ_k is the failure rate of v_k and $w_{i,k}$ is the execution time of the task t_i on the VM v_k. And the failure probability for t_i is

$$F(t_i,v_k) = 1 - R(t_i,v_k) = 1 - e^{-\lambda_k w_{i,k}} \tag{E.6}$$

When we use replication, the reliability of a task t_i with m replicas placed on m different VMs is

$$R(t_i) = 1 - \prod_{k=1}^{m} F_k(t_i) \tag{E.7}$$

The reliability of the workflow with all tasks should be

$$R(G) = \prod_{t_i \in T} R(t_i) \tag{E.8}$$

In this Thesis, we assume communication networks between VM provide fault-tolerance for themselves.

4.4 Cost model

Cloud computing environment contains many resources (datacentres), which include a number of hosts, where each host has a number of VMs with various configurations (CPU, memory, bandwidth, and storage) [25]. The cost includes execution cost, bandwidth cost, memory cost and storage cost [25][26][27][28][29] [30][31][32]. The cost model in this Thesis is based on a pay-as-you-go pricing model. The users are charged according to the amount of time and data that they have used computing resources [25][33].

The execution cost which computes the cost for the execution of the task t_i on the VM v_k is

$$Cost_E(t_i)_{v_k} = \frac{L(t_i)}{S(v_k)} \times C_{e,v_k} \tag{E.9}$$

Where $L(t_i)$ is the length of task t_i and $S(v_k)$ speed of VM v_k in millions of instructions per second (MIPS) and C_{e,v_k} is the cost of using processing on VM v_k.

The bandwidth cost of task t_i on VM v_k is

$$Cost_B(t_i)_{v_k} = \frac{(\sum_n S(d_{in_{i,n}}) + \sum_m S(d_{out_{i,m}})) \times 8}{BW_{v_k}} \times C_{b,v_k} \quad \text{(E.10)}$$

Where $S(d_{in_{i,n}})$ and $S(d_{out_{i,m}})$ are size of input and output dataset respectively (MB) of task t_i, BW_{v_k} is bandwidth of VM v_k in Mbps and C_{b,v_k} *is* the cost of using bandwidth on VM v_k.

The memory (RAM) cost of task t_i on VM v_k is

$$Cost_R(t_i)_{v_k} = (\sum_n S(d_{in_{i,n}}) + \sum_m S(d_{out_{i,m}})) \times C_{r,v_k} \quad \text{(E.11)}$$

Where $S(d_{in_{i,n}})$ and $S(d_{out_{i,m}})$ are sizes of input and output dataset respectively of task t_i and C_{r,v_k} *is* the cost of using memory on VM v_k.

- **The storage cost** of task t_i on VM v_k is

$$Cost_S(t_i)_{v_k} = (\sum_n S(d_{in_{i,n}}) + \sum_m S(d_{out_{i,m}})) \times C_{s,v_k} \quad \text{(E.12)}$$

Where $S(d_{in_{i,n}})$ and $S(d_{out_{i,m}})$ are sizes of input and output dataset respectively of task t_i and C_{s,v_k} *is* the cost of using storage in VM v_k.

The total cost of processing for mapped task t_i on VM v_k is

$$Cost(t_i)_{v_k} = Cost_E(t_i)_{v_k} + Cost_B(t_i)_{v_k} + Cost_R(t_i)_{v_k} + Cost_S(t_i)_{v_k} \quad \text{(E.13)}$$

The cost of all tasks on workflow is

$$Cost(G) = \sum_{t_i \in T, v_k \in V} Cost(t_i)_{v_k} \quad \text{(E.14)}$$

4.5 Task scheduling

Task scheduling for a DAG-based workflow includes two phases:

- ***Task prioritization:*** this phase orders tasks based priorities.
- ***Task allocation:*** this phase allocates each task to the appropriate VM, (see Figure 4.2).

Both task scheduling phases for a DAG-based workflow are NP-hard problem [9][10][11][13][15][17][24][25][34].

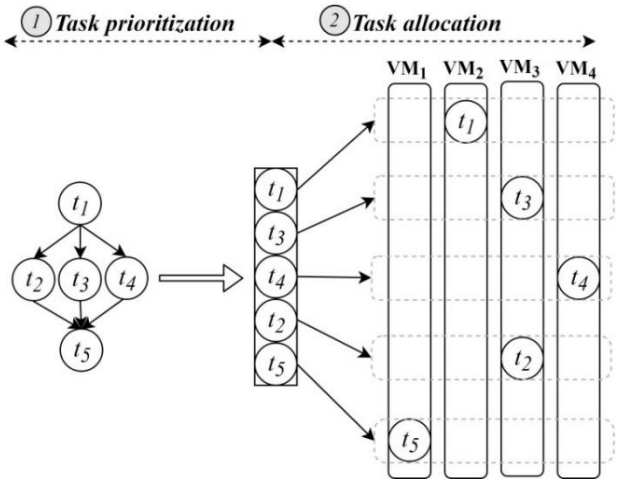


Fig. 4.2: Task prioritization and Task allocation for workflow

4.5.1 Task prioritization

Heterogeneous earliest finish time (HEFT) is one of the most famous scheduling algorithms for its low complexity, it is a classical static list scheduling algorithm [1][15][18][36][37]. HEFT uses the mean value of the computation cost and the mean value of communication cost as the rank value to determine the scheduling sequence [16], and it maintains a list of tasks sorted in decreasing order of their upward rank [17]. The tasks on the workflow are ordered by descending order of $rank_u$ [9][10][15][17][37][38][39], which is obtained by next equation .

$$rank_u(t_i) = \overline{w}_i + max_{t_j \in suss(t_i)}\{c_{i,j} + rank_u(t_j)\} \quad \text{(E.15)}$$

Where $\overline{w}_i$ represents the average execution times of task t_i, it is calculated by

$$\overline{w}_i = \frac{\sum_{k=1}^{|V|} w_{i,k}}{|V|} \quad \text{(E.16)}$$

Where $w_{i,k}$is the execution time of the task t_i on the virtual machine v_k ,each task has variable computation time on a different virtual machine, and $c_{i,j}$ is communication time of a data between two tasks t_i ,t_j ,it is calculated by

$$C_{i,j} = \frac{(\sum_n S(d_{out_{i,n}}) + \sum_m S(d_{in_{j,m}})) \times 8}{AVR(BW)} \quad \text{(E.17)}$$

Where $\sum S(d_{out_{i,n}})$ is the size of output datasets of task t_i and $\sum S(d_{in_{j,m}})$ is the size of input datasets of task t_j in MB.

$AVR(BW)$ is the average bandwidth of all virtual machines in Mbps, it is calculated by

$$AVR(BW) = \frac{\sum_{k=1}^{|V|} BW_{v_k}}{|V|} \quad \text{(E.18)}$$

4.5.2 Task allocation

Scheduling tasks find the solving which virtual machine resource that will be allocated to which task, for increasing the reliability and decreasing the execution cost [15][25][35][40][41][42][43]. We assume that we have a scientific workflow *G* and a set of heterogeneous virtual machines VM. The problem is to assign replicas and corresponding VMs for each task t_i; at the same time, we must ensure to minimize the execution cost of the workflow and ensure also that the obtained reliability value $R(G)$ satisfies required reliability $R_{req}(G)$.

The replica set of t_i is $\{t_i^1, t_i^2, \dots, t_i^{n_i}\}$, where t_i^1 is primary and the remainder is the backups. The total number of replicas for the workflow is

$$NRep(G)=\sum_{i=1}^{|T|} n_i \tag{E.19}$$

Let's suppose that, we want to execute workflow *G* at reliability level is (0.995), $R_{req}(G)=0.995$. The problem is to find the minimum execution cost of the workflow when assigning replicas and corresponding VMs for each task in a workflow.

$$Cost(G) = Minimum(\sum_{t_i \in T, v_k \in VM} Cost(t_i)_{v_k})$$
$$AND$$
$$R(G)= R_{req}(G)=0.995$$

5. RELIABILITY DRIVEN WORKFLOW SCHEDULING USING GENETIC ALGORITHM.

In this thesis, we propose an approach to optimise reliability and cost for Big Data scientific workflows in cloud computing environments using a genetic algorithm. We offer an idea to form chromosome structure, approach to encoding solution and build of genetic operators. The genetic algorithms can give several satisfying solutions by iterative evolutions over the first random generation of workflow scheduling.

To ensure fault-tolerant scheduling and improve the reliability and cost, each task is assigned on *m* distinct virtual machine (VM) resources. In this Thesis case *m=2*, a chromosome is a data structure in which a scheduling solution is encoded. As illustrated in (Figure 5.1), we use a two-dimensional string to represent a scheduling solution. One dimension of the string represents the first allocation of task t_i on virtual machine v_k, while the other dimension denotes the second allocation of task t_i on virtual machine v_j, where $v_k \neq v_j$.

The fitness function is used to measure each scheduled chromosome (solution). One of the most often used assessment methods is the weighted sum (as fitness function), which aggregates the objective values to a single quality measure. As the objective functions frequently have different scales, they are usually normalized [44]. This can be done for example by using equations (E.20) or (E.21) when minimizing and maximizing the objectives respectively:

$$f^{norm} = \frac{f - \min(f)}{\max(f) - \min(f)} \text{ for objectives to be minimized} \quad \text{(E.20)}$$

$$f^{norm} = 1 - \frac{f - \min(f)}{\max(f) - \min(f)} \text{ for objectives to be maximized} \quad \text{(E.21)}$$

In our Thesis, the fitness function is defined as:

$$f(s) = W_c \times \left(\frac{Cost(G) - \text{Min}\_Cost(G)}{\text{Max}\_Cost(G) - \text{Min}\_Cost(G)}\right) + W_R \times \left(1 - \frac{R(G) - \text{Min}\_R(G)}{\text{Max}\_R(G) - \text{Min}\_R(G)}\right) \quad \text{(E.22)}$$

Both cost and reliability are assigned weights W_C *and* W_R respectively, according to the trade-off requirement of the user, where $W_C + W_R = 1$, to calculate reliability *R(G)* and cost *Cost(G)* of the workflow we use the equations (E.8) and (E.14) respectively.

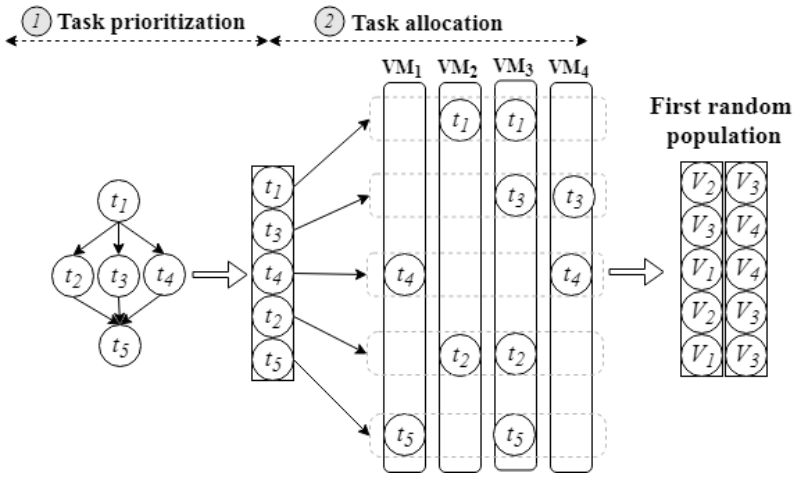


Fig. 5.1: Chromosome Structure

5.1 Experimental results of GA

The simulation is carried out by WorkflowSim, we create one datacenter, 6 hosts and 40 VMs; each host has several VMs based on its power. (Table 5.1) shows the characteristics of the resources used for the simulation.

Table 5.1 Characteristics of Resources

Virtual Machines			
MIPS of VM	**VM memory**	**Bandwidth**	**MTBF of VMs**
1000-3000 MIPS	512-1048MB	500-1000mbps	10^4 - 10^5 h
Virtual Machines Cost per unit			
Processing	**Memory**	**Bandwidth**	**Storage**
1.5-2.0	0.01-0.05	0.1-0.05	0.01-0.05

And we use three different sizes of the CyberShake workflow, Small (30 tasks), medium (100 tasks), and large (1000 tasks), (see Figure 5.2), and relative weights *WC and WR* are set as (Table 5.2).

Table 5.2 Relative Weights Values Used

W_C	0.99999	0.70000	0.50000	0.30000	0.00001
W_R	0.00001	0.30000	0.50000	0.70000	0.99999

(Figure 5.3, Figure 5.5 and Figure 5.7) present the reliability of the small, medium, and large CyberShake workflow respectively, and (Figure 5.4, Figure 5.5 and Figure 5.8) present the cost of the small, medium, and large CyberShake workflow respectively, according to previous relative weights and after 1000, 2500 and 50000 generations.

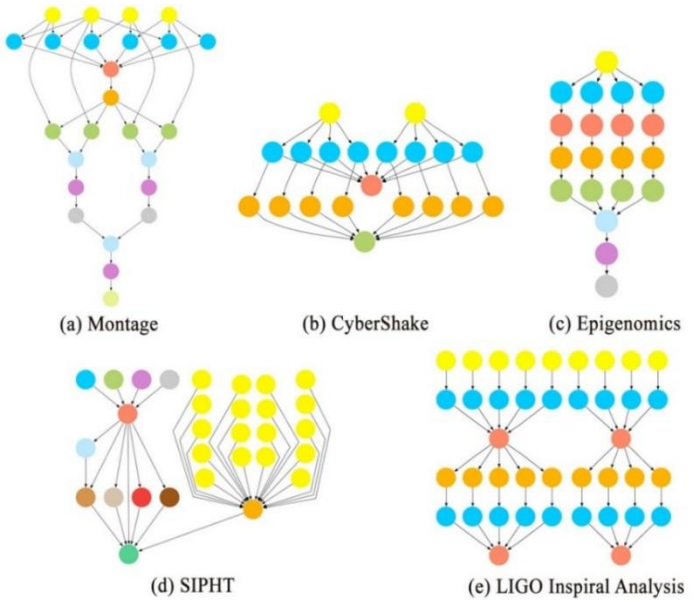


Fig. 5.2: The structures of some scientific workflows types.

Even though we achieved fast convergence of the solution to a close optimal level according to relative weights for small workflow (30 tasks), but for large workflow, we need to a larger number of generations to achieve optimization and stability in the output.

The number of generations set to 1000 for small workflows (30 tasks) and 25000 from the medium workflows (100 tasks) and 50000 for large workflows (1000 tasks). We kept the population size fixed for all workflow sizes under all choices for the number of generations and the number of virtual machines, in order to observe stability in the output. On another side, the number of virtual machines will have an important role to achieve optimization and stability in the output, for large workflows (1000 tasks), we achieve convergence of the solution to a close optimal level after 1000 generations when we use less number of virtual machines for allocating tasks, (see Figure 5.9 and Figure 5.10)

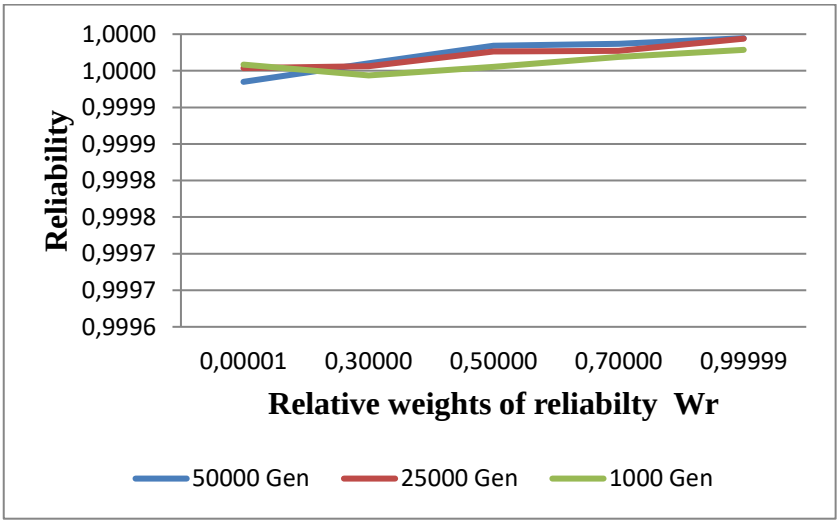


Fig. 5.3: The reliability of small CyberShake workflow (30 tasks)

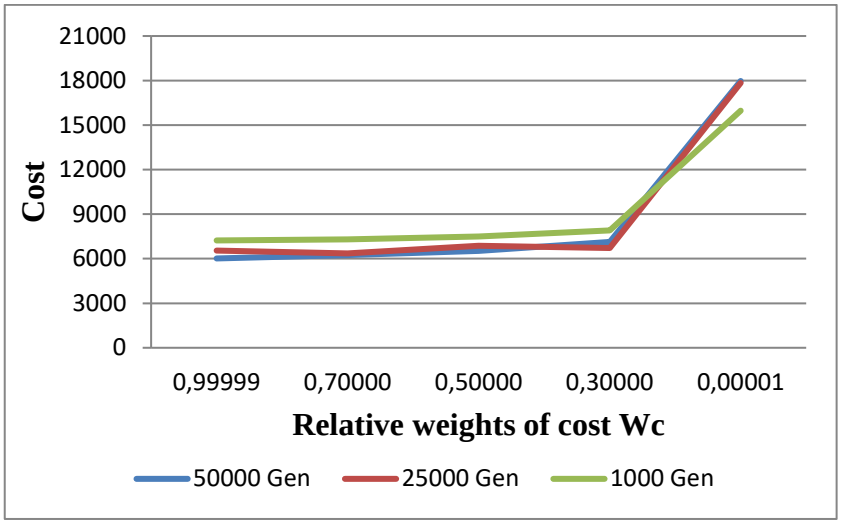


Fig. 5.4: The cost of small CyberShake workflow (30 tasks)

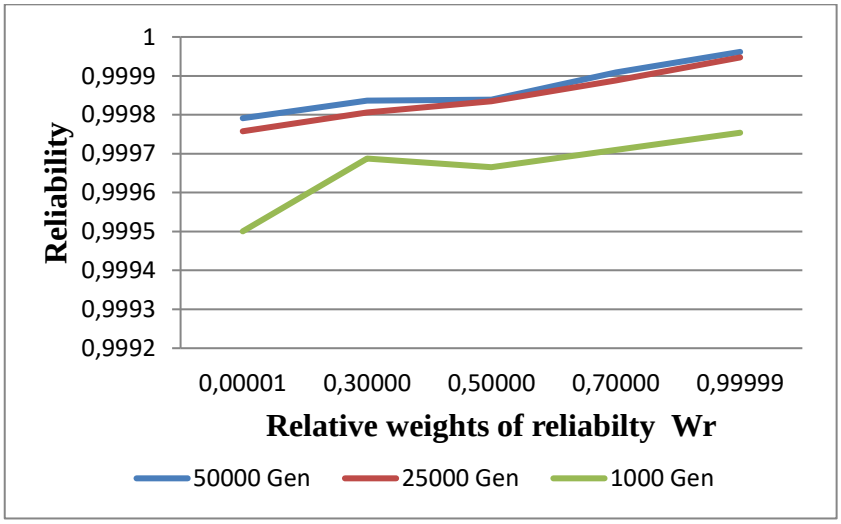


Fig. 5.5: The reliably of medium CyberShake workflow (100 tasks)

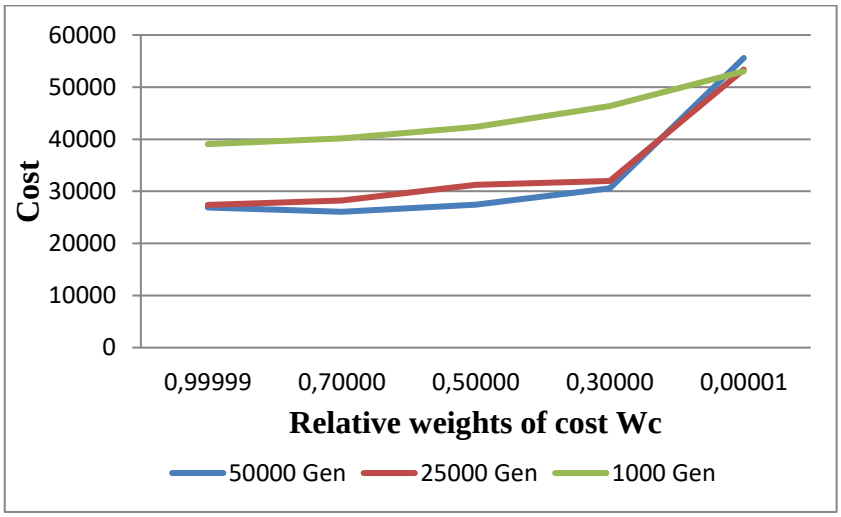


Fig. 5.6: The cost of medium CyberShake workflow (100 tasks)

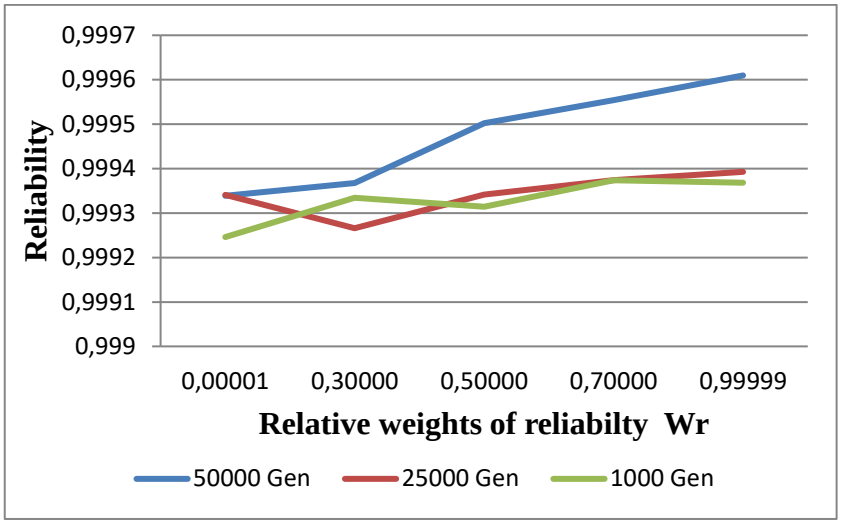


Fig. 5.7: The reliability of large CyberShake workflow (1000 tasks)

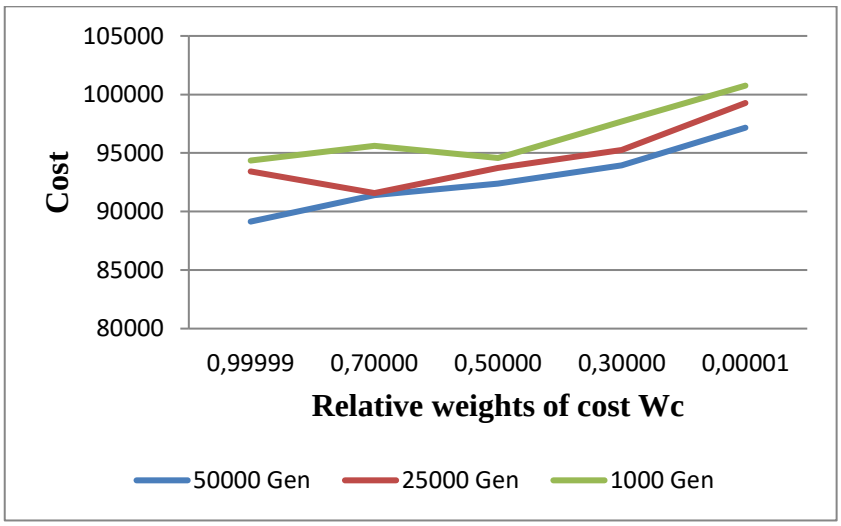


Fig. 5.8: The cost of large CyberShake workflow (1000 tasks)

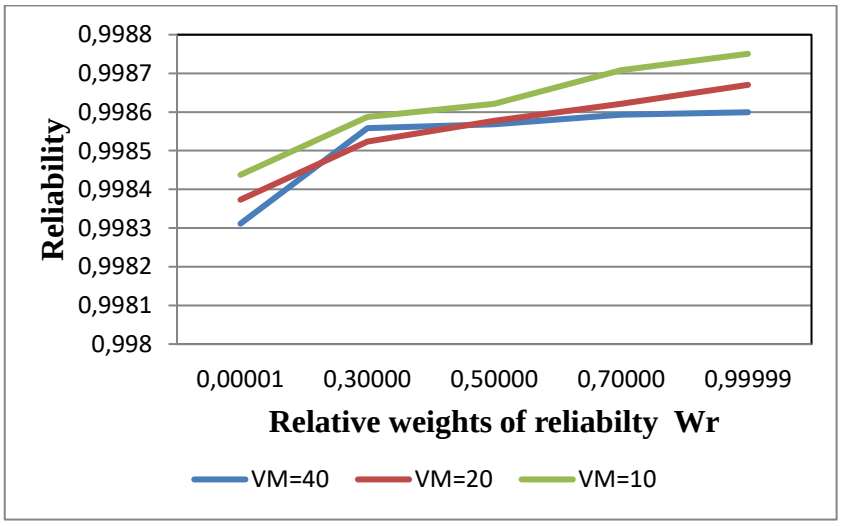


Fig. 5.9: The reliability of large CyberShake workflow after 1000 generations

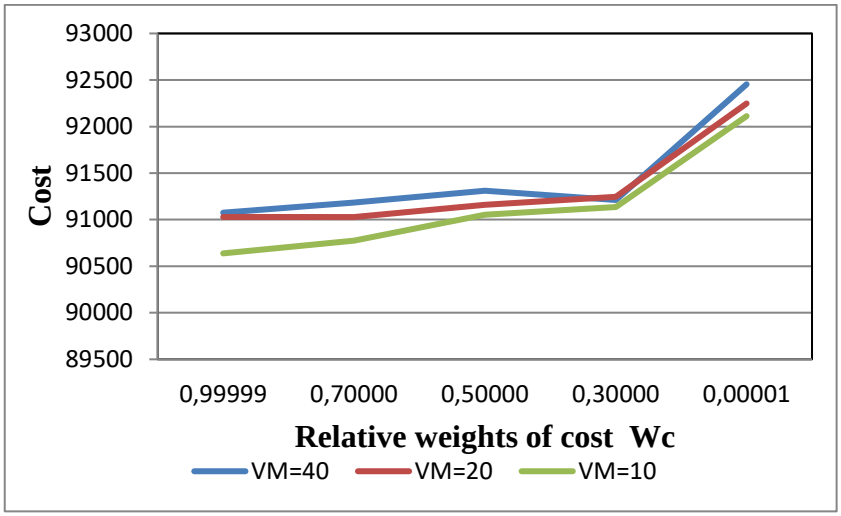


Fig. 5.10: The cost of large CyberShake workflow after 1000 generations

5.2 Multi-objective optimization using NSGA-II

In this thesis, we propose NSGA-II to optimize reliability and cost for scheduling scientific workflows in the cloud computing, to calculate reliability *R(G)* and cost *Cost(G)* of the workflow we use the equations (E.8) and (E.14) respectively.

Each individual or chromosome is represented as a vector of length equal to the number of tasks (1x100), the values specified in this vector are in the range (1, number of virtual machines (40)), the value corresponding to each position in the vector represents the VM to which task T is allocated.

5.3 Experimental results of NSGA-II

We create one datacenter, 6 hosts and 40 VMs; each host has several VMs based on its power. (Table 5.1) shows the characteristics of the resources used for the simulation, and we use three types of workflows, namely, CyberShake, Montage, and LIGO Inspiral workflows to validate the effectiveness of the proposed algorithm.

As it is noticeable in (Figure 5.11), (Figure 5.12) and (Figure 5.13) NSGA-II is capable to yield better optimal solutions to maximize reliability and minimize the cost for scheduling different types of workflows, where it gives consistent performance and has a good spread Pareto optimal set of solutions.

Sometimes, as well, MOO could give close results to single-objective optimization (reliability or cost) when achieves a trade-off between them. (Figure 5.14), (Figure 5.15) and (Figure 5.16) show a comparison of single-objective optimization of reliability, cost, and optimal solutions from MOO to schedule three different workflows; we can notice that some trade-off solutions are close to values of single-objective optimization. So, we can consider, the Pareto front of (reliability, cost) is a good option to make a decision regarding the optimized solution of scheduling big data scientific workflows on cloud computing.

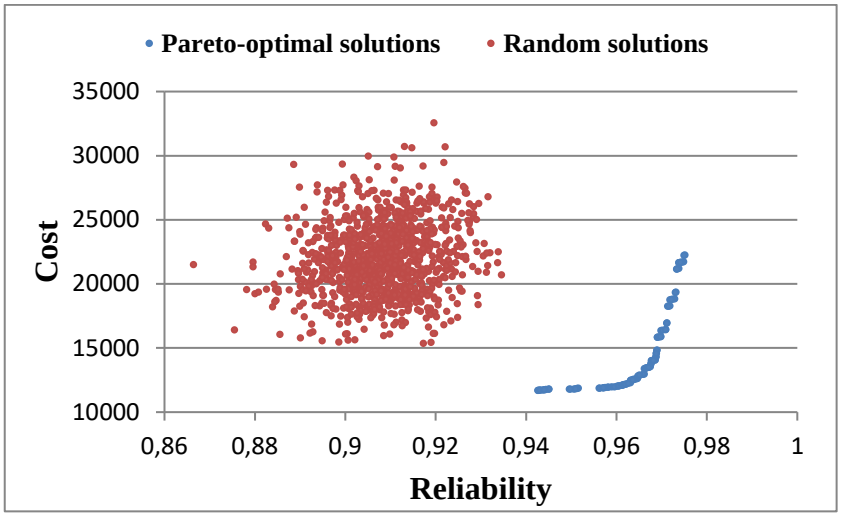


Fig. 5.11: Pareto-optimal solutions for scheduling CyberShake workflow

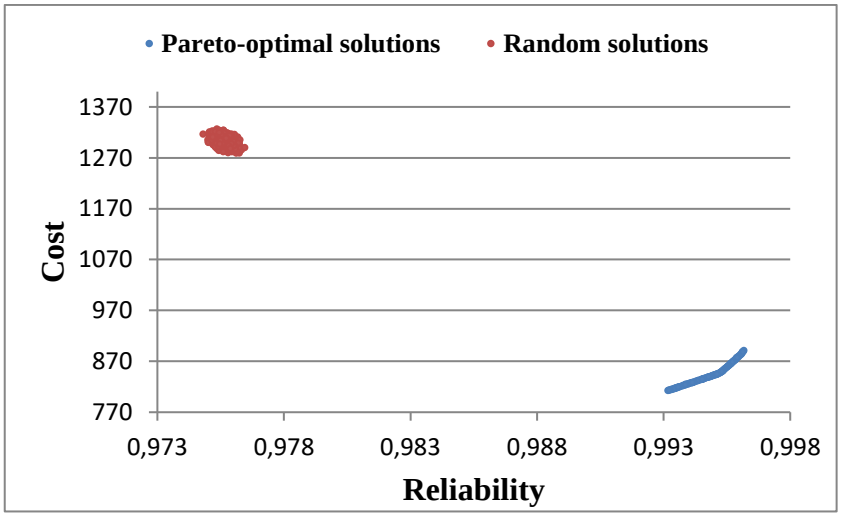


Fig. 5.12: Pareto-optimal solutions for scheduling Montage workflow

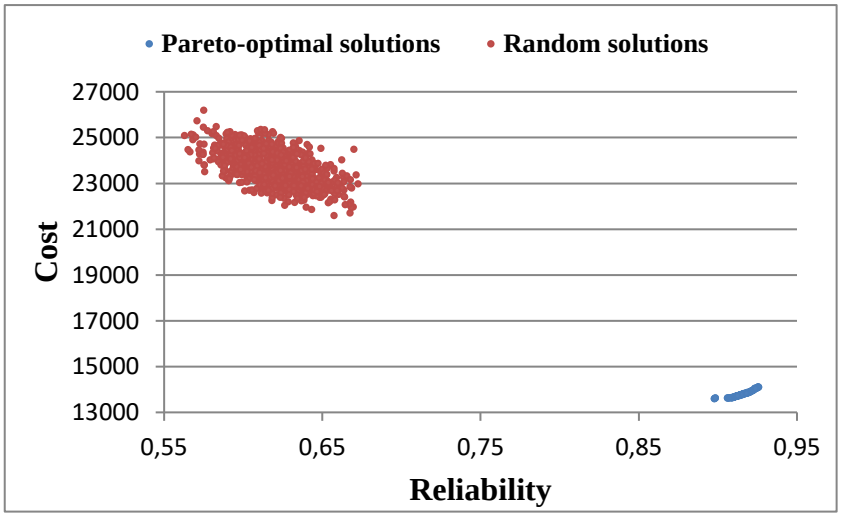


Fig. 5.13: Pareto-optimal solutions for scheduling Inspiral workflow

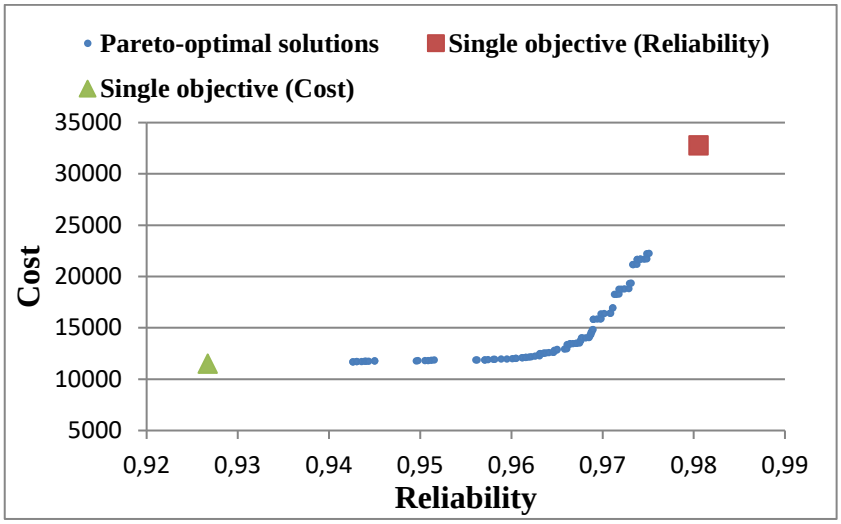


Fig. 5.14: Single objective against Multi-objective solutions for CyberShake workflow

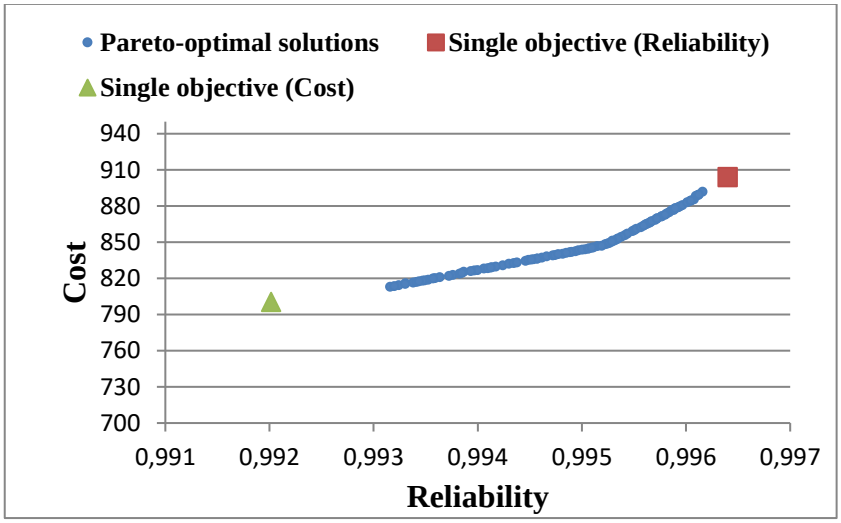


Fig. 5.15: Single objective against Multi-objective solutions for Montage workflow

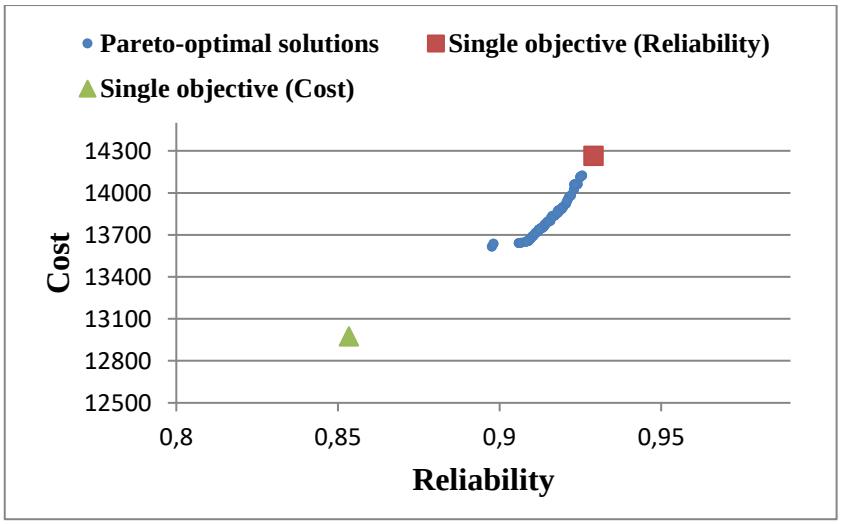


Fig. 5.16: Single objective against Multi-objective solutions for Inspiral workflow

6. DYNAMIC FAULT TOLERANCE USING GREEDY ALGORITHM

In this Thesis, we also propose the greedy scheduling algorithm that moves the reliability requirement of the workflow to the sub-reliability requirement of each task and finding replicas that satisfy sub-reliability with minimum execution cost. The use a static approach to determine how many copies are required to reach a certain level of reliability is impractical and insufficient.

6.1 Satisfying required reliability

Many algorithms were presented to transfer the reliability requirement of the workflow to the sub-reliability requirement of each task [9][10][14][15].

Older algorithms as MaxRe [9], the sub-reliability requirement of each task is calculated by

$$R_{req}(t_i) = \sqrt[|T|]{R_{req}(G)} \tag{E.23}$$

Later algorithms as RR [10] and QFEC+ [15], where the sub-reliability requirement of the entry task t_{entry} (t_1) is calculated by (E.23)

$$R_{req}(t_1) = \sqrt[|T|]{R_{req}(G)} \tag{E.24}$$

In RR algorithm, the sub-reliability requirements of the remainder of tasks (non-entry tasks) are calculated continuously based on the actual reliability achieved by previous allocations:

$$R_{req}(t_j) = \sqrt[|T|-j]{\frac{R_{req}(G)}{\prod_{x=0}^{j-1} R(t_x)}} \tag{E.25}$$

And QFEC+ algorithm, the sub-reliability requirements of the remainder of tasks (non-entry tasks) are calculated also based on the actual reliability achieved by previous allocations:

$$R_{req}(t_j) = \frac{R_{req}(G)}{\prod_{x=1}^{j-1} R(t_x) \times \prod_{y=j+1}^{|T|} R_{upper\_req}(t_y)} \tag{E.26}$$

Where $R_{upper\text{-}req}(t_i)$ is the upper bound on the reliability requirement of the task t_i, that is calculated by

$$R_{upper\_req}(t_i) = \sqrt[|T|]{R_{req}(G)} \tag{E.27}$$

MaxRe and RR choose replicas and VMs with the maximum reliability for each task to minimize the number of replicas until the sub-reliability of the task is

satisfied, thereby to reduce execution cost. On another side, QFEC+ algorithm selects replicas and VMs with the minimum execution time value for each task, thereby to reduce execution cost until the sub-reliability of the task is satisfied, then it removes some replicas that have minimum reliability while still satisfying its sub-reliability requirement. However, the minimum number of replicas does not mean minimum execution cost because of the heterogeneity of VMs.

In this Thesis, we use equation (E.26) to calculate sub-reliability for each task t_i, and we propose a novel approach which selects replicas and VMs with the minimum execution cost value and satisfies the sub-reliability for each task, therefore satisfies the reliability requirements of the workflow.

6.2 The DFTGA algorithm

In our algorithm DFTGA the reliability requirement of the workflow is transferred to the sub-reliability requirement of each task. Then, DFTGA simply locates replicas with the minimum execution cost on VMs for each task and sub-reliability requirement should be satisfied.

The main steps are as follows:

1. In lines 1-5, finding a possible maximum number of replicas *M*.
2. In line 6, If *M* larger than an allowed maximum number of replicas *MaxRep* then remove V_{min_R} that has minimum reliability from the cloud and go to 2.
3. In line 7, we order tasks descending according to $rank_u$, using the equation (E.15).
4. In lines 10-11, we calculate reliability and cost of t_i on each virtual machine, using the equations (E.5) and (E.13) respectively.
5. In line 14, we calculate the sub-reliability requirement of the entry task t_1 $R_{req}(t_1)$, using the equation (E.24).
6. In line 16, we calculate the sub-reliability requirement of non-entry tasks $R_{req}(t_i)\ where\ i \neq 1$, using the equation (E.26).
7. In line 18, for task t_i, create a set *M* of minimum execution cost of replicas on virtual machines.
8. In line 19, finding the sub-set *SM* of replicas from *M*, which achieve the minimum cost of task t_i and sub-reliability requirement is satisfied, and calculate $\mathrm{R}(\mathrm{t_i})$, $Cost(t_i)_{v_k}$ and n_i.
9. In line 21, DFTGA calculates the real reliability value *R(G)*, execution cost *cost(G)* and the number of replicas *NRep(G)* of the workflow, (see Figure 6.1).

Input: ***G= (T, W, D_{in}, D_{out} , E, C),V ,R_{req}(G)****, MaxRep.*

Output: ***Cost(G) , R(G) ,*** $NRep$***(G).***

1: *Finding a virtual machine* V_{min_R} *that has minimum reliability.*

2: *Finding the task that has maximum length* t_L.

3: *Calculate* $R_{req}(t_L)$ *, using the equation (E.24).*

4: *Calculate* $R(t_L, v_{min\_R})$ *, using the equation (E.5).*

5: *Finding M, the number of replicas of* t_L *that satisfy* $R_{req}(t_L)$ *on* V_{min_R}

6: ***If*** *M > MaxRep* ***then***
Call_Proc: *Remove* V_{min_R} *from V ,* ***Call_Proc:*** *Add new VM,*
Go to 2:

End if

7: *Sort(Tasks), descending order of* $rank_u$*, using the equation (E.15).*

8: ***For****(i=1; i≤|T| ; i++)*

9: ***For****(k=1; k≤|V| ; K++)*

10: *Calculate* $R(t_i, v_k)$ *,using the equation (E.5).*

11: *Calculate* $Cost(t_i)_{v_k}$ *,using the equation (E.13).*

12: ***End for***

13: ***If*** *(i==1)* ***then***

14: *Calculate* $R_{req}(t_1)$*, using the equation (E.24).*

15: ***Else***

16: *Calculate* $R_{req}(t_i)$*, using the equation (E.26).*

17: ***End if***

18: *Create a set |M| of minimum execution cost of* t_i

19: *finding the sub-set SM of replicas from M, where*
$Minimum(\sum_{t_i \in T, v_k \in SM} Cost(t_i)_{v_k})$ *and* $R(t_i) >= R_{req}(t_i)$
Calculate $R(t_i)$*, using the equation (E.7).*
Calculate $Cost(t_i)_{v_k}$ *, using the equation (E.13).*
n_i*=|SM|*

20: ***End for***

21: *Calculate* $R(G)$*, using the equation (E.8).*
Calculate $Cost(G)$*, using the equation (E.14).*
Calculate $NRep(G)$*, using the equation (E.19).*

Fig. 6.1: The DFTGA algorithm

6.3 Experimental results and performance evaluation of DFTGA

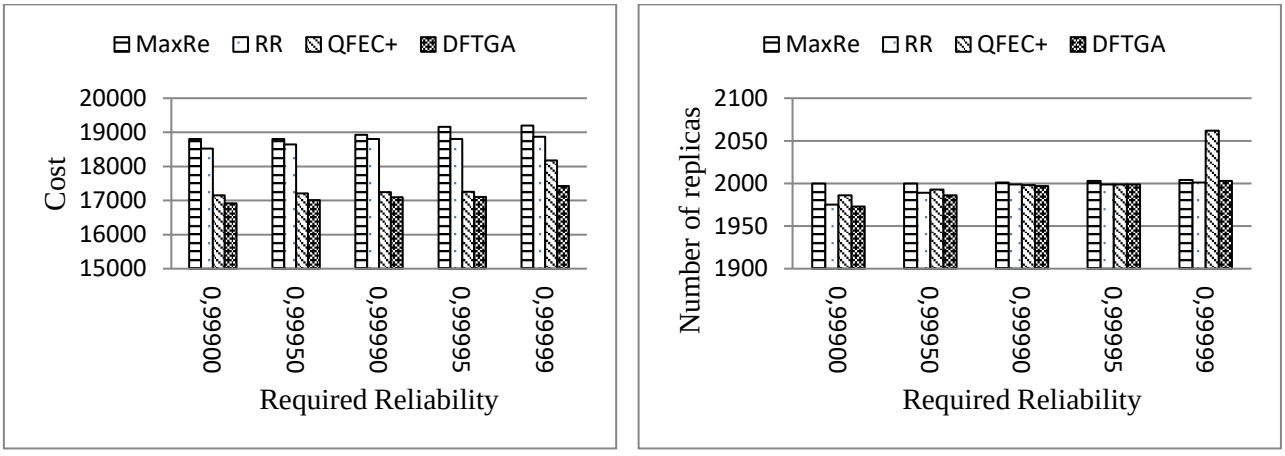
We use WorkflowSim to measure the performance of the proposed algorithm and compare with MaxRe, RR and QFEC+. WorkflowSim can use scientific workflows generated by Pegasus workflow management system [45]. The workflow characteristics are taken from diverse domains such as astronomy, earthquake science, and biology resemble real workflows [46].

We create one datacenter, 6 hosts and 40 VMs; each host has several VMs based on its power. Table 5.1 shows the characteristics of the resource. And we use five types of workflows, namely, *Montage, Sipht, LIGO Inspiral CyberShake, and Epigenomics workflows* (see Figure 6.2), and use several level of required reliability $R_{req}(G)$ (0.99900 to 0.99999) to validate the effectiveness of the proposed algorithm , and *MaxRep*=8.

(Figure 6.2A), (Figure 6.3A), (Figure 6.4A), (Figure 6.5A) and (Figure 6.6A) show the results of execution costs of five workflows types for varying reliability requirements, the execution costs increase with the increase in reliability requirements. In all cases, DFTGA produces minimum execution costs followed by QFEC+, RR, MaxRe, as we see, the results indicate that DFTGA is more effective in reducing execution cost than all previous algorithm. (Figure 6.2B), (Figure 6.3B), (Figure 6.4B), (Figure 6.5B) and (Figure 6.6B) show the results of the number of replicas of five workflows types for varying reliability requirements, the number of replicas increases with the increase in reliability requirements. The following observations are taken:

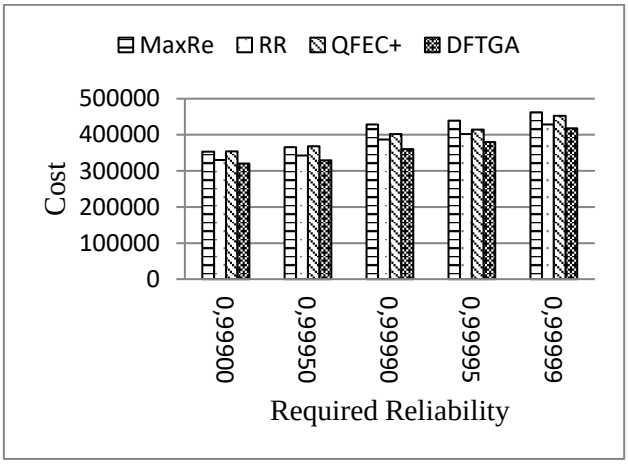
- In *Sipht* workflow, DFTGA produces a minimum number of replicas.
- In *Montage* workflow, DFTGA produces a minimum number of replicas in case low and medium reliability requirements.

In *LIGO Inspiral*, *CyberShake* and *Epigenomics* workflows, RR produces a minimum number of replicas.

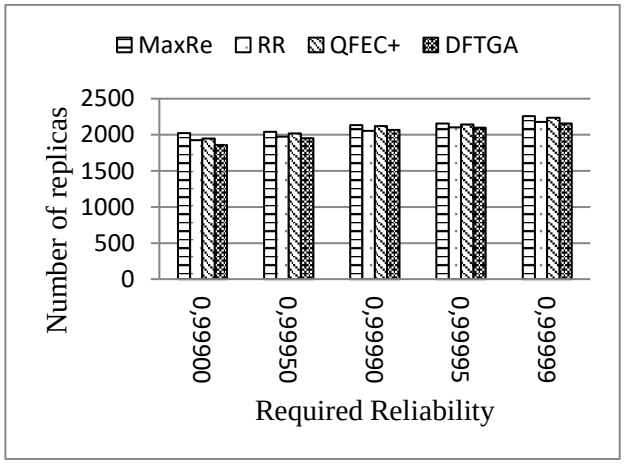


(A) Cost *(B) Number of Replicas*

Fig. 6.2: Cost and number of replicas of Montage workflow

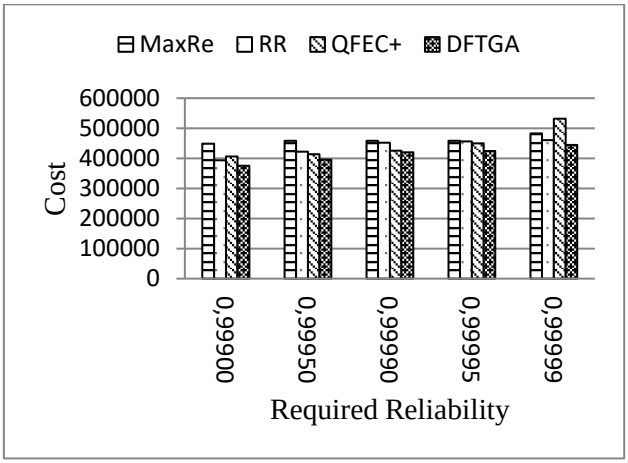


(A) Cost

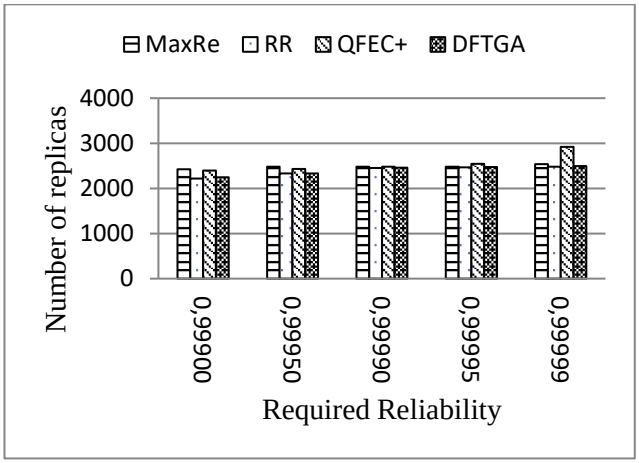


(B) Number of Replicas

Fig. 6.3: Cost and number of replicas of Sipht workflow

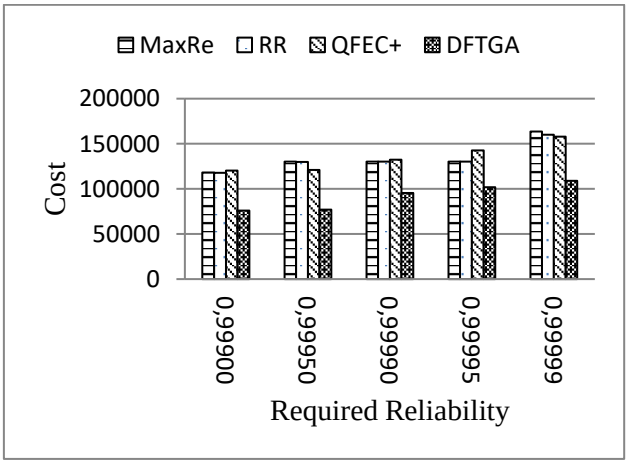


(A) Cost

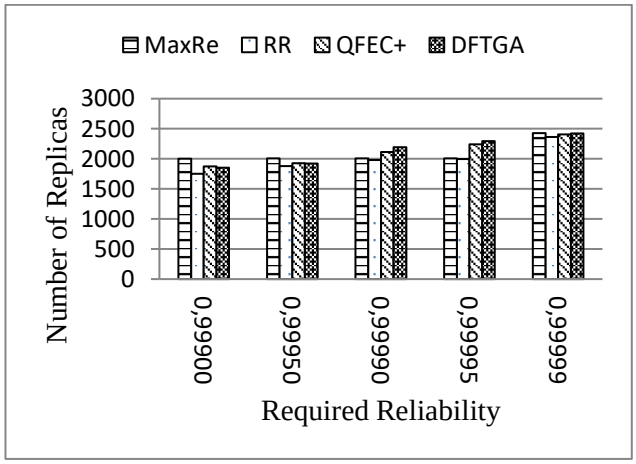


(B) Number of Replicas

Fig. 6.4: Cost and number of replicas of LIGO Inspiral workflow

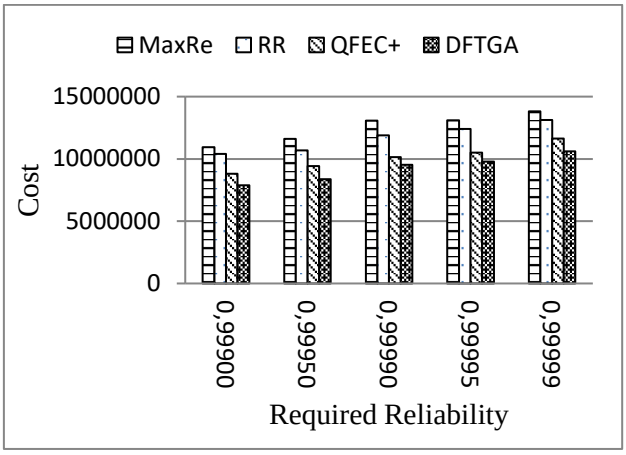


(A) Cost

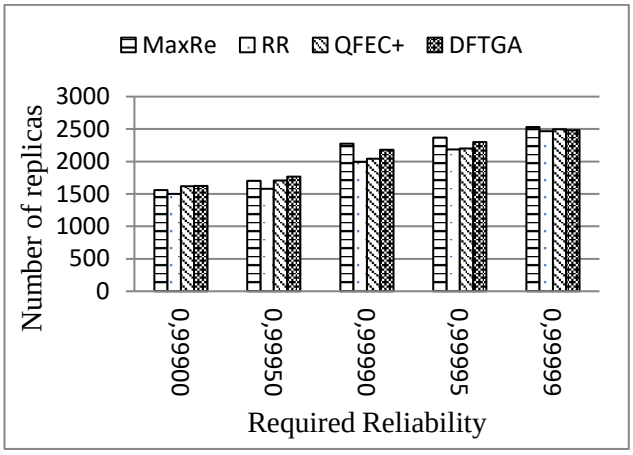


(B) Number of Replicas

Fig. 6.5: Cost and number of replicas of CyberShake workflow



(A) Cost



(B) Number of Replicas

Fig. 6.6: Cost and number of replicas of Epigenomics workflow

7. THESIS OUTCOMES

The challenges brought by the fault tolerance paradigm for execution Big Data scientific workflows reveal varied limitations such as the cost of execution. Providing fault-tolerant technologies for scientific workflow, a trade-off between the reliability and the cost and ensuring a predefined level of reliability with minimum cost are some of the issues identified in this work. The faults and the cost of fault tolerance prevent to fully benefit from the advantages brought by cloud computing such as elasticity and scalability.

In this Thesis, we addressed these issues by proposing two approaches that enhance the reliability to execute Big Data scientific workflow and reduce the cost within cloud computing. We demonstrated the advantages of our contributions by applying and compare them to state-of-the-art solutions, and consequently improving their performance, for tackling Big Data scientific workflows.

Model of fault tolerance for Big Data scientific workflows

The main objective of our work was to propose a fault tolerance paradigm and fulfill the reliability of Big Data scientific workflows on cloud computing. Because of the complexity of cloud computing, executing Big Data scientific workflows reliably is a challenge, we have developed a fault tolerance model that uses a replication mechanism for reliable execution of workflows on the cloud. It takes into consideration the cost of execution to determine the best fault tolerance strategy.

A trade-off between the reliability and the cost

Reaching a level of reliability to execute workflows on cloud computing is the main objective of the cloud providers and users. However, the costs users are willing to pay to execute these workflows and the reliability levels they seek are specific to each scenario. Therefore, an important focus of this work was to provide fault tolerant scheduling approach which optimizes for trade-offs between the cost and reliability.

To this purpose, we modeled the relation between cost and reliability for customizable levels of reliability according to cost constraints. As an intelligent search optimization technique, we have used a genetic algorithm which is considered an important approach to NP-hard and complex nature optimization problems, the results showed that our solution is able to optimize and the trade-off between cost and reliability.

Ensuring a predefined level of reliability with minimum cost

Replication is an important fault-tolerant technique applied to satisfy the reliability requirement. A static approach of fault tolerance to determine how many copies are required to reach a certain level of reliability is not the best fit for the dynamic environment of the cloud, due to the scientific workflows usually are executed in highly heterogeneous distributed environments.

Therefore, we proposed dynamic fault-tolerant scheduling for scientific workflow in the cloud computing environment. The purpose of DFTGA algorithm is to ensure a predefined level of reliability with minimizing cost which is based on tasks replication method that is one of the widely used faults tolerant mechanisms. The simulation results with real-world scientific workflow models show that DFTGA algorithm can offer best results compare with former researches in the same field.

8. CONCLUSIONS

The main goal of this thesis is to devise methods for improving the reliability of executing Big Data scientific workflows in cloud computing environments and reduce the cost that produces by the fault-tolerance mechanism. To accomplish this; this thesis described a model with two fault-tolerant approaches that manage scientific workflows on cloud computing with an objective to minimize execution cost, the first one was based on a genetic algorithm and the second one was built on a greedy algorithm.

The thesis formally described and defined the scheduling problems in the context of Big Data scientific workflows on the cloud, and provided an overview of scientific workflows, motivated by real-world examples that we used to evaluate our model. Following the problem definition based on motivational examples, this thesis presented state-of-the-art techniques to schedule workflows in distributed systems and described several works, and identified their contributions and shortcomings.

The thesis provided a background of faults that are common in cloud computing, and discuss the fault tolerance techniques and tools used for implementing fault tolerance techniques in cloud computing to execute scientific workflows.

The objective was also to propose a platform to be used to further experiments and evaluations because researchers often cannot reach the real cloud environment, and this was accomplished by implementing and evaluating our experiments using the WorkflowSim platform.

We have increased the reliability by the replication of tasks and at the same time achieved a trade-off between the reliability and cost to execute workflow on cloud computing by genetic algorithm. As well, this study experimentally demonstrates the positive impact of NSGA-II to support decision-makers in solving multi-objective problems by providing a set of final solutions. As a Pareto-based method, it provides a set of solutions that show different trade-offs between multiple objectives.

In DFTGA, we transferred the reliability requirement of the workflow to sub-reliability requirement of each task. And the required sub-reliability is ensured by creating enough replicas with the minimum execution cost. Simulation results show the improvement in the cost of workflows comparing to the other algorithms. In previous works, selecting VM depends on the reliability and ignores VM that has low reliability, but in DFTGA we improved the reliability of the cloud by removing VM with minimum reliability and add new VM, but more effort and further research are needed to optimize numbers of replicas.

Finally, this thesis demonstrated the applicability of a fault-tolerant model on scientific workflow applications and made significant contributions toward the advancement of the field, and formalized the model of the cloud computing environment.

9. REFERENCES

[1] da Silva, R.F., Glatard, T. and Desprez, F., (2015). Self-Management of Operational Issues for Grid Computing: The Case of The Virtual Imaging Platform. *In Emerging Research in Cloud Distributed Computing Systems* ,pp. 187-221. IGI Global. ISBN13: 9781466682139, ISBN10: 1466682132, EISBN13: 9781466682146.

[2] Zhang, F., Cao, J., Hwang, K., Li, K. and Khan, S.U., (2015). Adaptive workflow scheduling on cloud computing platforms with iterativeordinal optimization. *IEEE Transactions on Cloud Computing*, *Vol.3 ,No.2*, pp.156-168.

[3] Haryanti, S.C. and Sari, R.F., (2014). Reliability of resource allocation in mobile ad hoc grid with tasks replication. *Journal of Computers*, *Vol.9 ,No.2*, pp.328-336.

[4] Smith, D.J., (2011).*Reliability, Maintainability and Risk: practical methods for engineers, Eighth Edition*. Butterworth-Heinemann. ISBN: 978-0-08-096902-2.

[5] Software engineering daily*: Cloud Cost Optimization* [Online]. © 2018 [viewed 2019-03-09]. Available from: https://softwareengineeringdaily.com/2018/11/07/cloud-cost-optimization/

[6] Mesbahi, M.R., Rahmani, A.M. and Hosseinzadeh, M., (2018). Reliability and high availability in cloud computing environments: a reference roadmap. *Human-centric Computing and Information Sciences*, *Vol.8,No.1*, p.20. https://doi.org/10.1186/s13673-018-0143-8.

[7] Ahmed, W. and Wu, Y.W., (2013). A survey on reliability in distributed systems. *Journal of Computer and System Sciences*, *Vol.79,No.8*, pp.1243-1255.

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

[9] Zhao, L., Ren, Y., Xiang, Y. and Sakurai, K., (2010), September. Fault-tolerant scheduling with dynamic number of replicas in heterogeneous systems. In *2010 IEEE 12th International Conference on High Performance Computing and Communications (HPCC)* ,pp. 434-441. IEEE. https://doi.org/10.1109/HPCC.2010.72.

[10] Zhao, L., Ren, Y. and Sakurai, K., (2013). Reliable workflow scheduling with less resource redundancy. *Parallel Computing*, *Vol.39,No.10* , pp.567-585.

[11] Wang, X., Yeo, C.S., Buyya, R. and Su, J., (2011). Optimizing the makespan and reliability for workflow applications with reputation and a look-ahead genetic algorithm. *Future Generation Computer Systems*, *Vol.27,No.8*, pp.1124-1134.

[12] Sun, D., Zhang, G., Wu, C., Li, K. and Zheng, W., (2017). Building a fault tolerant framework with deadline guarantee in big data stream computing environments. *Journal of Computer and System Sciences*, *Vol.89*, pp.4-23.

[13] Wang, S., Li, K., Mei, J., Xiao, G. and Li, K., (2017). A reliability-aware task scheduling algorithm based on replication on heterogeneous computing systems. *Journal of Grid Computing*, *Vol.15,No.1*, pp.23-39.
[14] Stahl, P., Broberg, J. and Landfeldt, B., (2017), April. Dynamic Fault-Tolerance and Mobility Provisioning for Services on Mobile Cloud Platforms. In *2017 5th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)*, pp. 131-138. IEEE.
[15] Xie, G., Zeng, G., Li, R. and Li, K., (2017). Quantitative fault-tolerance for reliable workflows on heterogeneous IaaS clouds. *IEEE Transactions on Cloud Computing*. https://doi.org/10.1109/TCC.2017.2780098.
[16] Ebrahimi, M., Mohan, A., Kashlev, A. and Lu, S., (2015), March. BDAP: a big data placement strategy for cloud-based scientific workflows. In *2015 IEEE First International Conference on Big Data Computing Service and Applications* pp. 105-114. IEEE.
[17] Xie, G., Li, R. and Li, K., (2015). Heterogeneity-driven end-to-end synchronized scheduling for precedence constrained tasks and messages on networked embedded systems. *Journal of Parallel and Distributed Computing, Vol.83*, pp.1-12.
[18] Mei, J., Li, K., Zhou, X. and Li, K.,(2015). Fault-tolerant dynamic rescheduling for heterogeneous computing systems. *Journal of Grid Computing, Vol. 13, Issue 4*, pp.507-525.
[19] Chen, H., Wen, J., Pedrycz, W. and Wu, G., (2018). Big data processing workflows oriented real-time scheduling algorithm using task-duplication in geo-distributed clouds. *IEEE Transactions on Big Data*.
[20] Lackovic, M., Talia, D., Tolosana-Calasanz, R., Banares, J.A. and Rana, O.F. (2010). A taxonomy for the analysis of scientific workflow faults. In *2010 13th IEEE International Conference on Computational Science and Engineering* ,pp. 398-403. IEEE.
[21] Sozer, H., Tekinerdogan, B. and Aksit, M., (2007). Extending failure modes and effects analysis approach for reliability analysis at the software architecture design level. In *Architecting dependable systems IV* ,pp. 409-433. Springer, Berlin, Heidelberg.
[22] Koren, I. and Krishna, C.M., (2010). *Fault-tolerant systems*. ISBN 13: 978-0-12-088568-8, ISBN 10: 0-12-088568-9, Elsevier.
[23] Hwang, S. and Kesselman, C., (2003), June. Grid workflow: a flexible failure handling framework for the grid. In *High Performance Distributed Computing, 2003. Proceedings. 12th IEEE International Symposium on* (pp. 126-137). IEEE.
[24] Benoit, A., Dufossé, F., Girault, A. and Robert, Y., (2013). Reliability and performance optimization of pipelined real-time systems. *Journal of Parallel and Distributed Computing*, *Vol.73,No.6*, pp.851-865.
[25] Alworafi, M.A., Dhari, A., Al-Hashmi, A.A. and Darem, A.B., (2017). Cost-aware task scheduling in cloud computing environment. *International Journal of Computer Network and Information Security*, *Vol. 9, No. 5*, p.52.

[26] Calheiros, R.N., Ranjan, R., Beloglazov, A., De Rose, C.A. and Buyya, R., (2011). CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and experience*, *Vol.41,No.1*, pp.23-50.

[27] Calheiros, R.N., Ranjan, R., De Rose, C.A. and Buyya, R., (2009). Cloudsim: A novel framework for modeling and simulation of cloud computing infrastructures and services. *arXiv preprint arXiv:0903.2525*.

[28] Buyya, R., Ranjan, R. and Calheiros, R.N., (2009), June. Modeling and simulation of scalable Cloud computing environments and the CloudSim toolkit: Challenges and opportunities. In *2009 international conference on high performance computing & simulation* ,pp. 1-11. IEEE.

[29] Ahmed-Nacer, M., Gaaloul, W. and Tata, S., (2017), June. Occi-compliant cloud configuration simulation. In *2017 IEEE International Conference on Edge Computing (EDGE)* ,pp. 73-81. IEEE.

[30] Ahmed-Nacer, M., Suri, K., Sellami, M. and Gaaloul, W., (2017), June. Simulation of configurable resource allocation for cloud-based business processes. In *2017 IEEE International Conference on Services Computing (SCC)* ,pp. 305-313. IEEE.

[31] Chawla, Y. and Bhonsle, M., (2013). Dynamically optimized cost based task scheduling in Cloud Computing. *International Journal of Emerging trends & technology in computer science*, *Vol.2,No.3*, pp.38-42.

[32] Sheeja, Y.S. and Jayalekshmi, S., (2014), December. Cost effective load balancing based on honey bee behaviour in cloud environment. In *2014 First International Conference on Computational Systems and Communications (ICCSC)* ,pp. 214-219. IEEE.

[33] Abrishami, S., Naghibzadeh, M. and Epema, D.H., 2013. Deadline-constrained workflow scheduling algorithms for infrastructure as a service clouds. *Future Generation Computer Systems*, *Vol. 29, No. 1*, pp.158-169.

[34] Mohan, A., Ebrahimi, M., Lu, S. and Kotov, A., (2016), December. Scheduling big data workflows in the cloud under budget constraints. In *2016 IEEE International Conference on Big Data (Big Data)* ,pp. 2775-2784. IEEE.

[35] Dubey, K., Kumar, M. and Sharma, S.C., (2018). Modified HEFT algorithm for task scheduling in cloud environment. *Procedia Computer Science*, *Vol.125*, pp.725-732.

[36] Liu, J., Pacitti, E., Valduriez, P. and Mattoso, M., (2015). A survey of data-intensive scientific workflow management. *Journal of Grid Computing*, *Vol.13, No.4*, pp.457-493.

[37] Ghafarian, T., Javadi, B. and Buyya, R., (2015). Decentralised workflow scheduling in volunteer computing systems. *International Journal of Parallel, Emergent and Distributed Systems*, *Vol.30,No.5*, pp.343-365.

[38] Anwar, N. and Deng, H., (2018). Elastic scheduling of scientific workflows under deadline constraints in cloud computing environments. *Future Internet*, *Vol.10,No.1*, p.5.
[39] Wang, G., Wang, Y., Liu, H., & Guo, H. (2016). HSIP: A Novel Task Scheduling Algorithm for Heterogeneous Computing. *Scientific Programming, 2016*, pages11.
[40] Chronaki, K., Rico, A., Badia, R.M., Ayguadé, E., Labarta, J. and Valero, M., (2015), June. Criticality-aware dynamic task scheduling for heterogeneous architectures. In *Proceedings of the 29th ACM on International Conference on Supercomputing*, pp. 329-338. ACM.
[41] Chen, J.J., Yang, C.Y., Kuo, T.W. and Tseng, S.Y., (2007), April. Real-time task replication for fault tolerance in identical multiprocessor systems. In *13th IEEE Real Time and Embedded Technology and Applications Symposium (RTAS'07)* ,pp. 249-258. IEEE.
[42] Girault, A., Saule, E. and Trystram, D., (2009). Reliability versus performance for critical applications. *Journal of Parallel and Distributed Computing*, *Vol.69,No.3*, pp.326-336.
[43] Benoit, A., Canon, L.C., Jeannot, E. and Robert, Y., (2012). Reliability of task graph schedules with transient and fail-stop failures: complexity and algorithms. *Journal of Scheduling*, *Vol.15,No.5*, pp.615-627.
[44] Jakob, W. and Blume, C., (2014). Pareto optimization or cascaded weighted sum: A comparison of concepts. *Algorithms*, *Vol.7,No.1*, pp.166-185.
[45] Juve, G., Chervenak, A., Deelman, E., Bharathi, S., Mehta, G. and Vahi, K., (2013). Characterizing and profiling scientific workflows. *Future Generation Computer Systems*, *Vol.29,No.3*, pp.0682-692.
[46] Bharathi, S., Chervenak, A., Deelman, E., Mehta, G., Su, M.H. and Vahi, K., (2008), November. Characterization of scientific workflows. In *2008 third workshop on workflows in support of large-scale science* ,pp. 1-10. IEEE.

ABBREVIATIONS

DAG	Directed Acyclic Graph.
DCG	Directed Cyclic Graph
ETF	Earliest Time Finished
FTM	Fault Tolerance Manager
FTSA	Fault Tolerant Scheduling Algorithm
FTTs	Fault Tolerance Techniques
GA	Genetic Algorithms
HEFT	Heterogeneous-Earliest-Finish-Time
HPC	High Performance Computing
IaaS	Infrastructure as a Service.
IDC	International Data Corporation
MaxRe	Max Reliability
MOO	Multi-Objective Optimization
NSGA-II	Non-Dominated Sorting Genetic-II
OS	Operating System
PaaS	Platform as a Service
QFEC	Quantitative Fault-Tolerance With Minimum Execution Cost
QoS	Quality of Services
RA	Reliability Assessor
RD	Reliability-Driven
RM	Resource Manager
RR	Reliability Requirement
SWfMSs	Scientific Workflows Management Systems
TM	Task Monitor
TS	Task Scheduler
VM	Virtual Machine.
WEP	Workflow Execution Plan
XML	Extensible Markup Language
ZB	ZettaByte

LIST OF FIGURES

LIST OF THE PUBLICATIONS BY THE AUTHOR

1- Ali, A. A., Jasek, R., Krayem, S., & Zacek, P. (2017, April). Proving The Effectiveness Of Negotiation Protocols KQML In Multi-Agent Systems Using Event-B. *In Computer Science On-line Conference* (pp. 397-406). Springer, Cham. https://link.springer.com/chapter/10.1007/978-3-319-57264-2_40

2- Ali, A. A., Jasek, R., Krayem, S., Chramcov, B., & Zacek, P. (2018, April). Improved Adaptive Fault Tolerance Model For Increasing Reliability In Cloud Computing Using Event-B. In *Computer Science On-line Conference* (pp. 246-258). Springer, Cham. https://link.springer.com/chapter/10.1007/978-3-319-91192-2_25.

3- Ali, A. A., Vařacha, P., Krayem, S., Žáček, P., & Urbanek, A. (2018). Distributed Data Mining Systems: Techniques, Approaches And Algorithms. In *MATEC Web of Conferences*(Vol. 210, p. 04038). EDP Sciences. https://doi.org/10.1051/matecconf/201821004038.

4- Ali, A. A., Vařacha, P., Krayem, S., Jašek, R., Žáček, P., & Chramcov, B. (2018). Modeling Of Distributed File System In Big Data Storage By Event-B. In *MATEC Web of Conferences* (Vol. 210, p. 04042). EDP Sciences. https://doi.org/10.1051/matecconf/201821004042.

5- Capek, P., Jasek, R., Kral, E., Ali, A. A., & Senkerik, R. (2018, December). Cross Platform Configurable ERP Framework. (2018) *International Conference on Computational Science and Computational Intelligence (CSCI)* (pp. 1456-1457). IEEE.

6- Ali, A. A. , Krayem, S., Lazar,I , Kady ,M, Awwama, E. (2018). Solving NP-complete problem using formal method event-B. *9th Comparative European Research,CER 2018* (issue I.). http://www.sciemcee.org/library/proceedings/cer/cer2018_proceedings01.pdf

7- Ali, A. A., Krayem, S., Chramcov, B., & Kadi, M. F. (2018). Self-Stabilizing Fault Tolerance Distributed Cyber Physical Systems. *Annals of DAAAM & Proceedings*, *29*. https://www.daaam.info/Downloads/Pdfs/proceedings/proceedings_2018/169.pdf.

CURRICULUM VITAE

• PERSONAL INFORMATION

- **Full name:** Ammar Nassan Alhaj Ali.
- **Nationality:** Syrian.
- **Birth Date:** 28-2-1977.
- **Phone:** +965 960 82 510 (Kuwait).
- **Email:**Ammar282n@hotmail.com.

• EDUCATION

- Diploma Degree of Computer Engineering 2002, Aleppo University-Syria (5 years).
- Doctoral (PhD), Tomas Bata University in Zlin, Czech Republic, Since 2016.

• EXPERIENCE

- **Computer engineer:** Syria, Syrian Railways Corporation (2003-2005).
- **Senior web developer: Kuwait**, Ministry of education (2005-2021).
- **Software engineer:** Kuwait, Public AWQAF Foundation (2012-2013).
- **Lecturer:** in Computer Science Department, College of Basic Education (Kuwait -2015-2019).

• PROFESSIONAL PROFILE

- 18 years of professional experience in software design, development, debugging, deployment, documentation and testing of Client–Server, Web based Applications, Winforms and WPF.
- Extensive Knowledge in .NET Framework (All versions) and SQL Server.

• TECHNICAL SKILLS:

- **Languages:** C#, Python, JAVA, C, C++, PHP, VB.net and .NET Framework (All versions).
- **Machine learning:** Scikit-learn, Tensorflow, Pytorch, Keras.
- **Web Technologies:** ASP.NET, ASP, JavaScript, Web Services, JQuery, AJAX, HTML, DHTML, HTML5, CSS, CSS3, XML and ADO.NET.
- **RDBMS:** SQL Server (All versions), MySQL.
- **Report:** Crystal Report (All versions).

Fault Tolerance for Big Data Scientific Workflows in Cloud Computing Environments

Spolehlivost a odolnost vůči poruchám cloudových systémů pro automatické řízení a koordinaci procesů zpracování rozsáhlých a heterogenních vědecko technických dat

Doctoral Thesis Summary

Published by: Tomas Bata University in Zlín,

nám. T. G. Masaryka 5555, 760 01 Zlín.

Edition: published electronically

Typesetting by: Ammar Nassan Alhaj Ali

This publication has not undergone any proofreading or editorial review.

Publication year: 2021

ISBN 978-80-…….