Webology (ISSN: 1735-188X)
Volume 18, Number 6, 2021

Efficient Fault Tolerant Cost Driven Mechanism For Scientific Workflow Through Optimal Replication Strategy In Cloud Computing Environment

Asma Anjum[1] , Dr. Asma Parveen[2]

[1]Department of Computer Science and Engineering, Khaja Banda Nawaz College of Engineering, Kalaburagi, India.

[2]Department of Computer Science and Engineering, Khaja Banda Nawaz College of Engineering, Kalaburagi, India.

Abstract

Cloud Computing has been one of the distributed and effective computing paradigms; it provides enormous opportunities to tackle the scientific problems that possesses large scale attribute. Despite of being such a flexible computing paradigm, it possesses several challenges and fails to achieve the required QoS. Reliability requirement is one of the most important quality of services (QoS) and should be satisfied for a reliable workflow in cloud computing. Primary-backup replication is an important software fault-tolerant technique used to satisfy reliability requirement. Recent works studied quantitative fault-tolerant scheduling to reduce execution cost by minimizing the number of replicas while satisfying the reliability requirement of a workflow on heterogeneous infrastructure as a service (IaaS) cloud. However, a minimum number of replicas does not necessarily lead to the minimum execution cost and shortest schedule length in a heterogeneous IaaS cloud. In this research work, we develop a ODS (Optimal duplication strategy) for fault tolerance and cost driven mechanism also named as ODS-FTC; ODS-FTC uses the iterative based approach that selects VM and its duplicates that has minimum makes pan in case of individual task. Moreover, this provides the utility against failure occurrence and optimal selection with optimal redundancy causes the cost to be optimal. ODS-FTC is evaluated considering the scientific workflow like cyber shake, LIGO, montage and SIPHT; evaluation is carried out through designing instances. Furthermore, in case of all instances, ODS-FTC is proved to be marginally improvised than the existing model.

Keywords: Fault Tolerance, Reliability Requirement, Cost optimization, makes pan, efficient scheduling,

1 Introduction

Cloud Computing services are considered as primary effective commercial service model for computation that provides the computing platform as well as computing resources to its users; moreover, users can opt for “pay as you go”. Moreover, this virtual computing model provides the flexibility for users to present the requirement of QoS to providers [1]- [5]. Moreover, recent development in cloud computing have caused the extensive development in workflows application in various fields such as astrophysics, astronomy and bioinformatics in order to analyze these applications considering the CC (Cloud Computing) platforms. Further, characteristics of CC-model includes the dynamic resource allocation, storage resources; moreover, these characteristics can be exploited through efficient scheduling which solves the specific problems discussed later in same section to improvise the system performance [6]- [9]. Workflow scheduling is designed to optimize the heterogeneous cloud model; in here users focus on the QoS satisfaction which includes the cost execution, deadline while submitting the workflow applications. Furthermore, increase in demand for computation and services in scientific workflow application possesses problem of energy consumption, deadline constraint, makes pan optimization and cost minimization. thus, workflows are modelled through DAG (Direct Acyclic Graph); DAG is a workflow modelling where node is task and edge is the interlink among the task [10]-[12].

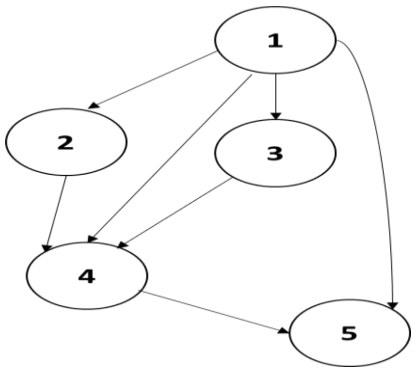

Figure 1 typical DAG model

Epigenomics workflow is a highly pipelined biology application which maps the epigenetic state of human cells. Most of its tasks have high CPU and low I/O utilization. LIGO workflow is used in the physics field to detect gravitational waves and has many CPU intensive tasks that consume large memory. [33]. We can synthesize workflows with different numbers of tasks using the generator provided by Pegasus project [34] and these workflows are available in the format of DAX (Directed Acyclic graph in XML). Execution time of the tasks in DAX files is based on a quad core, 2.4 GHz Intel Core 2 processor whose processing capacity is approximately equal to 8 ECUs ($2.33 \times 4/1.2 \approx 8$). For each of these workflows, we consider three sizes: Small (about 50 tasks), Medium (about 200 tasks) and Large (about 1000 tasks). Moreover, for each size 20 different instances are generated with the same structure but with different communication and computation workload. In general energy aware scheduling is developed by researchers which focuses on the green computing and tries to minimize the cost; most probably DVFS is used as the mechanism to reduce the energy. However, energy aware mechanism only focusses on the energy minimization and ignores fault tolerance and cost. Scientific workflows require huge number of resources to process the big data on clouds; furthermore, real time cloud services demand various Computation capacities which causes the increase in transient failure. Further, increase in complexities and failures puts an adverse effect on the resource management which results in QoS issue, especially reliability requirement [13]- [15]. Fault tolerance scheduling mechanism is one of the effective mechanisms to improvise the workflow reliability and further backup is used for satisfying the reliability requirement. Existing fault tolerance model which has been discussed in the next section uses one backup in case of failure; although as a novel concept it was interesting earlier but due to complex scientific workflows it fails model fails to tolerate more than one failure. Thus, it is necessary to design and develop a fault tolerance model that can tolerate multiple failure and enhance the reliability requirement; also cost optimization is one of the essential as optimal cost indicates the model efficiency [16] [17].

1.1 Motivation and contribution of research work

Fault-tolerant scheduling is an effective method to enhance the reliability of a workflow, and primary-backup replication is an important software fault-tolerant technique used to satisfy the reliability requirement. Existing fault tolerant scheduling algorithms either use one backup for each primary to tolerate one failure based on the passive replication scheme [18], [19], [20], which cannot tolerate potential multiple failures, or use fixed ε backups for each primary to tolerate ε failures in the same time based on active replication scheme, which can satisfy the reliability requirement, but can cause high redundancy and cost. Further, contribution of research work is highlighted through below points.

1. We design and develop a fault tolerance and efficient mechanism which satisfy the reliability requirements; moreover, proposed mechanism is named ODS-FTC (optimal duplication strategy with fault tolerance and cost optimization).
2. ODS-FTC uses the iterative approach for selection of VM and available duplicates that has minimum redundancy.
3. ODS-FTC is evaluated considering the cost parameter; In general, as the VM fails it requires more resources to cope up the failure and thus there is certain spike in cost.
4. Furthermore, scientific workflow is considered to prove the model efficiency; in order to evaluate four instances are designed that includes the certain number of VM.
5. comparative analysis is carried out and proposed methodology proves to be efficient than the existing model.

This research is organized in particular way as first section starts with background of computation and need of cloud computing phenomena; further we discuss the necessity of scheduling mechanism and end the section by motivation and contribution of research work. Second section focuses on the reviewing the various existing protocol and their shortcoming; further in third section ODS-FTC along with its mathematical formulation is designed. ODS-FTC is evaluated in fourth section by considering the various instance.

2 Related Work

Scientific workflow scheduling along with the optimization is considered as one of the essential research topic in cloud computing and further several aspects have been explored such as varying the number of workload, workflows; different platforms, different scheduling mechanism. Furthermore, in case of all the optimization objectives remains make span, energy consumption, cost, reliability or multi-objective; hence this section focuses on different related work in accordance with the fault tolerance along with the cost optimization. In [18], an online scheduling algorithm was developed to make robust mechanism against the missing information; in here, it was observed that in case of potential resource failure, workflow scheduling is more complicated. Further, a failure aware mechanism was proposed through Markov chain based prediction model of resource availability in [19]; however, the model was highly dependent, another dependent model was developed in [20] where replication strategy was adopted and additional schema was adopted in case of further failures, this results in high performance penalties, thus in [21], work queue with replication aka WQR was introduced considering that resource provisioning in cloud is elastic .In general fault tolerance is achieved through two distinctive approach i.e. passive replication [22] and active replication [23]; moreover, fault tolerance strategy can also be addressed as the improvisation in reliability which further minimizes the cost [24]- [27]. Passive replication refers to the backup replica which will be executed considering the VM whenever there is failure of primary one; however, it is quite difficult to adopt the passive replication through only backup [28] [29]; also, passive replication adoption costs are higher and the replication process leads the unpredictable redundancy. In case of active replication, primary one is replicated number of times in given each VM, where the task can be executed successfully. In [23], Max Re algorithm were presented for reliability requirement which produce the cost as well as redundancy; in here reliability requirements of entire task are similar and the requirement of each task is mathematically to n square root of defined reliability requirement where n is considered as the workflow. In [24], optimal resources mechanism is developed for assurance of reliability requirement for minimizing the reliability requirements which was compared with the previous mechanism; however, it possesses high cost to minimize the reliability requirement and redundancy. Similarly, [24] introduced ERRM approach which performs the iterative based approach for further implementation of quantitative based replication; this model does possess low-cost redundancy minimization for smaller workflows; however, fails miserably when used in large workflows this makes it as poor efficiency model.

The proactive fault-tolerance methods are applied in fault prediction as to replace the potential sections that would encounter fault, thus preventing fault event. [30] introduced a fuzzy task distribution method where by correct load distribution, the user task requests among available resources assure fault-tolerance by applying fault detection and discrepancies, which in turn reduce fault events in the system. Their experimental results indicate their method, when compared to other load balance algorithms, although the evolutionary and multi objective algorithms and energy utilization are not addressed, but it reduces fault event in a significant manner. [31] modified NSGA-III algorithm where intelligent fault tolerance technique is added on to increase system utilization. Moreover, the multiagent devices applied in their approach are highly contributive in many engineering aspects like service provision transportation, intelligent networks, cloud computing and complicated parallel computational devices. Therefore, their proposed intelligent system provides relatively stable assessment without evaluating fault distribution upon transient fault events. The hybrid fault-tolerance methods are a hybrid of reactive fault tolerant and proactive fault tolerant methods. [32] introduced a hybrid method for network fault tolerant and hardware fault tolerant where the square matrix multiplication concept is applied. Because in cloud computing data storage in a network takes place in remote sense, most faults occur due to system failure and network congestion. In their method, the servers' health is assessed through a health monitor, which predicts fault events and applies a migration technique to reduce data loss while no discussion is run on evolutionary and multi-objective algorithms and prediction through neural network.

3 Proposed Methodology

In general, cloud computing possesses high failure rates due to the existence of huge number of components and servers that are filled with workloads; moreover, these failures may lead to the constraint in VM availability; however, this issue is solved through the optimal fault tolerance strategy. Hence this section of the research presents the mathematical modelling of proposed methodology ODS-FTC which aims at providing the optimal reliability requirement and minimize the cost further. ODS-FTC comprises various sub section which are discussed below.

3.1 Preliminaries

Let's consider a particular set of Virtual Machine configuration denoted through variable F where $\mathrm{F} = \{\mathrm{F}_0, \mathrm{F}_1 \ldots\ldots, \mathrm{F}_\mathrm{m}\}$; moreover, this configuration comprises different parameter such as cost, memory, total number of VM and memory. Furthermore, considering the VM set, VM instance can be designed through the variable named V which is given as $\mathrm{V} = \{\mathrm{V}_0, \mathrm{V}_1, \ldots \ldots, \mathrm{V}_\mathrm{h}\}$ where V_h directly implicates the instances of VM with configurations; also, it is to be noted that ODS-FTC is designed for the parallel processing-based machine.

3.2 Workflow modelling

Let's consider a scientific workflow model with variable Z, which is further elaborated as$\mathrm{Z} = (\mathrm{W}, \mathrm{F})$ where both semi variables indicate the task set and dependencies; dependencies are commonly observed in the complicated, scientific and large workflow model. Furthermore, we initialize few additional parameters that are associated with the task are $\alpha(\mathrm{v}_\mathrm{x})$, $\beta(\mathrm{v}_\mathrm{x})$, $\gamma(\mathrm{v}_\mathrm{x})$ and $\delta(\mathrm{v}_\mathrm{x})$; for instance, if the task from v are performed with respect to the resources w, then workflow cost model can be mathematically computed through below equation.

$$\beth(\mathrm{v}_\mathrm{x}, \mathrm{v}_\mathrm{y}) = \begin{cases} \dfrac{\mathrm{ed\_wt}_{\mathbb{wx}}}{\mathrm{band}_{\mathbb{k},\mathbb{l}}} & \text{if } \mathrm{V}_\mathrm{l} \text{ is not equal to } \mathrm{V}_\mathrm{m} \\ 0 & \text{if } \mathrm{V}_\mathrm{l} \text{ is equal to } \mathrm{V}_\mathrm{m} \end{cases} \tag{1}$$

Also bandwidth resources can be designed as

$$\aleph_{l,m} = \mathrm{opt}\big(\aleph(\rho(V_l)), \aleph(\rho(V_m))\ \big) \tag{2}$$

3.3 Modelling of task with respect to the fault tolerance

Let's assume that individual task in task set is given same interval to perform the execution and the interval parameter of task can be mathematically designed as below equation; where n_p is considered as the ideal fault tolerance level along with the frequency of operation denoted as i_p

$$R_p = \left[\sqrt{\left(\left((n_p\, F_p)(F_t i_p)^{-1}\right) - 1\right)}\right] \tag{3}$$

The above equation makes the situation ideal where there is no occurrence of failure; which can be given as

$$\tau_n = F_p + R_p \times f_v \times i_p \tag{4}$$

Further, we assume that task w_x assigned to resource V_l and considering the above ideal situation, ideal execution can be formulated as below equation where $Q(w_x, V_l)$ implicating the total number of tasks along with Q_l as overhead.

$$\omega_{best}(w_x, V_l) = (\tau(w_x\ ))\ (vn((V_l)))^{-1} + Q(w_x, V_l). Q_l \tag{5}$$

Meanwhile, we compute the length of interval

$$\varepsilon(w_x, V_l) = vm(\tau(V_l))\ (\tau(w_x))^{-1}. (Q(w_x, V_l) + 1)^{-1} \tag{6}$$

Once the ideal case is designed, it is also important to formulate the worst situation where there is highest error occurrence with task and virtual machine defined earlier; further $Q(w_x, V_l)$implicates the total overhead and fault tolerance overhead is given through $\varepsilon(w_x, V_l)$

$$ET_{y}(w_x, V_l) = \ ((w_x))^{-1}. \left(vm\ (\tau(V_l))\right)^{-1} + seg(w_x, V_l). I_{max} + 2. P(w_x, V_l). Q_l \tag{7}$$

Henceforth, in order to optimize the worst situation, interval is optimized and the optimality of same is calculated through below equation

$$Q_{opt}(w_x, V_l) = \sqrt{\left(I_{max}(R_l)^{-1}\ (referload(w_x)). \left(vm\ (\tau(V_l))\right)^{-1}\right) - 1} \tag{8}$$

Moreover, in order to tolerate the fault, error probability is formulated which further defines the task reliability.

$$\xi(w_x, V_l, I) = I! \left(f^{-ɥ_l.ET_{y}(w_x,V_l)}(\lambda_{k}. ET_{y}(w_x, V_l)^{G})\right)^{-1} \tag{9}$$

Meanwhile task reliability is defined as the probable state where tasks are executed even there is failure; thus, the chances of being executed in successful manner is $\xi_{succed}(I, V_l\ ) = f^{-ɥ_l.ET_{y}(w_x,V_l)}$. Thus, reliability with consideration of task set is formulated as:

$$R(w_x, V_l) = \sum_{I=0}^{I_{max}} Pr(w_x, V_l, I_{max}) \cdot \xi_{succed}(I, V_l) \quad (10)$$

3.4 Reliability Requirement

In general, two distinctive types of system failure i.e., permanent failure and transient failure; moreover, this study considers the second type of failure; thus, we design the reliability with respect to an event in given time u.

$$\varsigma(u) = e^{-\upsilon v} \quad (11)$$

In the above equation υ i.e., nu indicates the frequent failure in a given unit time; υ_l is used to indicate the constant failure of given VM; further reliability of o_j in given time on υ_l is formulated as:

$$S(o_j, \upsilon_l) = e^{-\upsilon_l x_h l} \quad (12)$$

further, failure occurrence without using ODS-FTC is given as:

$$1 - S(o_j, \upsilon_l) = 1 - e^{-\upsilon_l x_h l} \quad (13)$$

Furthermore, considering that each task possesses certain number of duplicates, thus we define $num_h (num_h \leq \lfloor T \rfloor)$ as O_h. Further, we define duplicate set o_h is given as;$\langle o_h^1, o_h^2, \ldots, o_h^{num_h} \rangle$ where o_h^1 is absolute whereas other are duplicates; total number of duplicates for designed workflow is:

$$num_{dup(H)} = \sum_{h=1}^{|O|} num_j \quad (14)$$

once o_h duplication is completed, it is observed that there is no failure occurrence and reliability is updated as:

$$S(o_h) = 1 - \sum_{y=1}^{num_h} \left(1 - S\left(o_h^y, v_{pr(o_h^y)}\right)\right) \quad (15)$$

In the above equation, $v_{pr(o_h^y)}$ indicates o_h^y

$$S(H) = \sum_{o_h \epsilon O} S(o_h) \quad (16)$$

3.5 Cost Modelling of ODS-FTC

State of efficient fault tolerance is reducing the cost execution with the reliability which can be defined as: considering the designed workflow with provided VM set discussed earlier in same section; the problem is to assign the duplicates along with their VM for each individual task.

Meanwhile execution cost is reduced and assurance of fault tolerance through designed reliability requirement in above section. Thus, optimal assignment of duplicates (also known as backups) along with VM assignment is given as:

$$\mathrm{cost(h)} = \sum_{o_j \epsilon O} \mathrm{cost}(o_j) \tag{17}$$

Subject to reliability requirement

$S(H) = \sum_{o_j \epsilon O}\left(S(o_h) \text{ is less than or equivalent to } S_{rq}\right)$

3.6 Reliability Requirement for individual task

We formulate and satisfy the reliability requirement of individual tasks; at first, we compute the individual task reliability requirement considering the absolute reliability obtained through the earlier allocations in below equations.

$$S_{rq}\left(o_{sq(i)}\right) = \sqrt[1-i+\lfloor O \rfloor]{S_{rq}(H)\left(\sum_{y=1}^{i-1} S\left(o_{sq(y)}\right)\right)^{-1}} \tag{18}$$

in above equations, $o_{sq(i)}$ indicates the given ith tasks; further we optimize the reliability requirements through given points.

while computing the reliability requirements for individual task, we consider $\sqrt[\lfloor O \rfloor]{S_{rq}(H)}$ is in the upper bound on the reliability requirements for task O_h and it is given by

$$S_{URQ}(O_h) = \sqrt[\lfloor O \rfloor]{S_{rq}(H)} \tag{19}$$

Furthermore, it is assuming that the task assigned is $o_{sq(i)}$ for task I then the task set is given as the unassigned task $[o_{sq(1)}, o_{sq(2)}, \ldots\ldots, o_{sq(i-1)}\ ]$ and assigned task $[o_{sq(1)}, o_{sq(2)}, \ldots\ldots, o_{sq(i-1)}\ \ ]$. Further, ODS-FTC model assumes that each task in the workflow model is assigned with VM along with reliability parameter value formulated in above equation that provides the reliability assurance. Hence, the reliability requirement as whole is computed as;

$$S_{rq}(H) = \sum_{y=1}^{i-1} \left(S\left(o_{sq(i)}\right)\right)\left(S\left(o_{sq(y)}\right)\right)\left(\sum_{z=i+1}^{[O]} S_{URQ}o_{sq(z)}\right) \tag{20}$$

Also for individual task it can be computed as equation below; moreover, to satisfy the reliability iterative approach is used to choose duplicate and Vm that has minimum makespan.

$$S_{rq}\left(o_{sq(i)}\right) = \left(S_{rq}(H)\right)\left(\sum_{y=1}^{i-1} \left(S\left(o_{sq(i)}\right)\right)\left(S\left(o_{seq(y)}\right)\right)\left(\sum_{z=i+1}^{[O]} S_{URQ}o_{sq(z)}\right)\right)^{-1} \tag{21}$$

Further, we design an algorithm which reduces the cost and provides the efficient fault tolerance towards the workflow

3.7 ODC-FTC Algorithm

The main intention of algorithm is to offload the reliability requirement on sub division considering each and every individual task; further proposed algorithm i.e., ODS-FTC minimizes the execution cost through selecting the duplicates and optimal VM; here optimal VM are the one that has efficient execution time; however, there is still some redundancy was observed; also, it was observed that few of the duplicates can be discarded.

Table 1 ODS-FTC algorithm

Step1:	Start
Step2:	Input as DAG information such as nodes set, execution time, communication time, VM set reliability requirement. Output: Reliability value, cost, schedule length
Step3:	Task ordering through in descending order
Step4:	$\mathrm{for}(k = 1; k \leq \lvert O\rvert; k++)\mathrm{do}$ compute $S_{rq}(O_{sq(k)})$ $A_{sq(k)} = 0$ $So_{sq(k)} = 0$
Step5:	Define a backup list $\mathrm{dup}(o_{sq(k)})$ and store it in $o_{sq(k)}$
Step6:	$\mathrm{for}\ (l = 1; l \leq \lvert V\rvert; l++)\mathrm{do}$ $\mathrm{Compute}S_{rq}(o_{sq(k)}, v_l)$ for $o_{sq(k)}$ Compute opt_finish_time end for
Step7:	while $\left(S\left(o_{sq(k)}\text{is less than } S_{rq}\left(o_{sq(k)}\right)\right)\right)$ do choose replica $o^{y}_{sq(k)}$ and VM $v_{pr(o^{y}_{sq(k)})}$ with optimal execution time $x_{seq(k),pr(o^{x}_{sg(j)}}; num_{sq(k)}++;$
Step8:	Place $o^{y}_{seq(k)}$ to the dup_list in descending order; further discard the earlier allocations $num_{sq(j)} = 0$ $S(o_{sq(k)}) = 0$
Step9:	while $S(o_{sq(k)})$ is less than $S_{rq}(o_{seq(k)})$ do choose duplicate $o^{y}_{sq(k)}$ and $v_{pr(o^{y}_{sq(k)})}$ in given list; also discard duplicate $o^{y}_{seq(k)}$ from given list and increment $num_{seq(k)}$

Step7:	$\text{compute finish}_{\text{time}}\left(o_{sq(k)}^{y}\right) = \text{opt\_finish\_time}(n_{sg(j)}^{x}, u_{pr(n_{sg(j)}^{x})})$
Step8:	compute $S(o_{sq(k)})$ end while (step6)
Step9:	end for
Step10:	compute makespan, cost, reliability

In ODS-FTC algorithm schedules the task in an order; optimal order is computed through below equation where w_i is the execution time for task o_h.

$$\eta_v(o_h) = x_h + \max_{o_h \in scc(o_h)}\{d_{h,i} + \eta_v(n_j)\} \quad (22)$$

Further, ODS-FTC choose the duplicates with optimal VM and these VMs are reserved and sorted in optimal order. Once, optimal VMs are sorted, ODS-FTC clears the earlier allocations and further assignment of duplicates are carried out. ODS-FTC chooses only VM that has higher order of reliability and further duplicates, execution cost and optimal makes pan are computed.

4 Performance Evaluation

Cloud Computing resources has emerged as one of the efficient computing models in the real time for vibrant uses, accessible, cost effective and can be accessed from anywhere through internet. Moreover, it can be applied to various application, one of the applications is workflow scheduling of scientific workflows where the tasks are dependent and independent, also possesses various merits and demerits. This section presents the simulation results of the proposed algorithms and their detailed comparisons; Simulation methods are now commonly used to test novel scheduling algorithms for workflow scheduling problems. It enables researchers to evaluate the performance of their proposed algorithms under a controlled setting and repeatable manner. For this purpose, here the simulations are conducted using Java coding environment on an Intel(R) Core (TM) i5-8300H CPU with 2.30 GHz and 8GB RAM running on Windows 10.

To evaluate our proposed algorithms with a realistic workload, we consider five scientific workflow applications with different data and computational characteristics: Montage (I/O intensive), Cyber Shake (data intensive), Epigenomics (CPU intensive), LIGO (memory intensive), and SIPHT (CPU intensive). Each of these workflows has different structure as shown in Fig. 2 and we can see that they are different combinations of the basic structural components (pipeline, data distribution, data aggregation, and data redistribution).

4.1 Instance Design

In order to evaluate the model, we have designed four distinctive instances; each instances comprises certain number of VM and workflow variant; Instance based comparison helps in performance evaluation in a random way. Moreover, four workflows are considered i.e., cyber shake, Inspiral, Montage and SIPHT; in case of cyber shake workflow, first instance is design considering the workflow variant of cybershake_30 and 20 virtual machine, second and third instance comprises cysber shake 50 with 40 virtual machine and cyber shake 100 with 60 virtual machines respectively. Moreover, fourth instance comprises 80 virtual machine and cybershake_1000. Similarly, for Inspiral_30, Inspiral_50, Inspiral _100 and Inspiral _1000, number of virtual machines is 20, 40, 60 and 80 for four distinctive instances respectively. Furthermore, in case montage and SIPHT, number of virtual machines remains same with varying in workflow variant.

4.2 Workflow Cost comparison

After designing the Instance, comparison for each instance is carried with respect to the execution cost;

4.2.1 Cyber shake

Cyber Shake is an earthquake science application used to characterize earthquake hazards through combining large datasets and has large requirements for memory and CPU.

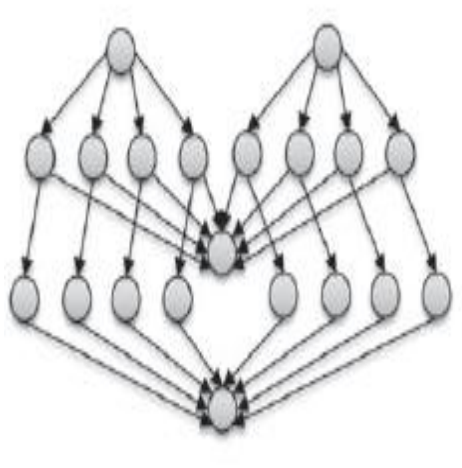

Figure 2 cyber shake

figure 2 shows the comparison of four instance considering the cyber shake workflow; In case of first instance execution cost for existing model is 19332.71 whereas proposed model execution cost is 18418.71. Similarly, in case of second and third instance execution cost of existing model is 21451.33 and 26222.62 whereas execution cost of ODS-FTC is 20038.49 and 23877.95 respectively. In case of fourth instance existing model cost is 177836.21 whereas proposed model cost is 153904.64.

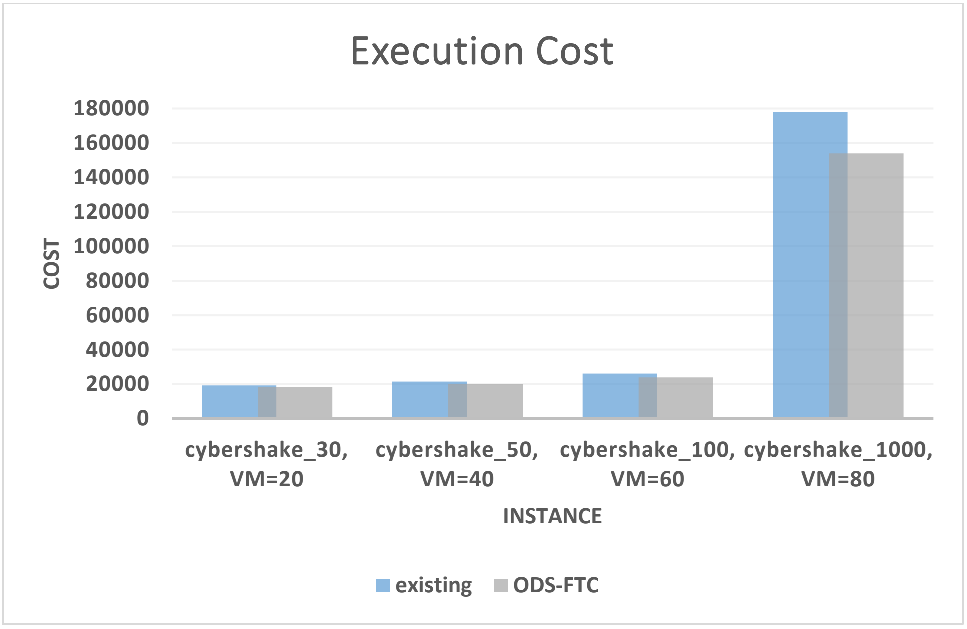

Figure 3 execution cost comparison for cyber shake work

4.2.2 LIGO workflow

LIGO workflow is used to generate and analyze gravitational waveforms from data collected during the coalescing of compact binary systems;

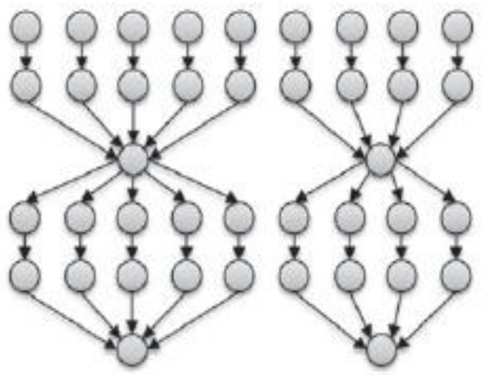

Figure 4 LIGO

figure 3 shows the execution cost comparison of existing and ODS-FTC. In case of first instances, cost of existing model is 19957.79 whereas cost of ODS-FTC is 13340.13. Similarly, in case of second and third instance 35468.35 and 63426.63 whereas ODS-FTC execution cost is 23705.17 and 42400.89 respectively. Furthermore, for fourth instance, existing model execution cost is 6886731.17 whereas ODS-FTC execution cost is 459009.39.

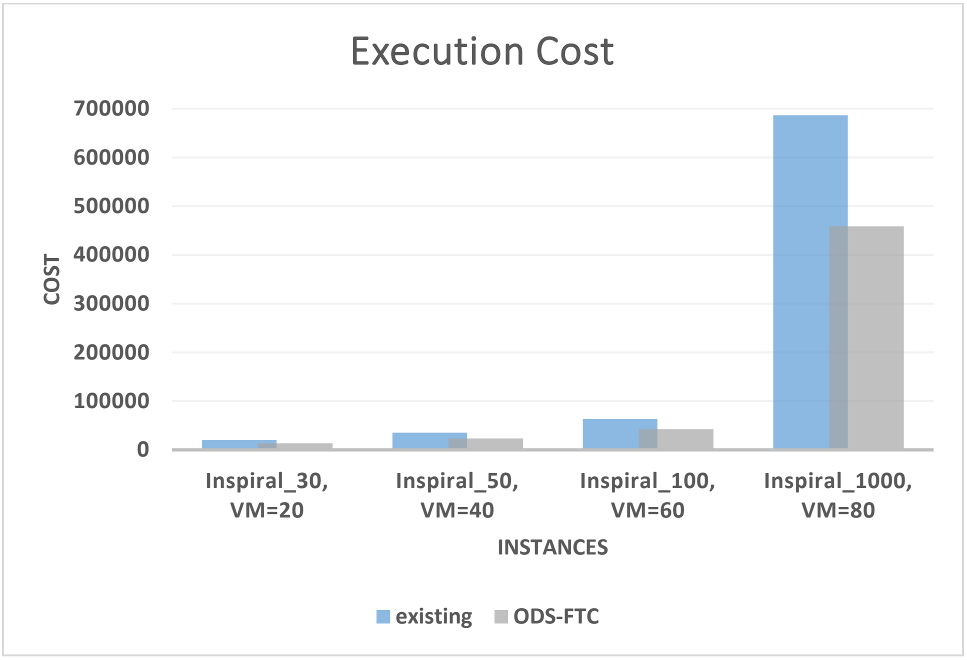

Figure 5 execution cost comparison for Inspiral work

4.2.3 Montage workflow

Montage workflow is used in astronomy with the aim of constructing the desired mosaic of the sky on the basis of input images. Most of its tasks perform frequent I/O operations and need less CPU capacity.

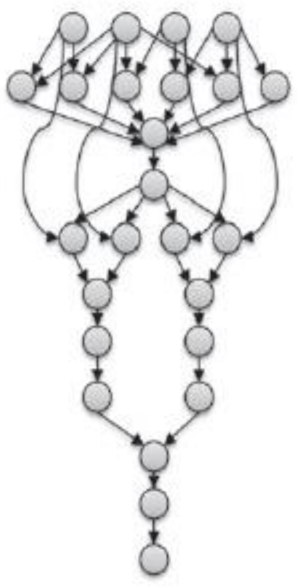

Figure 6 Montage workflow

4.2.4 Cost comparison

Figure 4 shows the execution cost comparison of existing and proposed ODS-FTC considering the four instances; in case of first and second instance, execution cost is 736.37 and 1628.23 whereas cost execution of ODS-FTC is 508.31 and 1118.82 respectively. Similarly, for third and fourth instance, execution cost of existing model is 3430.49 and 36032.25 in comparison with the proposed model cost with 2349.4 and 24636.46 respectively.

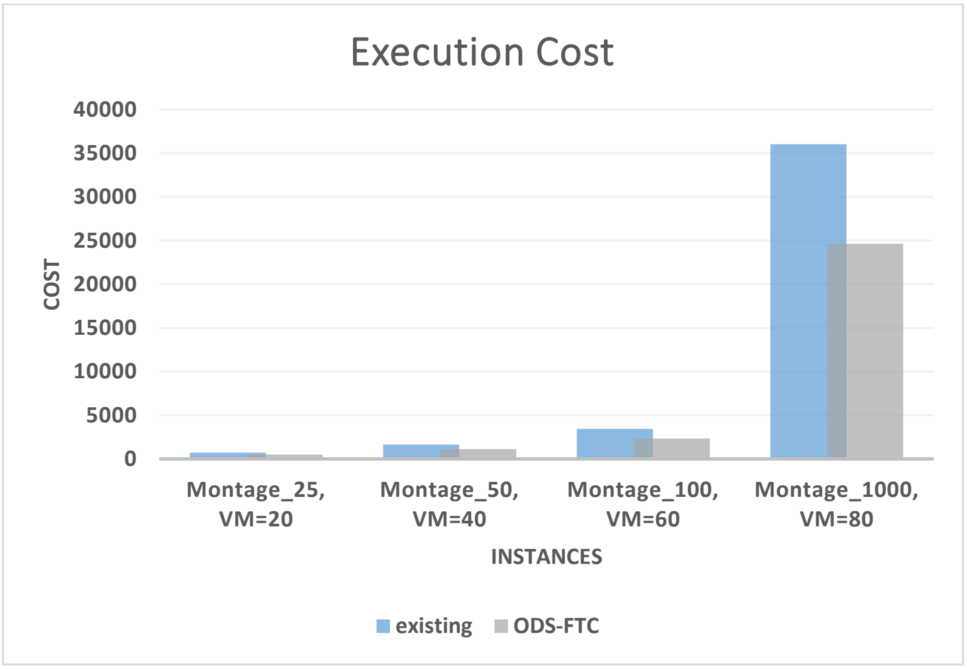

Figure 7 montage cost comparison

4.2.5 Makes pan comparison

Figure 3 shows the makes pan comparison in all four instances; In general, makes pan is defined as the complete time required to complete the task from start to end. In case of first instance, existing model makes pan is 58.72 whereas ODS-FTC takes 48.67; in case of second instance, existing model takes 97.84 whereas proposed model takes 82.13. Similarly in case of third and fourth instance existing model makes pan is 125.93 and 100.84 respectively whereas ODS-FTC makes pan is 299.84 and 1679.35 respectively.

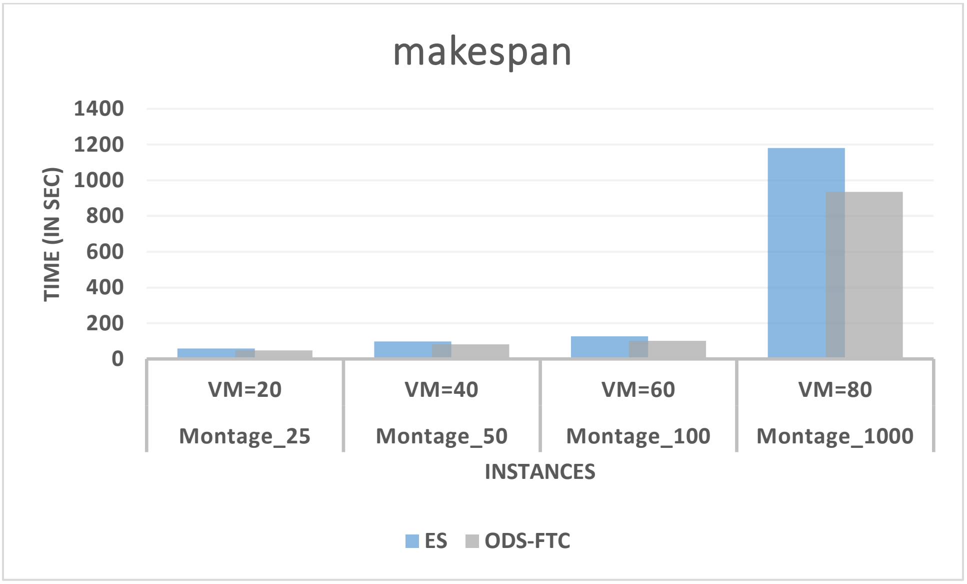

Figure 8makespan comparison

4.2.6 SIPHT workflow

SIPHT workflow is used for automating the search of sRNA encoding-genes for bacterial replicons from bioinformatics.

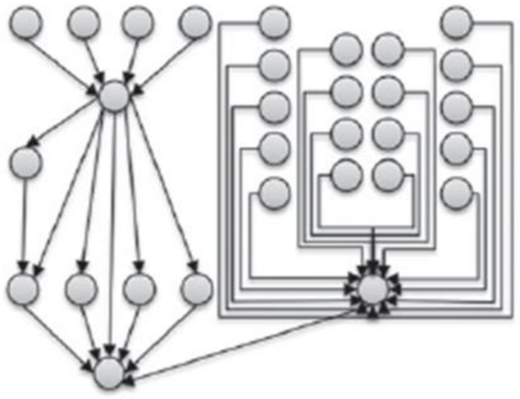

Figure 9 SIPHT

In case of first and second instance execution cost of existing model is 16668.77 and 35056.42 in comparison with the execution cost of ODS-FTC is 11121.29 and 23386.35 respectively. Similarly, in case of third and fourth instance execution cost of existing model is 52215.16 and 5216668.61 in comparison with the proposed model i.e., 34833.32 and 347999.82 respectively.

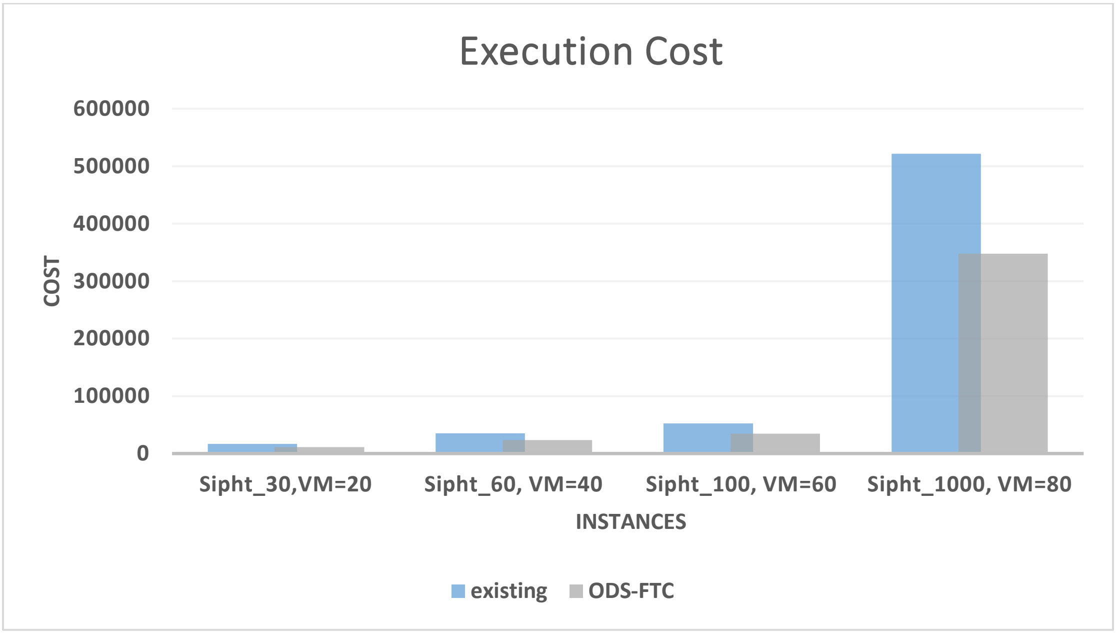

Figure 10 Cost Comparison of existing and proposed model.

4.3 Comparative analysis

Table 2 shows the improvisation of ODS-FTC considering cyber shake workflow; in case of first instance, second instance, third instance and fourth instance, ODS-FTC observes 4.72%, 6.58%, 8.94% and 13.45% respectively.

Table 2 cost improvisation over the existing model for cyber shake workflow

Instances	Improvisation
Instasnce1	4.72%
Instance2	6.58%
Instance3	8.94%
Instance4	13.45%

Table 3 shows the improvisation of ODS-FTC considering the LIGO workflow; in case of all four instance, ODS-FTC achieves 33.15%, 33.16%, 33.14% and 93.33% respectively.

Table 3 cost improvisation over the existing model for LIGO workflow

Instances	Improvisation
Instasnce1	33.15%
Instance2	33.16%
Instance3	33.14%
Instance4	93.33%

Table 4 shows the improvisation of ODS-FTC over the existing model considering the montage workflow; in case of all four instances, ODS-FTC observes the improvisation of 30.97%, 31.28%, 31.51% and 31.62% respectively.

Table 4 cost improvisation over the existing model for LIGO workflow

Instances	improvisation
Instasnce1	30.97%
Instance2	31.28%
Instance3	31.51%
Instance4	31.62%

Table 5 shows the improvisation of ODS-FTC over the existing model considering the montage workflow with respect to makes pan; in case of all four instances, ODS-FTC observes the improvisation of 17.11%, 16.05%, 19.92% and 20.76% respectively

Table 5 makes pan Improvisation over the existing model for montage workflow

Instances	improvisation
Instasnce1	17.11%
Instance2	16.05%
Instance3	19.92%
Instance4	20.76%

Table 6shows the improvisation of ODS-FTC over the existing model considering the SIPHT workflow; in case of all four instances, ODS-FTC observes the improvisation of 33.28% for three instances and 93.32% for fourth instances

Table 6 cost improvisation over the existing model for LIGO workflow

Instances	improvisation
Instasnce1	33.28%
Instance2	33.28%
Instance3	33.28%
Instance4	93.32%

Moreover, through the above comparative analysis it is observed that ODS-FTC possesses marginal improvisation over the existing model not only in terms of cost but also makespan.

Conclusion

Increase in cloud computing popularity along with its universal acceptance has led to be applicable in implementation in large scale application; moreover, in such cases cloud environment is chosen by scientific association for smooth execution of designed workflows. Despite of being so dynamic, cloud computing possesses higher number of failure rate; failure can occur due to different reason; these failure results in unavailability of virtual machine to process the work. This issue can be solved through designing the fault tolerance approach. Hence in this research work, a mechanism ODS-FTC (optimal duplication strategy for fault tolerance cost drive) is developed that aims at enhancing the reliability and execution cost. ODS-FTC uses the duplication strategy where ODS-FTC uses the iterative approach for selection of VM and available duplicates that has minimum redundancy; moreover, ODS-FTC not only minimizes the cost but also minimizes the makes pan. Furthermore, in order to evaluate the ODS-FTC model, four random instances were designed each instance contains the workflow variant and certain number of virtual machines; moreover, average instances with respect to cost shows the improvisation of 8.42%, 48.19%, 31.34% and 48.29% improvisation for cyber shake, Ligo, Montage and SIPHT workflow in respective manner; also, it is to be noted that in case of montage workflow an average of 18.46 % of improvisation is observed by ODS-FTC.

Although, ODS-FTC proved to be efficient fault tolerance with the cost optimization than the other mechanism for different scientific workflow, it has to be noted that it can vary from one workflow to other workflow depending on the workload and task complexity; thus, further research could be considering the varying the workflow model and complexities.

Reference

1. M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, M. Zaharia, ‘‘A view of cloud computing,’’ Commun. ACM, vol. 53, no. 4, pp. 50–58, 2010.
2. H. Chen, X. Zhu, D. Qiu, L. Liu, and Z. Du, ‘‘Scheduling for workflows with security-sensitive intermediate data by selective tasks duplication in clouds,’’ IEEE Trans. Parallel Distrib. Syst., vol. 28, no. 9, pp. 2674–2688, Sep. 2017.
3. M. A. S. Netto, R. N. Calheiros, E. R. Rodrigues, R. L. F. Cunha, and R. Buyya, ‘‘HPC cloud for scientific and business applications: Taxonomy, vision, and research challenges,’’ ACM Comput. Surv., vol. 51, no. 1, pp. 1–29, Apr. 2018.
4. G. B. Berriman, G. Juve, E. Deelman, M. Regelson, and P. Plavchan, ‘‘The application of cloud computing to astronomy: A study of cost and performance,’’ in Proc. 6th IEEE Int. Conf. e-Sci. Workshops, Dec. 2010, pp. 1–7.
5. Z. Lv and L. Qiao, ‘‘Analysis of healthcare big data,’’ Future Gener. Comput. Syst., vol. 109, pp. 103–110, Aug. 2020.
6. J. Ekanayake, T. Gunarathne, and J. Qiu, ‘‘Cloud technologies for bioinformatics applications,’’ IEEE Trans. Parallel Distrib. Syst., vol. 22, no. 6, pp. 998–1011, Jun. 2011.
7. Y. Liu, C. Yang, and Q. Sun, ‘‘Thresholds based image extraction schemes in big data environment in intelligent traffic management,’’ IEEE Trans. Intell. Transp. Syst., early access, Jun. 5, 2020, doi: 10.1109/TITS.2020. 2994386.
8. Z. Lv and W. Xiu, ‘‘Interaction of edge-cloud computing based on SDN and NFV for next generation IoT,’’ IEEE Internet Things J., vol. 7, no. 7, pp. 5706–5712, Jul. 2020.
9. Z. Lv and H. Song, ‘‘Mobile Internet of Things under data physical fusion technology,’’ IEEE Internet Things J., vol. 7, no. 5, pp. 4616–4624, May 2020.

10. H. Chen, J. Wen, W. Pedrycz, and G. Wu, ‘‘Big data processing workflows oriented real-time scheduling algorithm using task-duplication in geo distributed clouds,’’ IEEE Trans. Big Data, vol. 6, no. 1, pp. 131–144, Mar. 2020.
11. X. Zeng, S. Garg, M. Barika, A. Y. Zomaya, L. Wang, M. Villari, D. Chen, and R. Ranjan, ‘‘SLA management for big data analytical applications in clouds: A taxonomy study,’’ ACM Comput. Surv., vol. 53, no. 3, pp. 1–40, Jul. 2020.
12. Z. Li, J. Ge, H. Hu, W. Song, H. Hu and B. Luo, "Cost and Energy Aware Scheduling Algorithm for Scientific Workflows with Deadline Constraint in Clouds," in IEEE Transactions on Services Computing, vol. 11, no. 4, pp. 713-726, 1 July-Aug. 2018, doi: 10.1109/TSC.2015.2466545.
13. X. Tang, "Reliability-Aware Cost-Efficient Scientific Workflows Scheduling Strategy on Multi-Cloud Systems," in IEEE Transactions on Cloud Computing, doi: 10.1109/TCC.2021.3057422.
14. Zhongjin Li, Victor Chang, Haiyang Hu, Hua Hu, Chuanyi Li, Jidong Ge, Real-time and dynamic fault-tolerant scheduling for scientific workflows in clouds, Information Sciences, Volume 568, 2021,Pages 13-39,ISSN 0020-0255, https://doi.org/10.1016/j.ins.2021.03.003.
15. X. Zhu, J. Wang, H. Guo, D. Zhu, L. T. Yang, and L. Liu, ‘‘Fault-tolerant scheduling for real-time scientific workflows with elastic resource provisioning in virtualized clouds,’’ IEEE Trans. Parallel Distrib. Syst., vol. 27, no. 12, pp. 3501–3517, Dec. 2016.
16. I. Casas, J. Taheri, R. Ranjan, L. Wang, and A. Y. Zomaya, ‘‘A balanced scheduler with data reuse and replication for scientific workflows in cloud computing systems,’’ Future Gener. Comput. Syst., vol. 74, pp. 168–178, Sep. 2017.
17. R. Buyya, R. Ranjan, and R. N. Calheiros, ‘‘Modeling and simulation of scalable cloud computing environments and the Cloud Sim toolkit: Challenges and opportunities,’’ in Proc. Int. Conf. High Perform. Comput. Simul., Jun. 2009, pp. 1–11.
18. Z. Ahmad, A. I. Jehangiri, M. Iftikhar, A. I. Umer, and I. Afzal, ‘‘Data oriented scheduling with dynamic-clustering fault-tolerant technique for scientific workflows in clouds,’’ Program. Comput. Softw., vol. 45, no. 8, pp. 506–516, Dec. 2019.
19. J. Schneider, “Grid workflow scheduling based on incomplete information,” Ph.D. dissertation, Berlin Institute of Technology, 2010.
20. Z. Yu, C. Wang, and W. Shi, “Failure-aware workflow scheduling in cluster environments,” Cluster Computing, vol. 13, no. 4, pp. 421–434, 2010.
21. Y. Tao, H. Jin, S. Wu, X. Shi, and L. Shi, “Dependable grid workflow scheduling based on resource availability,” Journal of Grid Computing, vol. 11, no. 1, pp. 47–61, 2013.
22. M. Tao, S. Dong, and K. He, “A new replication scheduling strategy for grid workflow applications,” in Proceedings of 6th Annual Conf. Chinagrid. IEEE, 2011, pp. 74–80.
23. Q. Zheng, B. Veeravalli, and C.-K. Tham, “On the design of fault to lerant scheduling strategies using primary-backup approach for computational grids with low replication costs,” IEEE Trans. Comput., vol. 58, no. 3, pp. 380–393, Mar. 2009.

24. L. Zhao, Y. Ren, Y. Xiang, and K. Sakurai, “Fault-tolerant scheduling with dynamic number of replicas in heterogeneous systems,” in Proc. 12th IEEE Int. Conf. on High Performance Computing and Communications. IEEE, 2010, pp. 434–441.
25. G. Xie, G. Zeng, Y. Chen, Y. Bai, Z. Zhou, R. Li, and K. Li, “Minimizing redundancy to satisfy reliability requirement for a parallel application on heterogeneous service-oriented systems,” IEEE Trans. Serv. Comput., doi: 10.1109/TSC.2017.2665552, Feb. 2017.
26. G. Xie, G. Zeng, R. Li, and K. Li, “Quantitative fault-tolerance for reliable workflows on heterogeneous iaas clouds,” IEEE Trans. Cloud Comput., doi: 10.1109/TCC.2017.2780098, Dec. 2017.
27. A. Benoit, M. Hakem, and Y. Robert, “Fault tolerant scheduling of precedence task graphs on heterogeneous platforms,” in Proc. IEEE Int. Symp. Parallel and Distributed Processing. IEEE, 2008, pp. 1–8.
28. A. Girault and H. Kalla, “A novel bicriteria scheduling heuristics providing a guaranteed global system failure rate,” IEEE Trans. Dependable Secure Comput., vol. 6, no. 4, pp. 241–254, Oct. 2009.
29. G. Koslovski, W.-L. Yeow, C. Westphal, T. T. Huu, J. Montagnat, and P. Vicat-Blanc, “Reliability support in virtual infrastructures,” in Proc. IEEE 2nd Int. Conf. on Cloud Computing Technology and Science. IEEE, 2010, pp. 49–58.
30. F. Pop and M. Potop-Butucaru, “Adaptive resource management and scheduling for cloud computing,” Lecture Notes in Computer Science, vol. 8907, 2014.
31. H. Arabnejad, C. Pahl, G. Estrada, A. Samir, F. Fowley, A fuzzy load balancer for adaptive fault tolerance management in cloud platforms, in: European Conference on Service-Oriented and Cloud Computing, Springer, Cham, 2017, pp. 109–124, http://dx.doi.org/10.1007/978-3-319-67262-5_9.
32. B. Wu, K. Hao, X. Cai, T. Wang, An integrated algorithm for multi-agent fault-tolerant scheduling based on MOEA, Future Gener. Comput. Syst. 94 (2019) 51–61, http://dx.doi.org/10.1016/j.future.2018.11.001.
33. T. Tamilvizhi, B. Parvathavarthini, A novel method for adaptive fault tolerance during load balancing in cloud computing, Cluster Comput. 22 (5) (2019) 10425–10438, http://dx.doi.org/10.1007/s10586-017-1038-6.
34. R. Buyya, R. Ranjan and R. N. Calheiros, "Modeling and simulation of scalable Cloud computing environments and the Cloud Sim toolkit: Challenges and opportunities," 2009 International Conference on High Performance Computing & Simulation, 2009, pp. 1-11, doi: 10.1109/HPCSIM.2009.5192685.
35. Ewa Deelman, Karan Vahi, Gideon Juve, Mats Rynge, Scott Callaghan, Philip J. Maechling, Rajiv Mayani, Weiwei Chen, Rafael Ferreira da Silva, Miron Livny, Kent Wenger, Pegasus, a workflow management system for science automation, Future Generation Computer Systems, Volume 46,2015, Pages 17-35, ISSN 0167-739X, https://doi.org/10.1016/j.future.2014.10.008.

36. G. Yao, Y. Ding and K. Hao, "Using Imbalance Characteristic for Fault-Tolerant Workflow Scheduling in Cloud Systems," in IEEE Transactions on Parallel and Distributed Systems, vol. 28, no. 12, pp. 3671-3683, 1 Dec. 2017, doi: 10.1109/TPDS.2017.2687923.

Asma Anjum is a full-time research scholar at Visvesvaraya Technological University, Belagavi, Karnataka. She has completed her B. E and MTech in computer science and engineering in the year 2015 and 2017 respectively. Her research interest includes networking and cloud computing. She can be contacted at email: asmacs13@gmail.com

Dr. Asma Praveen got graduation in Electrical Engg., in 1993 and completed post-graduation in Computer Science and Engg in 2004 and in 2016 she was awarded Ph.D.in Computer Science and Engg. She has a rich experience of teaching for more than 25 years. She has published 30 research papers in leading international journals and conference proceedings, and presented three papers in national conferences.

• Life member of Institute of Electronics and Telecommunication Engineers (IETE).

• Treasurer of IETE Gulbarga Sub-Centre for 2018-19 and 2019-2020.