A Novel Multilevel Cost Effective Fault Tolerance (CEFT] Framework Approach for High Performance Computing [HPC] Cloud

Sharavana. K[1], Josephine Prem Kumar[2]

[1]*Associate Professor, Department of Information Science and Engineering, HKBK College of Engineering, Bengaluru, India*

[2]*Professor, Department of Computer Science and Engineering, Cambridge Institute Of Technology, Bengaluru, India.*

[1]sharatanuj@gmail.com, [2] josephine.prem@gmail.com

Abstract-

Cloud with HPC is capable of handling large applications like scientific workflow on scalable and powerful hardware without owning or maintaining it Although cloud computing is flexible, it has several limiting factors that lead to failure of the required Quality of Service (QoS). The most essential QoS for this computing is the requirement relating to reliability that should be achieved for the cloud computing workflow. In the most recent research on scheduling, this technique is done to decrease the expense of execution by reducing the count of replicants such that the necessary reliability constraint is met for the workflow on IaaS (infrastructure as service) clouds that is heterogeneous. This research work proposes Multilevel CEFT-HPC-Cloud (Cost Effective Fault Tolerance) mechanism in HPC Cloud; CEFT-HPC-Cloud is a multilevel scalable mechanism that chooses virtual machines along with its duplicates with optimal scheduling length at various levels. This further provides the edge in case of any failure, also reduction in redundancy and VM selection leads the expense to be optimal. Moreover, CEFT-HPC-Cloud develops an optimal algorithm that aims to reduce redundancy, cost and reliability awareness through offloading the task. CEFT-HPC-CLOUD is evaluated considering the workflow such as LIGO, SIPHT, *cybershake* as well as *montage*. This calculation is performed by the design of instances. Further, considering all instances cases, this mechanism of CEFT-HPC-Cloud is proven more improved in comparison to the model already existing.

DOI Number: 10.14704/nq.2022.20.9.NQ44795 Neuro Quantology 2022; 20(9):6800-6812

1 Introduction

The services of cloud computing are the primary model for commercial services that are effective for computational services that give the platform for computation and also the resources of computation to the consumer. The mode of payment "pay as you work" make it economical for the consumer. Also, the virtual model for computation makes it flexible for the consumers to put forward the necessary QoS requirements to the associated providers [1-5].

Lately, cloud computing advancements have resulted in large-scale development of the applications of workflow for different domains that include bioinformatics, astrophysics as well as astronomy for the analysis of applications based on platforms relating to cloud computing. Furthermore, the model features of cloud computing have dynamic resource allocation and storage of resources. However, these features can be maximized and exploited by the use of scheduling that is efficient that resolves the particular issues that are later discussed in this section to improve the performance of the system [6-9]. The scheduling workflow is developed for the optimization of the heterogeneous cloud model. The main focus of the consumers is on the satisfaction of QoS that involves the execution cost, and submission within the deadline for the given applications of workflow. Further, the expansion in the demand for services as well as computation for the

scientific application workflow has the issues of consumption of energy, constraints on the deadline, optimization of makespan as well as expense minimization. Therefore, the modelling of the workflow is done using an acyclic graph that is a direct termed DAG. This DAG is the modelling of a workflow in which the node represents the task and the interlink between the tasks is given as the edge [10-12]. Generally, the aware scheduling of energy is designed by researchers that aim on decreasing the cost as well as emphasise green computing. Mostly, the DVFS mechanism is utilized to decrease energy. Although, the energy-aware mechanism emphasises only the reduction of energy and ignores cost as well as fault tolerance. The scientific workflow needs a large count of resources for the processing of big data on the clouds. Further, the real-time services of the cloud require different capacities of computation that lead to the growth of transient failure. Furthermore, the growth in the failure and complexities affect the management of resources that lead to problems in QoS, specifically the requirement of reliability.

Furthermore, recent development in AI(Artificial Intelligence), Social Networking, Big Data ML (Machine Learning) and cloud-HPC leads to additional demand for a resource and computational power which further increases the model failure. Thus there is a requirement for a reliable FTM(Fault tolerant Management System) that can guarantee the smooth operation and run of infrastructure. Moreover, to achieve the ideal availability and reliability, FT becomes an eminent characteristic to be considered as it provides model flexibility to function the model optimally in event of system failure. Even though HPC based model is used for solving the HPC-based problems by researchers and scientists, Fault tolerance has a major role in overall system maintenance. In a real-time scenario, a large-scale HPC model in the cloud runs on various Virtual Machines or Nodes, there is a high probability of system failure due to variation in the execution environments, the intensive workload on a given server and the addition or removal of system components. An increase in electronic components and several VM and software complexities make availability and reliability one of the important factors that the HPC cloud should contend with. Hence, these discussed things have made Fault Tolerance one of the challenging and demanding research areas; further to avoid these, certain steps need to be taken that can handle the possible failures and allow the system to function optimally.

[13]-[15].

The scheduling mechanism for fault tolerance is a mechanism that is effective to improve the reliability workflow and is later used to back up the requirement that satisfies reliability. The already existing model for fault tolerance is discussed in the next part of the paper. Since the novel concept was interesting before but its complexity in the scientific workflow causes the model to fail for tolerating more than a single failure. Therefore, it is essential to develop and design a model for fault tolerance that is capable of tolerating more than one failure and can enhance the requirement of reliability. Also, the optimization of cost is necessary since the optimal expense indicates the efficiency of the model [16-17].

1.1 Motivation and Contribution of Research work

The Cloud Computing model has attracted HPC (High-Performance Computing) community to execute large applications like scientific workflow as unlikely the traditional cluster system, CC(Cloud Computing) provides the services without any investment in hardware and software facilities. Moreover, the cloud model follows the elasticity approach along with the billing cycle of pay per use, thus it is possible to schedule the task to be cost-effective even if the model turns out to be faulty. The existing scheduling algorithms for fault tolerance utilized either one backup for every primary for the tolerance of a single failure based on a passive scheme of replication, that is not capable of tolerating multiple potential failures or the use of a fixed number of ω backups for every primary to tolerate ω failures for the same period based on an active replication scheme, this can satisfy the requirement of reliability, although it is capable of causing a high redundancy and high expense. Furthermore, the contribution towards this research work is emphasised using the below points:

1. In this paper, we develop and design a mechanism for fault tolerance that satisfies the requirements of reliability. However, the mechanism proposed is termed an optimal

strategy of duplication along with optimization of cost and fault tolerance (CEFT-HPC-CLOUD).

2. CEFT-HPC-CLOUD has an approach that is iterative for the process of selecting the virtual machine, as well as duplicates available, have redundancy that is minimum at multi-level, also CEFT-HPC-CLOUD model adopts the scalability feature.
3. CEFT-HPC-CLOUD is calculated by considering the parameter of cost. Generally, the failure of the virtual machine needs more resources for coping with the failure and therefore, there exists a sudden increase in the cost.
4. Further, the scientific workflow has been considered in proving the efficiency of the model. For the evaluation of four instances, are developed that consist of a certain count of virtual machines.
5. A comparative analysis is performed and the methodology that is CEFT-HPC-CLOUD has proven to be more efficient than the model that exists.

The proposed research has been organized in a specific way such that the first section begins with the background of computation and the requirement for the phenomena of cloud computing. Further, there is a discussion about the requirements of the scheduling mechanism, the section ends with the contribution and motivation of the proposed research. The second section emphasises the review of the already existing different protocols as well as their limitations. The third section consists of the design and development of the CEFT-HPC-CLOUD with the mathematical formulation. The fourth section has the evaluation of the CEFT-HPC-CLOUD under the consideration of different instances.

2 Related Work

The scientific scheduling workflow with optimization is assumed as an important topic of research for cloud computing as well as various aspects that are explored that include number of workflows and workloads that differ for various platforms and various mechanisms of scheduling. In case the objectives of optimization remain as makespan, consumption of energy, expense as well as multi-objective or reliability. Therefore, this section emphasises various kinds of research concerning fault tolerance with the optimization of cost. In the paper [18], an online algorithm for scheduling was introduced as a robust mechanism against information that was missing. In case there was a potential failure of resources, the scheduling of workflow has higher complexity. Furthermore, a mechanism for failure awareness was developed using a prediction model based on the Markov chain for the availability of resources [19]. Although this model was largely dependent, a different model that is dependent was developed in [20] in which the strategy of replication was adopted as well as there was an extra scheme in case there existed further failures. This resulted in high penalties for performance. Therefore, in [21], WQR was introduced which is the replication along with work queue while considering that the cloud resources are elastic. Generally, fault tolerance is performed using two distinct approaches that include a replication that is passive [22] as well as a replication that is active [23]. However, the strategy of fault tolerance is also considered for improving reliability that in turn reducing the expense [24-27]. The replication that is passive addresses the backup replica that is executed using the virtual machine when there exists a primary failure. Although, it is tough for passive replication to be adopted by only the use of backup [28-29]. The expense of passive replication is more as well its process results in redundancy that is unpredictable. Considering active replication, there are multiple replications for every virtual machine for which the task is successfully executed. In the paper [23], the algorithm called MaxRe is introduced to meet the requirement of reliability that decrease the expense and the redundancy. In this, the requirements of reliability for the whole task are similar as well as every task has requirements that are the square root of n mathematically to explain the requirement of reliability in which n is assumed to be the workflow. In paper [24], the mechanism for optimal resources has been designed that assures the requirement of reliability that reduces these requirements in comparison to the prior mechanisms. Although, it involves high expense for the reduction of the requirements reliability as well as the redundancy. Also, the paper [24] developed an approach for EERM where the performance is on an approach based on iteration for future implementations relating to the replication that is based on quantity. This

particular model has a low redundancy of cost considering lesser workflow. Although, this fails drastically when huge workflows are considered in making it a model that is low in efficiency.

The methods that are proactive for fault tolerance are also used for the prediction of faults for the replacement of potential sections which would face faults, therefore the event of fault would be prevented. In paper [30], a fuzzy method of task distribution system would be introduced in which the right distribution of load, as well as the requests of the user tasks among resources that are available, reassure tolerance of fault by implementing the detection of fault as well as discrepancies that decrease the systems events of faults. The results that are obtained by the experiments performed indicate that in comparison to the other algorithms of load balance, the algorithms that are evolutionary and have multi objectives, as well as utilization of energy, are not considered. Although, the event of faults is decreased considerably. In the paper [31], the modified algorithm of NSGA-III in which intelligent techniques of fault tolerance are used for the system utilization to increase. However, multiagent devices are used for this method proving to have a high contribution considering various aspects of engineering that include provisions for service transportation, cloud computing networks that are intelligent as well as parallel devices for computation that are complicated. Hence, this model gives a relative assessment of stability without the evaluation of the distribution of faults considering the events of fault. The hybrid methods of fault tolerance are used as reactive and proactive methods of fault tolerance. Paper [32] has developed a hybrid technique for the fault tolerance of a network as well as hardware in which the concept of a square multiplication matrix is applied. Since the data storage for cloud computing in the network happens remotely and most of the faults take place due to failure of the system as well as network congestion. Considering this method, faults are predicted as well as applied using a technique of migration for the reduction of loss of data.

3 Proposed Methodology

Cloud Computing offers a novel flexible solution with the optimal capacity for HPC for running large scientific applications with the provision of a huge number of VM (Virtual Machines). Moreover, fault tolerance provides an HPC model on a cloud with various nodes to execute the large application. Generally, there exist high failure rates for cloud computing because of the large count of components as well as servers that have workloads. The result of these failures has constrained the availability of the virtual machine. Although, this is resolved using the optimal strategy for fault tolerance. Therefore, in this section, the mathematical model is introduced for the CEFT-HPC-CLOUD method that focuses on giving an optimal requirement of reliability as well as reducing the further cost. This method has different sub-sections that are discussed and explained below.

3.1 preliminary

Consider E to represent the set of VMs for which $E = \{E_0, E_1, \dots, E_l\}$. The configuration of these virtual machines takes into account various parameters that include memory, expense, final count of virtual machines as well as a final count of memory. Further, the instance of the virtual machine is developed by denoting it as U as $U = \{U_0, U_1, \dots, U_g\}$ for which U_g indicated the instances of the virtual machine along with configurations. The proposed technology of OPT_FAULT_MEC is a machine that is designed based on parallel processing.

3.2 Workflow Modelling

Considering a scientific model of workflow denoted as Y, also given as $Y = (V, E)$ in which the V denotes the set of tasks and E denoted the dependencies. The scientific workflow models, as well as complex, have the existence of dependencies. Further, a few parameters are added that require to be initialized considering the task they are $\varphi(u_w), \phi(u_w), \chi(u_w) and\ \psi(u_w)$. Consider the tasks from u are performed for resources v, then the cost of the workflow model is formulated as given below

$$\zeta(u_w, u_x) = \begin{cases} (wt_{\mathrm{vw}})\left(\mathfrak{b}_{\mathbb{j},\mathbb{k}}\right)^{-1} & when\ U_k \neq U_l \\ 0 & when\ U_k = U_l \end{cases} \quad (1)$$

The bandwidth $\mathfrak{b}$ is also given as,

$$\varsigma_{k,l} = op(\varsigma(\rho(U_k)), \varsigma(\rho(U_l))) \quad (2)$$

3.3 Task Design considering Fault Tolerance

Consider a single task taken from a group of tasks such that they have the same period for execution as well as parametric internal of the tasks are formulated as stated below.

$$P_o = (((m_o E_o)(E_s h_o)^{-1}) - 1)^{-1/2} \quad (3)$$

In the above equation (3), the ideal level of fault tolerance is denoted as m_o and the operation frequency is denoted as h_o. In this circumstance it is assumed that the situation is ideal, meaning there is no occurrence of a fault. This is given as

$$\varpi_m = E_o + P_o h_o d_u \quad (4)$$

Consider the task v_w is given to a resource that is denoted as U_k and is assumed to follow the ideal circumstance that is mentioned above, therefore this also states that the execution will be ideal such that the equation for this calculation is given below where $P(v_w, U_k)$ which has the entire tasks with the overhead denoted as P_k

$$\alpha_{best}(V_w, U_k) = P(V_w, U_k).P_k + (\varpi(v_w))(um((U_k)))^{-1} \quad (5)$$

The periodic length is calculated using the below equation

$$\beta(V_w, U_k) = ul(\varpi(U_k))(\varpi(v_w))^{-1}.(P(V_w, U_k) + 1)^{-1} \quad (6)$$

After the design of the ideal circumstance, that is the best-case scenario, it is also important to design the worst-case scenario in which the occurrence and possibility of an error are at maximum with the tasks as well as the VM that was described earlier. Furthermore, the total overhead is denoted as $P(V_w, U_k)$ and the overhead fault tolerance is denoted as $\beta(V_w, U_k)$.

$$DS_{\mathbb{x}}(V_w, U_k) = \frac{1}{V_w}.(ul(\varpi(U_k)))^{-1} + 2O(V_w, U_k).P_k + segment(V_w, U_k).H_{maximum} \quad (7)$$

Therefore, for the optimization of the worst-case scenario, the optimized interval is evaluated as given below

$$P_{opt}(V_w, U_k) = ((H_{maximum}(Q_k)^{-1}(refload(V_w)).(ul(\varpi(U - 1)^{-1/2} \quad (8)$$

However, for fault tolerance, the probability of an error is calculated that is used in the definition of the reliability of the task,

$$\daleth(V_w, U_k, H) = H!.\frac{1}{d^{-k_k DS_{\mathbb{x}}(V_w, U_k)}(\sigma_{\mathbb{j}}.DS_{\mathbb{x}}(V_w, U_k)^F)} \quad (9)$$

Whereas, the reliability of the task is given as the state of probability in which the execution of the task even if there is a failure. Therefore, the possibility of a successful execution is $\daleth_{success}(H, U_k) = d^{-k_k DS_{\mathbb{x}}(V_w, U_k)}$. Hence, the reliability of the task group is given as

$$Q(V_w, U_k) = \sum_{k=0}^{H_{maximum}} r(V_w, U_k, H_{maximum}).\daleth_{success}(H, U_k) \quad (10)$$

3.4 Requirement of QoS Reliability

Generally, there are two different kinds of failures relating to the system, namely, permanent and temporary. However, this research considers the second kind of failure. Therefore, we develop the event reliability for the given period t.

$$\aleph(t) = e^{-tu} \quad (11)$$

Considering the above equation, the frequent failures that occur are given by u, the fail that occurs constantly is denoted as u_k. Furthermore, the reliability of n_i for the time interval u_k is given as follows:

$R(n_i, u_k) = e^{-u_k w_g k}$	(12)

Furthermore, the occurrence of failure without the proposed technique OPT_FAULT_MEC is given as:

$1 - R(n_i, u_k) = 1 - e^{-u_k w_g k}$	(13)

Further, assuming that every task has a particular count of replicants, therefore, $numb_g(numb_g \leq \lfloor S \rfloor)$ as N_g. However, the set of replicants n_g is defined as $(n_g^1, n_g^2, \ldots, n_g^{numb_g})$ for which n_g^1 is considered absolute and the other variables are considered replicants. The final count of replicants that are developed for the workflow is as given below:

$numb_{rep(G)} = \sum_{g=1}^{\lvert n \rvert} numb_i$	(14)

When n_g the replication process is complete, there are zero occurrences of failure as well as reliability that is as follows:

$R(n_g)=1-\sum_x^{numb_g}(1 - R(n_g^x, u_{r(n_g^x)}))$	(15)

Consider the equation (15), $u_{r(n_g^x)}$ denotes n_g^x

$R(G) = \sum_{n_{g\epsilon N}} R(n_g)$	(16)

3.5 Optimal Cost Modelling

Fault tolerance systems that are efficient and decrease cost execution along with the reliability that is defined as, considering the workflow development with a required virtual machine that was discussed in the previous part of this section. The challenge is assigning the replicants with their respective virtual machines for every specific task. However, the cost of execution is decreased and the fault tolerance is assured by the use of the requirement of reliability that is developed in the previous sub-section. Therefore, the optimal replicants also called backups, which are assigned with the designated virtual machines are given below

$expense(g) = \sum_{n_i \epsilon N} expense(n_i)$	(17)

The reliability of the system is subjective to the requirement, given as follows:

$R(G) = \sum_{n_i \epsilon N} (R(n_g) \leq R_{qp})$	(18)

3.6 Single task requirement for reliability QoS

For every single task, the requirement of reliability is formulated as well as met. Initially, the requirement for reliability is computed for every single task by taking into consideration the reliability that is absolute given below

$R_{qp}(n_{rp(i)})=(1 - h + \lfloor N \rfloor)(R_{qp}(G).\frac{1}{\sum_{x=1}^{h-1} R(n_{rp(x)})})^{-1/2}$	(19)

Considering the equation (19), the ith task is denoted by $n_{rp(i)}$. Furthermore, the requirements' reliability is optimized using the points that are given. During the computation of the requirements of reliability for single tasks, we assume that $(\lfloor O \rfloor)(R_{qp}(G))^{-1/2}$ indicates the value of the upper bound for the requirements of reliability for a given task N_g which is given below:

$R_{TQP}(N_g)=(\lfloor O \rfloor)(R_{qp}(G))^{-1/2}$	(20)

However, we consider that the assignment of the task $n_{rp(i)}$ is for task 1 after which the group of tasks is given for the tasks that are not assigned such that $(n_{rp(1)}, n_{rp(2)}, \ldots, n_{rp(i-1)})$ and the tasks assigned are given as $(n_{rp(1)}, n_{rp(2)}, \ldots, n_{rp(i-1)})$. Furthermore, the proposed model OPT_FAULT_MEC considers that every task for the model workflow is assigned to the virtual machine with the parametric value of reliability, which is formulated in equation 19

assuring reliability. Therefore, the complete requirement of reliability is evaluated as follows:

$$R_{qp}(G) = \left(\sum_{y=h+1}^{[N]} R_{TQP} n_{rp(y)}\right) \sum_{x=1}^{h-1} \left(R\left(n_{rp(x)}\right)\right)\left(R(\right. \quad (21)$$

For a single task, the computation is performed as given in the equation below. However, the reliability constraints are met using an iterative approach for the replication and the virtual machine has a makespan that is least.

$$\left(R\left(n_{rp(i)}\right)\right) = R_{qp}(G) \cdot \frac{1}{\left(\sum_{y=h+1}^{[N]} R_{TQP} n_{rp(y)}\right) \sum_{x=1}^{h-1} \left(R\left(n_{rp(x)}\right)\right)\left(R(r\right.} \quad (22)$$

In the next phase of this paper, we develop the proposed algorithm that decreases the cost as well as provides fault tolerance that is efficient considering the workflow.

3.7 Multilevel CEFT-HPC-CLOUD Algorithm

The focus of the CEFT-HPC-CLOUD algorithm is the offloading of the requirement reliability in this sub-section for every single individual task. Furthermore, the algorithm proposed, namely the CEFT-HPC-CLOUD algorithm reduces the cost of execution by choosing replicants and virtual machines that are optimal. In this case, the virtual machines that are optimal have a period for execution that is efficient. Although, it has been observed that some redundancy still exists. It has also been seen that some of the replicants could be discarded. CEFT-HPC-Cloud is a multilevel and scalable fault-tolerant approach that makes the novelty of the model. We explain the entire working of the CEFT-HPC-CLOUD algorithm below

	CEFT-HPC-CLOUD Algorithm
Step 1	Initializing
Step 2	Input: Reliability requirement is DAG data that includes execution time, virtual machine set, nodes set and time of communication. Output: Value of reliability, expense, length of schedule
Step 3	Reducing the order of tasks
Step 4	$For\ (j = 1; j \leq \|0\|; j++)$ do computation $R_{qp(j)}(N_{qp(j)})$ $Z_{qp(j)}\ is\ equal\ to\ zero$ $A_{qp(j)}\ is\ equal\ to\ zero$
Step 5	A backup list is defined $rep(n_{qp(j)})$ and stored in $n_{qp(j)}$
Step 6	$For(k = 1; k \leq \|U\|; k++)$ do computation $R_{qp(j)}(N_{qp(j)}, u_k)$ computation op_end_time
Step 7	End for;
Step 8	While $\left(R\left(n_{rp(j)} < R_{rp}\left(n_{rp(j)}\right)\right)\right)$ do select the replica $n^x_{rp(j)}$ and the virtual machine $u_{r(n^x_{rp(j)})}$ with execution period as optimal $w_{seq(j),r(n^w_{rf(i)}}$ $numb_{rp(j)} ++$
Step 9	$n^x_{seq(j)}$ is placed in the replication list rep_list in decreasing order. This discards the prior allocations $numb_{rp(i)}\ is\ equal\ to\ zero$ $R(n_{rp(j)})\ is\ equal\ to\ zero$
Step 10	While $R(n_{rp(j)}) < r_{rp}(n_{seq(j)})$ do select the replicant $n^x_{rp(j)}\ and\ u_{r(n^x_{rp(j)})}$ for the list given. Also, discard the replicant $n^x_{seq(j)}$ from the list given and increased $numb_{seq(j)}$
Step 11	Computation $End_{time}\left(n^x_{rp(j)}\right) = op\_end\_time(m^w_{rf(i)}, t_{r(m^w_{rf(i)})})$

Step 12	Compute R($n_{rp(j)}$)
Step 13	End while; (step 8)
Step 14	End For;
Step 15	$compute\ \ cost, makespan, reliability$

In the CEFT-HPC-CLOUD algorithm, the task scheduling is in order. The optimization order is computed by the equation given below in which v_w is a time of execution for a task n_g.

$$\delta_u(n_g) = w_g + \underset{n_g \in rcc(n_g)}{maximum}\{c_{g,j} + \delta_u(m_i)\} \quad (23)$$

Furthermore, Multilevel CEFT-HPC-CLOUD is the proposed algorithm that replicates the optimal virtual machine and these virtual machines are stocked and sorted according to an optimal order. After the optimal virtual machines are sorted out, the algorithm is cleared of its allocations. Furthermore, the replicants are assigned. The algorithm selects only those virtual machines that have an order that is higher for reliability and further replicates, execution of cost.

4 Performance Evaluation

Cloud Computing and HPC together have emerged as highly efficient modelling in a real-time scenario as both integrated hold the characteristics of being cost-effective, pay-as-you-use service, high-performance computing, and remotely accessible. Moreover, CEFT-HPC-CLOUD is simulated considering the Java Coding Environment on windows 10 packed with 2TB of hard disk and 4GB of Nvidia graphics. Furthermore, the CEFT-HPC-CLOUD model is evaluated on a scientific workflow dataset; this dataset contains different computational and data characteristics. Dataset Details are given in the next section.

4.1 Dataset Details

Four distinctive datasets are used for model evaluation; the first workflow is the Montage workflow which is used for astronomy that aims for designing the desired mosaic sky based on the given input images. Moreover, it requires less CPU capacity and frequent Input/output operations. The second workflow is the SIPHT workflow which is used to automate the RNAs' search of encoding genes for the bacterial replicas' through bioinformatics. The third workflow is LIGO; LIGO workflow generates and analyses the gravitational waveforms from data gathered while coalescing CBS (Compact Binary System). The fourth Workflow is cybershake workflow; it is an earthquake application that is designed for characterizing earthquake hazards it comprises the large requirements and a large dataset for CPU and memory respectively.

4.2 Structuring Instance and cost comparison on considering cost

Model evaluation is carried out considering the different instances; moreover, these instances are structured for model evaluation each instance comprises a certain number of hosts, VM, and particular workflow.

4.2.1 Instance A

Moreover, Instance A is designed with VM number 30 with cybershake_30, Montage_25, SIPHT_30, and Inspiral_30. Moreover, considering the designed instance with a certain number of hosts and VM, In the case of cybershake_30 workflows, the existing model costs 21265.98 whereas the CEFT-HPC-CLOUD model costs 20260.8. Similarly considering other workflows with Montage_25 and SIPHT_30, execution costs is 810 and 18335.66 whereas the CEFT-HPC-CLOUD model requires execution cost of 559 and 12223 respectively. For SIPHT_30, the existing model requires an execution cost of 21953.57 whereas the CEFT-HPC-CLOUD model requires an execution cost of 14674.14.

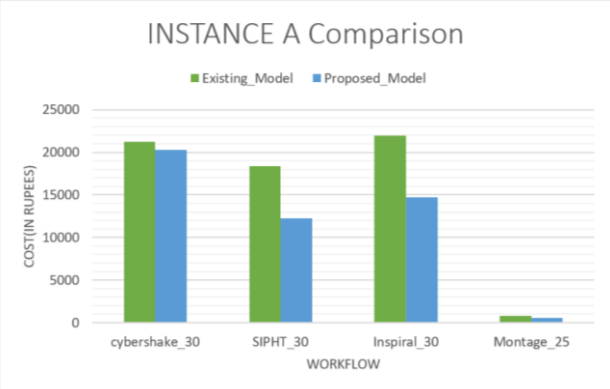

Figure 1 Instance A evaluation

4.3 Instance B

Instance B is designed with VM number 60 with cybershake_50, Inspiral_50, Sipht_60 and Montage_50; Moreover, the execution cost of cybershake_50 and Inspiral_50 through the existing model is 23596 and 39015 respectively whereas the execution cost through CEFT-HPC-CLOUD model is 22042 and 26075.38 respectively. Similarly considering instance B along with workflows Sipht_60 and montage_50 execution cost by the existing model is 38562.73 and 1791.05 whereas execution cost by the CEFT-HPC-CLOUD model is 25724.98 and 1230.7 respectively.

Figure 2 Instance B Evaluation

4.4 Instance C

Instance C is designed with number VM number as 90 with Inspiral_100, Sipht_100, cybershake_100, Montage_100; Furthermore, considering the Instance C with Inspiral_100 and Sipht_100, execution cost by the existing model is 69769.53 and 57437.64 respectively whereas execution cost by the model is 46640.98 and 38317.21 respectively. Similarly, for cybershake_100 and montage_100 workflows, the execution cost by the existing model is 28844.28 and 3773.54 respectively whereas the execution cost by the CEFT-HPC-CLOUD model is 26265.74 and 2584.35 respectively.

Figure 3 Instance C cost evaluation with different workflow

4.5 Instance D

Instance D is designed with number VM number as 90 with Inspiral_1000, Sipht_1000, cybershake_1000, Montage_1000; Furthermore, considering the instance D with Inspiral_1000 and Sipht_1000, the execution cost by the existing model is 755410.37 and 573869.37 whereas execution cot by CEFT-HPC-CLOUD model is 504910.93 and 382799.81 respectively. Similarly, for cybershake_1000 and Montage_1000, execution cost by the existing model is 195619.83 and 39637.35 whereas execution cost by the CEFT-HPC-CLOUD model is 169295.1 and 27101.36 respectively.

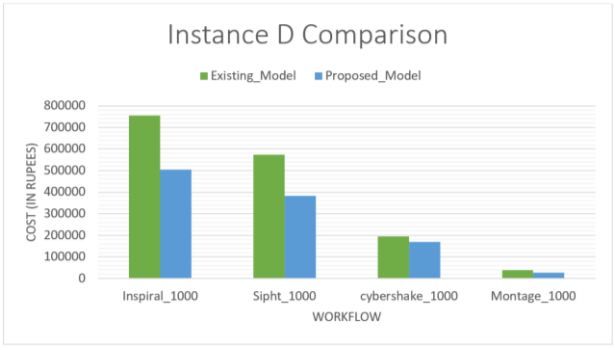

Figure 4 Instance D cost evaluation with different workflow

4.6 Comparative Analysis

This section evaluates the comparative analysis that shows the improvisation of an existing model over the CEFT-HPC-CLOUD model. Moreover, Figure 5 shows the improvisation of the CEFT-HPC-CLOUD model over the existing model considering the designed instance as stated earlier. For cybershake_30 and Montage_25 CEFT-HPC-CLOUD model observes 4.72% and 30.97% respectively; similarly, for SIPHT_30 and Inspiral_30, the CEFT-HPC-CLOUD

model observes improvisation of 33.15 % and 33.28% respectively.

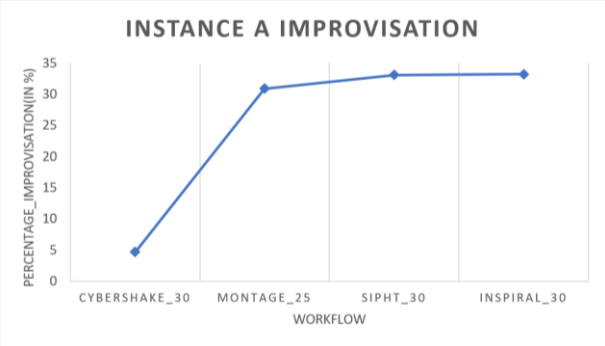

Figure 5 Improvisation over the existing model considering Instance A

Figure 6 shows the Improvisation of the CEFT-HPC-CLOUD model over the existing model; In the Case of Cybershake_50 and Inspiral_50, the CEFT-HPC-CLOUD model observes 6.58 % and 31.28% respectively. Similarly, for SIPHT_60 and Montage_50, the CEFT-HPC-CLOUD model observes improvisation of 33.16% and 33.29%

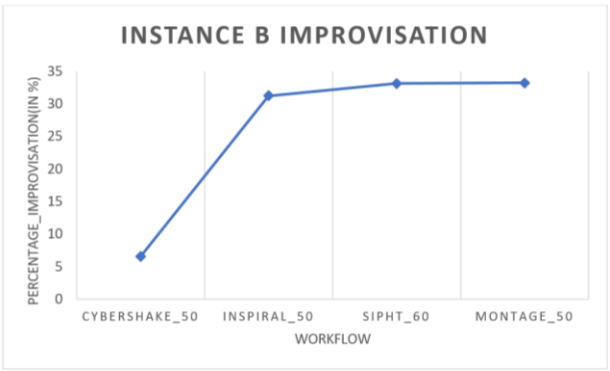

Figure 6 Improvisation over the existing model considering Instance B

Figure 7 shows the Improvisation of the CEFT-HPC-CLOUD model over the existing model; In the case of Inspiral_100 and SIPHT_100, the CEFT-HPC-CLOUD model observes 8.92 and 31.51 respectively. Similarly, Cybershake_100 and Montage_100 observe 33.14 and 33.28 respectively.

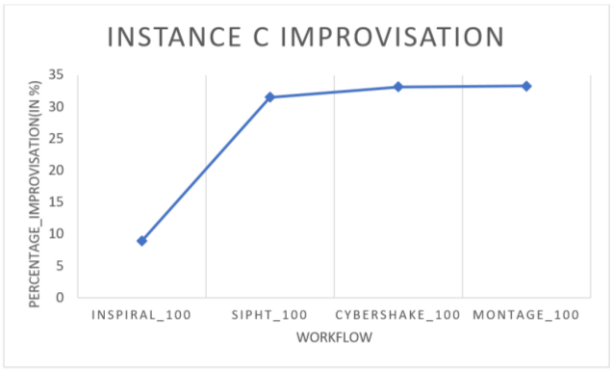

Figure 7 Improvisation over the existing model considering Instance C

Figure 8 shows the improvisation of the CEFT-HPC-CLOUD model over the existing one considering the instance D; In the case of Inspiral_1000 and SIPHT_1000, the CEFT-HPC-CLOUD model observes improvisation of 13.41% and 31.62 % respectively. Similarly, in the case of Cybershake_1000 and Montage_1000, the CEFT-HPC-CLOUD model observes improvisation of 33.16% and 33.29 % respectively.

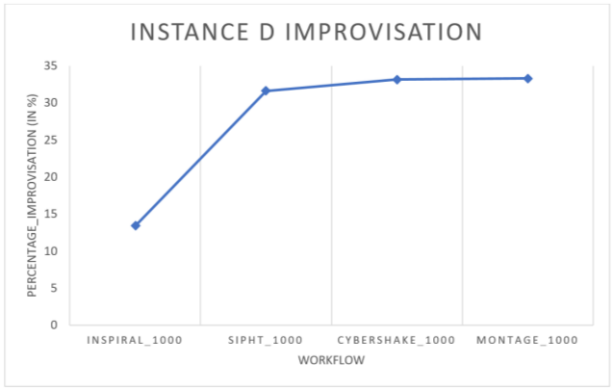

Figure 8 Improvisation over the existing model considering Instance D

Conclusion

Integration of cloud computing and HPC is efficient, high-performance computing which led to universal acceptance for large-scale based application; Moreover, HPC cloud is vulnerable due to handling large computation application and possesses a huge number of failure, these failure occurs due to various reason and it amounts to VM (Virtual Machine) unavailability and leads to higher cost. These issues are

resolved by developing a fault-tolerant approach. This research work develops the CEFT-HPC-CLOUD mechanism, which tends to minimise the cost along with a fault tolerance approach; the CEFT-HPC-CLOUD algorithm optimally schedules the VM and selects the duplicates with minimum scheduling length, which reduces the cost in terms of fault tolerance.

Model evaluation is carried out considering the four different workflow cybershake, Montage, SIPHT and Inspiral with different variants; critical evaluation is carried out through designing four instances that comprise certain host and VM on four different workflows. Moreover, comparative analysis suggests that for instance A, Instance B, Instance C and Instance D proposed model observes improvisation up to 33%. CEFT-HPC-CLOUD aim to provide a cost-effective fault tolerance model and outperforms the existing model, however, failure occurrence happens due to various, hence other factors should be considered for future research directions.

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. Sharavana. K, Asghar Pasha, and Dr. Josephin Premkumar K, Approach for Deploying the Hybrid Cloud in Diverse Open Source Tools, IOSR Journal of Computer Engineering (IOSR-JCE),e- ISSN: 2278-0661, p-ISSN: 2278-8727, 20(3)(2018), 25-34.
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. M. A. Mukwevho and T. Celik, "Toward a Smart Cloud: A Review of Fault-Tolerance Methods in Cloud Systems," in IEEE Transactions on Services Computing, vol. 14, no. 2, pp. 589-605, 1 March-April 2021, doi: 10.1109/TSC.2018.2816644.
5. J. Kadupitiya, V. Jadhao and P. Sharma, "SciSpot: Scientific Computing On Temporally Constrained Cloud Preemptible VMs," in IEEE Transactions on Parallel and Distributed Systems, doi: 10.1109/TPDS.2022.3157272.
6. S. Zhang et al., "Practical Adoption of Cloud Computing in Power Systems—Drivers, Challenges, Guidance, and Real-World Use Cases," in IEEE Transactions on Smart Grid, vol. 13, no. 3, pp. 2390-2411, May 2022, doi: 10.1109/TSG.2022.3148978.
7. A. C. Zhou, J. Lao, Z. Ke, Y. Wang and R. Mao, "FarSpot: Optimizing Monetary Cost for HPC Applications in the Cloud Spot Market," in IEEE Transactions on Parallel and Distributed Systems, doi: 10.1109/TPDS.2021.3134644.
8. M. A. Shahid, N. Islam, M. M. Alam, M. M. Su’ud and S. Musa, "A Comprehensive Study of Load Balancing Approaches in the Cloud Computing Environment and a Novel Fault Tolerance Approach," in IEEE Access, vol. 8, pp. 130500-130526, 2020, doi: 10.1109/ACCESS.2020.3009184.
9. F. Zahid, A. Taherkordi, E. G. Gran, T. Skeie and B. D. Johnsen, "A Self-Adaptive Network for HPC Clouds: Architecture, Framework, and Implementation," in IEEE Transactions on Parallel and Distributed Systems, vol. 29, no. 12, pp. 2658-2671, 1 Dec. 2018, doi: 10.1109/TPDS.2018.2842224.
10. Z. Chen, L. Yang, Y. Huang, X. Chen, X. Zheng and C. Rong, "PSO-GA-Based Resource Allocation Strategy for Cloud-Based Software Services With Workload-Time Windows," in IEEE Access, vol. 8, pp. 151500-151510, 2020, doi: 10.1109/ACCESS.2020.3017643.
11. Pretto, G.R., Dalmazo, B.L., Marques, J.A. *et al.* Janus: a framework to boost HPC applications in the cloud based on SDN path provisioning. *Cluster Comput* **25,** 947–964 (2022). https://doi.org/10.1007/s10586-021-03470-6
12. Patricia González, Roberto R. Osorio, Xoan C. Pardo, Julio R. Banga, Ramón Doallo,
An efficient ant colony optimization framework for HPC environments,Applied Soft Computing, Volume 114, 2022, 108058, ISSN 1568-4946.
13. Yin, F., Shi, F. A Comparative Survey of Big Data Computing and HPC: From a Parallel Programming Model to a Cluster Architecture. *Int J Parallel Prog* **50,** 27–64

(2022). https://doi.org/10.1007/s10766-021-00717-y.

14. 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.
15. 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.
16. 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.
17. 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.
18. 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.
19. G. Yao, X. Li, Q. Ren and R. Ruiz, "Failure-aware Elastic Cloud Workflow Scheduling," in IEEE Transactions on Services Computing, 2022, doi: 10.1109/TSC.2022.3188414.
20. J. Schneider, “Grid workflow scheduling based on incomplete information,” Ph.D. dissertation, Berlin Institute of Technology, 2010.
21. Z. Yu, C. Wang, and W. Shi, “Failure-aware workflow scheduling in cluster environments,” Cluster Computing, vol. 13, no. 4, pp. 421–434, 2010.
22. 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.
23. 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.
24. Q. Zheng, B. Veeravalli, and C.-K. Tham, “On the design of faulttolerant 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.
25. 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.
26. 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.
27. 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.
28. 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.
29. 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.
30. 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.
31. F. Pop and M. Potop-Butucaru, “Adaptive resource management and scheduling for cloud computing,” Lecture Notes in Computer Science, vol. 8907, 2014.
32. 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.

33. 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.
34. Sharavana .K L. Rahul Resources, International Journal of Advanced Research in Computer and Communication, IJARCCE, Volume 3, issue 4-2014/4