Adaptive Workflow Scheduling to Increase Fault Tolerance in Cloud Computing

Abdolreza Pirhoseinlo[1], Nafiseh Osati Eraghi[2*], Javad Akbari Torkestani[3]
1- Department of Computer Engineering, Arak Branch, Islamic Azad University, Arak, Iran.
Email: a.pirhoseinlo@gmail.com
2- Department of Computer Engineering, Arak Branch, Islamic Azad University, Arak, Iran.
Email: n-osati@iau-arak.ac.ir (Corresponding author)
3- Department of Computer Engineering, Arak Branch, Islamic Azad University, Arak, Iran.
Email: j-akbari@iau-arak.ac.ir

Received: September 2020 Revised: January 2021 Accepted: March 2021

ABSTRACT:
Cloud computing in the field of high-performance distributed computing has emerged as a new development in which the demand for access to resources via the Internet is presented in distributed servers that dynamically scale are acceptable. One of the important research issues that must be considered to achieve efficient performance is fault tolerance. Fault tolerance is a way to find faults and failures in a system. Predicting and reducing errors play an important role in increasing the performance and popularity of cloud computing. In this study, an adaptive workflow scheduling approach is presented to increase fault tolerance in cloud computing. The present approach calculates the probability of failure for each resource according to the execution time of tasks on the resources. In the present method, a deadline is set for each of the tasks. If the task is not completed within the specified time, the probability of failure in the source increases and subsequent tasks are not sent to the desired source. The simulation results of the proposed method show that the proposed idea can work well on workflows and improve service quality factors.

KEYWORDS: Cloud Computing, Fault Tolerance, Scheduling.

1. INTRODUCTION

The advent of cloud computing is considered to be the biggest change in information technology. This change has attracted the attention of everyone, from people in the community to large corporations. Today, the popularity of cloud computing is so widely accepted that organizations are moving their traditional information processing systems to cloud services to store large amounts of data [1], [2].

Cloud computing has emerged as a demand-based computing services model for use by small-scale users and large-scale scientific and commercial applications. This model is defined as a model for accessing a network with a shared set of configurable computing resources (e.g., networks, servers, storage space, applications, and services) that can be quickly and efficiently defined. Minimize managerial effort or service provider interaction, access to demand, resource independence, fast flexibility and always availability are the basic features of cloud computing [3], [4].

Due to the popularity of cloud computing among users and service providers, this environment has been very popular in the last decade. The presence of users and service providers from different places and with heterogeneous systems in this medium have caused many challenges. In cloud computing, fault tolerance is an important problem and resource failure time is one of the metrics that affect performance, throughput, response time and system and network performance [5].

Load balance fault tolerance is one of the main challenges in cloud computing that is required to distribute network load evenly across all nodes. This load is the amount of work that a computing system does, which can be classified as network load, storage capacity, memory capacity, and CPU load [6].

Error tolerance is a way to find faults and failures in a system. If an error occurs or there is a hardware failure or software defect, the system should also work properly. Malfunctions must be controlled in a dynamic way for reliable cloud computing. It will also provide availability and reliability against hardware and software failures of the system in the organization [7], [8].

Predicting and reducing errors has two types of techniques: active and passive. Active fault tolerance

Paper type: Research paper
DOI: https://doi.org/10.52547/mjee.15.3.25
How to cite this paper: A. R. Pirhoseinlo, N. Osati Eraghi and J. Akbari Torkestani, "**Adaptive Workflow Scheduling to Increase Fault Tolerance in Cloud Computing**", *Majlesi Journal of Electrical Engineering*, Vol. 15, No. 3, pp. 25-33, 2021.

policy, avoiding fault recovery by anticipating and actively replacing a suspicious component to detect it before a fault occurs. Passive fault tolerance reduces the impact of failures on executable programs when effective failures occur [9].

Therefore, in this study, an adaptive workflow scheduling approach is proposed to increase fault tolerance in cloud computing. The present approach calculates the probability of failure for each resource according to the execution time of tasks on the resources. In fact, in the present method, a time limit is set for each of the tasks. If the task is not completed within the specified time limit, the probability of failure in the source increases and the next tasks are not sent to the desired source.

In the proposed method, the input flow is timed through a resource management unit. At this stage, according to the characteristics of the resources, the tasks are assigned to the nearest source in terms of service quality factors to be executed. The scheduling plan is then provided to a task management unit to set a time limit for each task, as mentioned, according to the number of instructions for each task, and to review the time frame for performing previous tasks in the resources. Slowly If a resource task is running for more than the specified time, the probability of resource failure increases.

Thus, the task management unit calculates a failure probability for each resource. Based on the probability of failure, the tasks are then sent to the resources, which are virtual machines, to complete the continuation of the task execution process. The main innovation in the present study is the addition of a task management unit which, according to the number of instructions for each task and the characteristics of the resources, assigns an adaptive time limit to each task to be performed in each resource. Then, according to the history of performing tasks in the resources, the probability of failure for each resource is calculated.

2. LITERATURE REVIEW

Fault tolerance provides complete and permanent operation even when incomplete components are present. It is the continuation of work satisfactorily in the face of obstacles, art and science that creates the computational system. The fault tolerance system does not allow one or more types of errors to occur, including momentary, definitive or permanent hardware or software failures, design errors, operational errors, or damage caused by the program itself or physical damage. In a real-time cloud application, these operations are performed remotely in the computational node and the probability of error occurs [10]. These events emphasize the need for fault tolerance technology to achieve the reliability of real-time computing in the cloud infrastructure.

Abu Hamama et al. have proposed a comprehensive framework that incorporates a number of error tolerance methods to improve the reliability of cloud environments for hosting applications in real time, if reliability is achieved and access conditions are provided. Then the real-time fault tolerance scheduling algorithm is proposed to minimize the number of missing deadlines, and the amount of load imbalance [11].

Yan et al. presented the uncertainty model by estimating task execution time and a contrast-task assignment mechanism that strategically uses two error-tolerant task scheduling models, while considering uncertainty. In addition, an overlap mechanism has been used to improve the use of cloud resources. Based on two mechanisms, a dynamic innovation-based scheduling and scheduling algorithm based on Dynamic Innovation (DEFT) is proposed for scheduling real-time tasks in the cloud in which system performance fluctuations must be considered. The purpose of DEFT is to achieve both fault tolerance and resource utilization [12].

Ding et al. presented a flexible common error scheduling algorithm for cloud workflow (FTESW). After analyzing the initial scheduling constraints, backup in cloud systems due to interdependence between tasks in the workflow is presented, a tensile mechanism in the field of fault tolerance is designed to dynamically adjust resources based on resource requests by adopting resource migration technology, FTESW is then proposed to achieve both fault tolerance and full resource utilization for workflow in cloud systems. To evaluate the effectiveness of the proposed FTESW, a series of simulation experiments were performed on both randomly generated workflows and real-world workflows [13].

Talvani et al. have effectively evaluated and processed barriers to gain robustness and reliability in cloud computing. Various error detection methods and architectural models have been proposed to increase cloud fault tolerance. Given this fact, this article focuses on solving the problem of error in cloud computing. This research, in order to identify the work that has been done in this field, shows the basic concept of fault tolerance that has been done by different researchers and different algorithms to solve the problem of fault tolerance in cloud computing [14].

3. METHODOLOGY

As mentioned, in this study, in order to present the error tolerance approach in cloud infrastructure, an approach of workflow distribution and adaptive scheduling based on task execution time in resources has been used, which will be described in detail in the following method.

3.1. Workflow

As cloud end users use host applications, the workload includes tasks that dynamically go to the central scheduler. Each task contains a set of tasks that is a set of $T = \{t_1, t_2, \dots, t_n\}$ with n tasks. Each of the tasks has a time limit or deadline to be performed, which in case of exceeding this deadline, the performance of the task and its related tasks will be adversely affected. In this research, we use the non-distance directional graph (DAG) to show the workflow provided by the customer to cloud systems with $Dead_L$ software deadline. A $DAG\ V = (T, E)$ contains n tasks that are connected through control and data flow.

$$E = \{(t_i, t_j, Data_{ij}, Cont_{ij}) | (t_i, t_j) \in T \times T, i \neq j\} \quad (1)$$

Where, $Data_{ij}$ represents the amount of data to be transferred from t_i to t_j, and $Cont_{ij}$ represents the control relationship, such as the loop and the selected structure, between t_i and t_j. The size of each t_i task is measured by millions of MI_i instructions that affect its execution time. We also use $pred(t_i)$ and $succ(t_i)$ to indicate the previous and next set of tasks of task t_i, which means that $pred(t_i)$ must be completed before starting t_i. An example of a workflow is shown in Fig. 1.

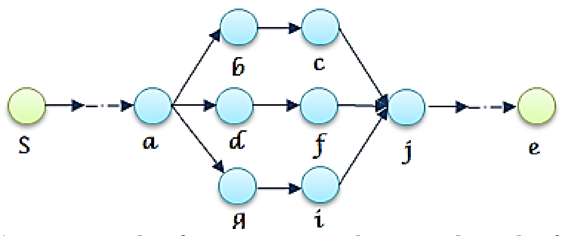


Fig. 1. Example of a non-rotating directional graph of a workflow.

For each task t_i sent in the workflow, two versions are provided as ${t_i}^P$ and ${t_i}^B$ for the initial version and the backup, respectively. They run on different hosts for fault tolerance. $vm({t_i}^P)$ and $vm({t_i}^B)$ are used to represent virtual machines containing ${t_i}^P$ and ${t_i}^B$, while $h({t_i}^P)$ and $h({t_i}^B)$ are hosts, respectively. The task execution time t_i in the vm_{kl} virtual machine is calculated as $et_{kl}(t_i)$ as the ratio of the task size to the processing power of the virtual machine. For example, $vm({t_i}^P) = vmkl, et_{kl}({t_i}^P) = \frac{MI_i}{vp_{kl}}$ means that the initial version of t_i is scheduled in the vm_{kl}, virtual machine hosted on h_k, and has an execution time of $et_{kl}({t_i}^P)$ of $\frac{MI_i}{vp_{kl}}$. In addition, the time used to transfer $Data_{ij}$ can be stated as follows:

$$TTD_{j,X}^{i,X} = \begin{cases} 0 & if \quad h(t_i^X) = h(t_j^X) \\ \frac{Data_{ij}}{BW_{h(t_i^X)h(t_j^X)}} & if \quad h(t_i^X) \neq h(t_j^X) \end{cases} \quad (2)$$

3.2. Fault Model

Tasks will fail and stop running when an error occurs in the processor unit in which they are located. In this study, we focus on host failure. A host fails when all the virtual machines in that host can no longer work and cannot complete the copies of the tasks in the virtual machines. Errors can be temporary or permanent, and it is assumed that the error is in one independent host and affects only one host. The maximum number of hosts that are expected to fail at any given time is assumed to be one. This means that if the original version of a task fails, the backup can always be completed before another host fails. For each task, backup is done after scheduling the initial version. However, it is not necessary to back up all tasks in a workflow after scheduling initial versions of all tasks in the same workflow. It is also assumed that the minimum value required for the average error time (MTTF) is always greater than or equal to the maximum execution time of a virtual machine from the same host.

There is an error detection mechanism such as an error signal and an acceptance test to detect errors. If an error is detected in an initial version, the backup will be executed and new tasks will not be sent to the host that received the error. If the completion of the initial version is successful, the resources will be restored by deleting the backup. Restoring resources is essential so that backup time can be allocated to new tasks.

3.3. Predict Fault

Predicting an error in a host can speed up the completion of tasks and prevent new tasks from being sent to hosts suspected of error. In this module, error prediction is based on the average time of error, which was introduced in the previous section. As mentioned, the minimum value required for the average error time is always greater than or equal to the maximum execution time of a virtual machine from the same host. Given that the virtual machines are heterogeneously distributed in one host, it can be said that the minimum value required for the average error time is always greater than or equal to the maximum execution time in the slowest virtual machine of the same host. By assigning a task to a virtual machine from a host, the average amount of error time is determined based on the number of instructions in each task and the processing power of the virtual machines in the host, based on Equation 3 for each task.

$$MTTF_{kl} = \min \{\frac{vp_{kl}}{MI_i}\} \quad (3)$$

Subject to

$vp_{11} \leq vp_{12} \leq vp_{13}, \ldots$
$MTTF_{kl} \leq Deadline_i$
$MI_i \geq 0$

The scheduler includes a workflow analyzer, a prototype controller, a backup controller, a resource manager, a task manager, and an error forecasting module. When a workflow is entered, the scheduler determines whether it is first accepted in collaboration with the workflow analyst and resource manager. For the accepted workflow, the workflow analyst analyzes the relationship and prioritizes scheduling among tasks. A backup is then made by the backup controller. The fault-tolerant scheduler is then connected by the main controller and the backup controller. If active resources fail to meet the schedule request, the resource manager switches some hosts from sleep to active to increase active resources. In addition, the host reports information about its status, including the status of scheduled tasks, the next access time to each active virtual machine for other tasks, and the resource usage of each active host, directly to the scheduler, and the resource manager and records the status of all hosts in cloud systems. If the initial version of a task is successful, the initial version success information is sent to the resource manager. The resource manager then notifies the virtual machine to which it has sent the backup of the assigned task to cancel the backup. Otherwise, the resource manager will not notify the virtual machine to cancel the backup, and scheduling backups will normally be performed for fault tolerance. In addition, if the host is in low workflow mode and some virtual machines remain idle for a long time, the resource manager decides that some virtual machines should be used to improve resource utilization and be deleted or transferred to other hosts. Since cloud systems resources are shared among all customers and workflows are dynamically sent to cloud systems at runtime, the above process for each stream something is repeated.

Each workflow is given to the task manager as a set of tasks, so that the task manager can create backups and backups of each task and provide their information to the resource manager for allocation to the machine and sends virtual error prediction modules to calculate the average error value. The initial version of each task is assigned to one of the virtual machines in the host, followed by the task manager backing up the task. The task backup is also sent by the resource manager to another active virtual machine on the other host. When the initial version of a task starts working in a virtual machine, time is allotted for that task. If the execution of the initial version of the task in the virtual machine is completed successfully by the time of the average error, the error forecasting module sends the command to cancel the execution of the backup to the task manager. The Task Manager also records the success information of the original version on the target host and cancels the backup. The task manager also sends information about the success and cancellation of the backup to the resource manager to prevent the backup from running on another virtual machine on another host.

If the execution of the initial version of the task in the virtual machine is not completed successfully by the time of the average error, the error forecasting module sends the command to stop the execution of the initial version in the current virtual machine and transfer it to another virtual machine in the same host. Slowly The error prediction module waits until the time limit set for task i expires. If the execution of the initial version of the task in the current host virtual machines is completed by the end of the deadline for task i, the error prediction module sends the command to cancel the backup to the task manager. Otherwise, the error forecasting module sends a backup command to the task manager by imposing a penalty on the present host. Due to the penalty imposed on the current host, new tasks will not be transferred to this host and the current host, despite the fact that there may be active virtual machines in that host, in the queue of hosts that if they make a mistake, they will be placed. Assignment of tasks to these hosts is delayed until the completion of tasks related to the current workflow and the arrival of a new workflow.

4. RESULTS AND DISCUSSION

In order to implement the proposed method, according to the previous chapters, in this dissertation, 20 tasks have been used in the form of a workflow. In order to assign these tasks to run in the cloud, we used 5 hosts with 15 virtual machines. The method is simulated in MATLAB software version 2015. Tasks that reach the cloud environment are workflows in which tasks can communicate with each other. Therefore, due to the relationship between tasks, the discussion of data transfer and priority in execution occurs, which is one of the main discussions in the cloud environment in order to serve the tasks. As mentioned, the workflow includes a number of tasks in line with a specific goal, which, based on the priorities between the tasks, the order of execution of these tasks, has a decisive role in the implementation of the entire workflow. Fig. 2 shows an example of the workflow sent to the cloud environment.

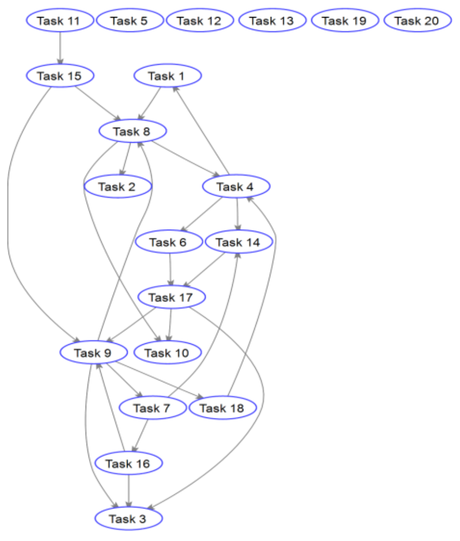


Fig. 2. Workflow.

As shown in Fig. 2, the current workflow has 20 tasks, some of which are related to other tasks. The margins between tasks are directional, indicating the priority and lag of tasks in estimating the overall goal of the workflow.

Due to the fact that the proposed method aims to increase the fault tolerance in the cloud environment, it tries to optimize the main factors in the quality of service, so first prepare and maintain a copy of the received tasks as a backup. Tasks that reach the cloud are identified based on the number of instructions in each task, which is the instruction measurement scale of instructions, million instructions per second (MIPS). In addition, each resource has processing power that is released from service providers. Therefore, the task execution time in the virtual machine can be calculated as the ratio of the task size to the processing power of the virtual machine. In addition, the time used to transfer data while performing tasks between two virtual machines can be calculated using Equation 2. Therefore, Table 1 shows the specifications of the original and backup versions, the time required to perform the task and the time required to perform the previous task, if any, the time required to transfer data between virtual machines based on the amount of data.

Table 1. Real-time Execution of Tasks.

Task	${t_i}^P$	${t_i}^B$	vp_{kl}	et_{kl}	et_{kPred}	TTD
1	70	70	0.1818	0.5343	0.2357	279
6	49	49	0	0.0297	0.1467	334
7	52	52	0.2365	0.1087	0.1413	368
8	68	68	0.0101	0.1757	0.1521	447
9	80	80	0.0850	0.2944	0.1626	492
10	53	53	0.1755	0.1185	0.1370	387
14	27	27	0	0.0915	1.5	18
15	5	5	0.1417	0.0117	0.0725	69

As shown in Table 1, the execution time of the related tasks is calculated according to the items in the workflow.

4.1. Implement Adaptive Fault Tolerance Scheduling

As mentioned in the previous chapter, in the proposed method, when a workflow is entered, it first approves or rejects the task acceptance scheduler according to the number of virtual machines and the connections between the tasks. After confirming the acceptance of the workflow, a backup of the tasks is prepared by the task controller. The scheduler sends each version to a virtual machine to run. If the initial version of a task is successful, the initial version success information is sent to the resource manager. The resource manager then notifies the virtual machine to which it has sent the backup of the assigned task to cancel the backup. Otherwise, the resource manager will not notify the virtual machine to cancel the backup, and scheduling backups will normally be performed for fault tolerance. Since cloud systems resources are shared among all customers and workflows are dynamically sent to cloud systems at runtime, the above process for each stream something is repeated. In the continuation of this chapter, according to the implementation of the proposed method, the main factors that affect the quality of service are calculated in order to evaluate the performance of the proposed idea.

4.2. Implementation of Fault Tolerance

In the proposed method as mentioned in the previous chapter, the proposed method of predicting an error in a host can expedite the completion of tasks and not send new tasks to hosts suspected of error. In this module, error prediction is based on the average time of error, which was introduced in the previous section. As mentioned, the minimum value required for the average error time is always greater than or equal to the maximum execution time of a virtual machine from the same host. Given that virtual machines are heterogeneously distributed across a host, it can be said that the minimum value required for the mean error time is always greater than or equal to the maximum execution time of the slowest virtual machine from the same host. In order to implement the error according to the optimization model specified in Equation 3, the fault tolerance rate and error prediction in cloud resources are shown in Table 2.

As shown in Fig. 3, the error tolerance rate increases at a slower rate due to the assignment of tasks and the use of backups.

Table 2. An example of fault tolerance calculation rate.

VM	Fault Tolerance Rate	Fault Prediction
1	0.833	0.2356
2	0.833	0.07633
3	0.875	0.1572
4	0.9167	0.108
5	0.9167	0.2744
6	1	0.1467
7	1	0
8	1	0.1521

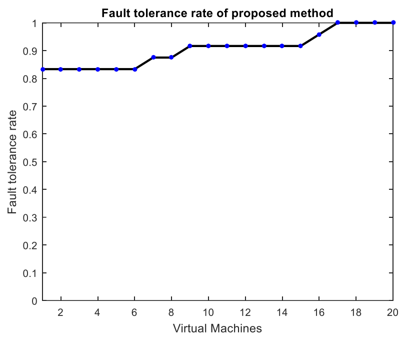


Fig. 3. Diagram of fault tolerance rate in the proposed method.

4.3. Implement the Cost in the Proposed Method

According to the proposed method, the cost of completing tasks in resources consists of costs related to processing and costs related to the transfer of information in the network between resources while performing related tasks. Therefore, it can be noted that the processing cost for a given task depends on the task volume and the fixed cost (processor price per instruction). Also, in order to calculate the cost of data transfer in the cloud network between resources, the relationship between tasks can be used to calculate the volume of information sent in the network and the cost of this information. Table 3 shows the costs associated with completing tasks in each resource in the cloud environment.

Table 3. The calculated cost of completing tasks in cloud resources.

VM	C_{total}
1	0.0585994181021479
2	0.0189793098978603
3	0.0391018838637520
4	0.268519276434928
5	0.0682357081722127
6	0.0364755060581933
7	0
8	0.0378227500156913

As shown in Table 3, the cost of performing tasks in the resources to which the independent tasks are assigned is less than the resources in the dependent resources. Resources on which the cost of performing tasks is zero are resources to which no task has been submitted; these resources are the same resources that have been penalized by the proposed method and will not be assigned a task until the end of the workflow. Fig. 4 shows the cost-completion task chart in the resources.

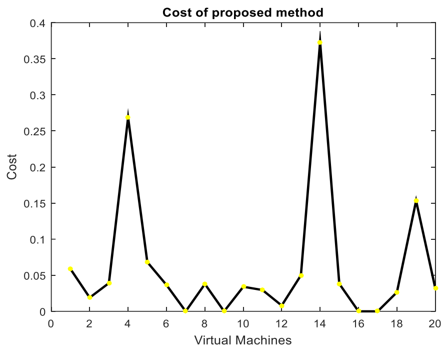


Fig. 4. Cost of completing tasks in each resource.

Also, in order to show the total costs in the resources to perform the tasks in the proposed method, the cumulative cost function is used, the diagram of the cumulative cost function is shown in Fig. 5.

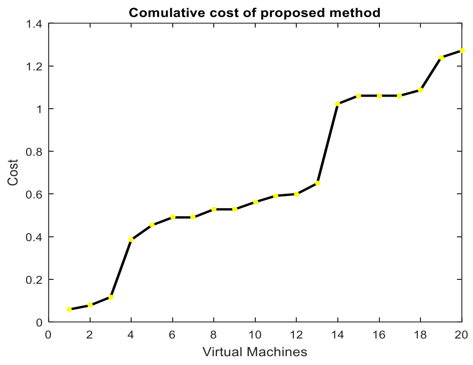


Fig. 5. Cumulative function diagram of the cost of completing tasks in all resources.

As shown in Fig. 5, the cost of completing tasks in all sources has increased slightly, and finally for 20 virtual machines in 4 hosts, this cost has reached 1.25.

4.4. Implement Energy in Cloud Resources

As mentioned in the previous chapter, due to the fact that virtual machines are heterogeneous in the proposed method, the energy required to perform tasks varies from source to source. Therefore, in each fixed source, energy consumption will be different from other sources. Fig. 6 shows the energy consumption diagram in the proposed method.

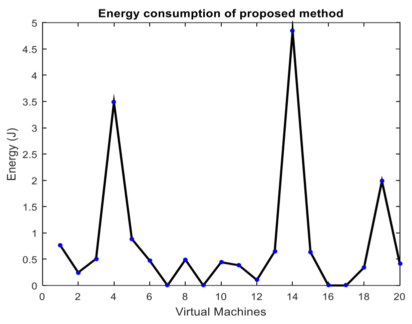


Fig. 6. Energy consumption in cloud resources.

As shown in Fig. 6, resources where independent tasks are loaded consume less energy, while energy consumption in resources with dependent tasks is higher.

4.5. Completion of Tasks in the Proposed Method

In order to implement the task execution time in the proposed method, the total task completion time in the cloud environment includes the task execution time and the transmission time in the network. Therefore, the time of completion of tasks in the cloud environment is directly related to independent and dependent tasks. In other words, resources that need to exchange information with other sources need more time to complete tasks. Naturally, there will be more delays in completing tasks in such resources. Fig. 7 shows the completion time chart in each resource. Fig. 8 also shows the delay diagram in the proposed method.

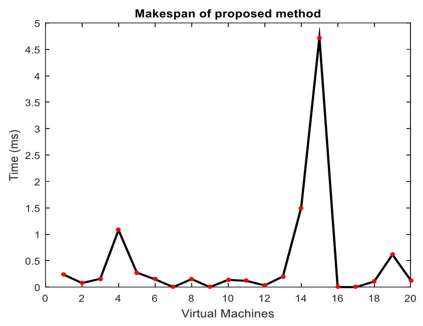


Fig. 7. Completion of tasks in resources.

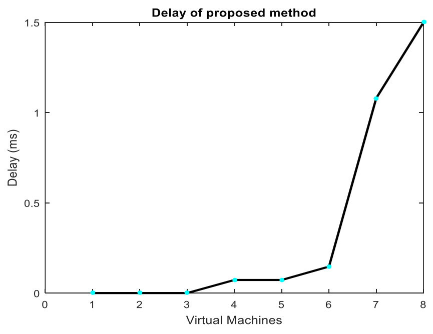


Fig. 8. Delay diagram in the proposed method.

Also, in order to calculate the completion time of all tasks in all resources, the cumulative time function is used, which summarizes the time spent in each resource and shows it graphically during the completion of tasks in the resources. Fig. 9 shows the cumulative function diagram of task completion time across resources.

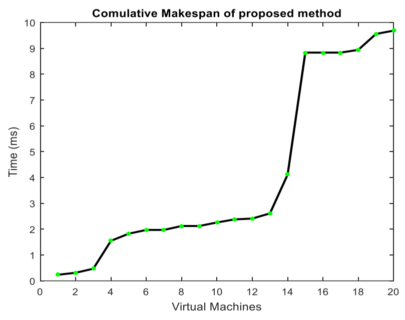
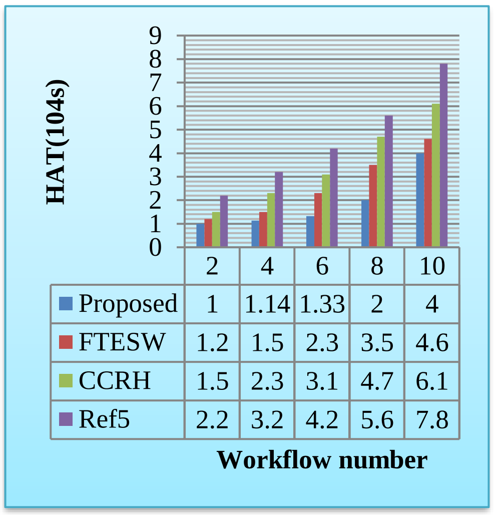


Fig. 9. Makespan cumulative function diagram.

As shown in Fig. 9, the task completion time, which increases due to the availability of resources with dependent tasks, eventually reaches 9.5 seconds for a 30-task workflow at 20 sources in 4 hosts.

4.6. Comparing the Proposed Method with the Previous Work

In order to compare the proposed method with the previous method, in this section we compare the proposed method with other scheduling methods in the basic article. This comparison is first based on the criterion (HAT). This criterion indicates the number of active hosts when assigning tasks to virtual machines. Fig. 10 shows a comparison between the proposed method and the previous methods in terms of HAT criteria.

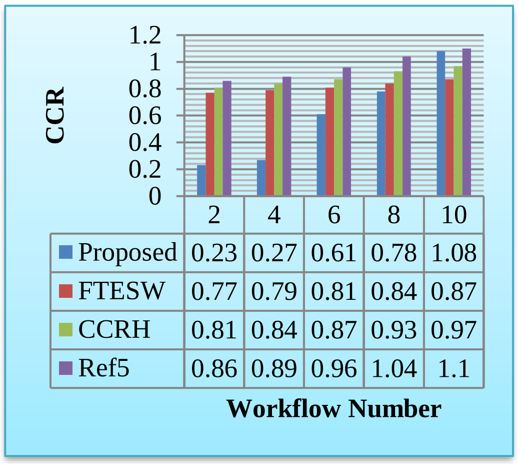


	2	4	6	8	10
🟦 Proposed	1	1.14	1.33	2	4
🟥 FTESW	1.2	1.5	2.3	3.5	4.6
🟩 CCRH	1.5	2.3	3.1	4.7	6.1
🟪 Ref5	2.2	3.2	4.2	5.6	7.8

Fig. 10. Comparison of the proposed method with previous methods in terms of HAT criteria.

As shown in Fig. 10, the number of active hosts in the proposed method is less than the other methods in the basic article, and the amount of energy consumed will be less according to the proposed method.

Another criterion used in this dissertation for comparison is the criterion (CCR), which is the average cost of transferring tasks over the cost of performing tasks, which is compared in Fig. 11 between the proposed method and previous methods in terms of The CCR criterion.



	2	4	6	8	10
🟦 Proposed	0.23	0.27	0.61	0.78	1.08
🟥 FTESW	0.77	0.79	0.81	0.84	0.87
🟩 CCRH	0.81	0.84	0.87	0.93	0.97
🟪 Ref5	0.86	0.89	0.96	1.04	1.1

Fig. 11. Comparison of the proposed method with previous methods in terms of CCR criteria.

As shown in Fig. 11, the transfer cost per implementation of the proposed method is lower than other methods in the base article. Therefore, it can be said that the performance of the proposed method has been better in allocating tasks to resources and reducing the time and cost of completing the entire work and error tolerance.

5. CONCLUSION

In the cloud environment, computing resources need to be timed so that both providers get the most out of their resources and users get the applications they need at the least time and cost. On the other hand, given that the cloud environment is heterogeneous and needs and demands are changing at all times, a good decision to schedule the workflow in the cloud environment is complex and vital at the same time. Therefore, in this study, an adaptive workflow scheduling approach is proposed to increase fault tolerance in cloud computing. The present approach calculates the probability of failures for each resource according to the execution time of tasks on the resources. In fact, in the present method, a time limit is set for each of the tasks. If the task is not completed within the specified time limit, the probability of failure in the source increases and the next tasks are not sent to the desired source. The simulation results show that the proposed method, in addition to increasing error tolerance, also improves other factors affecting service quality.

REFERENCES

[1] V. Mohammadian, N. J. Navimipour, M. Hosseinzadeh, and A. Darwesh, **"Comprehensive and Systematic Study on the Fault Tolerance Architectures in Cloud Computing,"** *Journal of Circuits, Systems and Computers*, p. 2050240, Jun. 2020, doi: 10.1142/s0218126620502400.

[2] T. Welsh and E. Benkhelifa, **"On Resilience in Cloud Computing,"** *ACM Computing Surveys*, Vol. 53, no. 3. Association for Computing Machinery, Jun. 01, 2020, doi: 10.1145/3388922.

[3] M. Nazari Cheraghlou, A. Khadem-Zadeh, and M. Haghparast, **"A survey of fault tolerance architecture in cloud computing,"** *Journal of Network and Computer Applications*, Vol. 61. Academic Press, pp. 81–92, Feb. 01, 2016, doi: 10.1016/j.jnca.2015.10.004.

[4] S. Kumar, D. S. Rana, and S. C. Dimri, **"Fault tolerance and load balancing algorithm in cloud computing: A survey,"** *International Journal of Advanced Research in Computer and Communication Engineering*, Vol. 4, No. 7, pp. 92–96, 2015.

[5] U. Dwivedi and H. Dev, **"A review on fault tolerance techniques and algorithms in green cloud computing,"** *Journal of Computational and Theoretical Nanoscience*, Vol. 15. American Scientific Publishers, pp. 2689–2700, Sep. 01, 2018, doi: 10.1166/jctn.2018.7560.

[6] M. Hasan and M. S. Goraya, **"Fault tolerance in cloud computing environment: A systematic survey,"** *Computers in Industry*, vol. 99. Elsevier B.V., pp. 156–172, Aug. 01, 2018, doi: 10.1016/j.compind.2018.03.027.

[7] D. Jain, N. Zaidi, R. Bansal, P. Kumar, and T. Choudhury, **"Inspection of fault tolerance in cloud environment,"** in *Advances in Intelligent Systems and Computing*, 2018, Vol. 672, pp. 1022–1030, doi: 10.1007/978-981-10-7512-4_103.

[8] C. Kathpal and R. Garg, **"Survey on Fault-Tolerance-Aware Scheduling in Cloud Computing,"** in *Lecture Notes in Networks and Systems*, Vol. 40, Springer, 2019, pp. 275–283.

[9] P. Kumari and P. Kaur, **"A survey of fault tolerance in cloud computing,"** *Journal of King Saud University - Computer and Information Sciences*, Oct. 2018, doi: 10.1016/j.jksuci.2018.09.021.

[10] G. Singh and S. Kinger, **"A survey on fault tolerance techniques and methods in cloud computing,"** *International Journal of Engineering Research and Technology*, Vol. 2, No. 6, 2013.

A. S. Abohamama, M. F. Alrahmawy, and M. A. Elsoud, **"Improving the dependability of cloud environment for hosting real time applications,"** *Ain Shams Engineering Journal*, Vol. 9, No. 4, pp. 3335–3346, Dec. 2018, doi: 10.1016/j.asej.2017.11.006.

[11] H. Yan, X. Zhu, H. Chen, H. Guo, W. Zhou, and W. Bao, **"DEFT: Dynamic Fault-Tolerant Elastic scheduling for tasks with uncertain runtime in**

cloud," *Information Sciences,* Vol. 477, pp. 30–46, Mar. 2019, doi: 10.1016/j.ins.2018.10.020.

[12] Y. Ding, G. Yao, and K. Hao**, "Fault-tolerant elastic scheduling algorithm for workflow in Cloud systems,"** *Information Sciences*, Vol. 393, pp. 47–65, Jul. 2017, doi: 10.1016/j.ins.2017.01.035.

[13] S. Talwani and I. Chana, **"Fault tolerance techniques for scientific applications in cloud,"** in *2nd International Conference on Telecommunication and Networks, TEL-NET 2017*, Apr. 2018, vol. 2018-January, pp. 1–5, doi: 10.1109/TEL-NET.2017.8343578.