Fault Tolerant Scheduling of Workflows using Improved Check Pointing Technique

[1] Kanagaraj K, [2] Swamynathan S
[1] Assistant Professor, Department of MCA, MEPCO Schlenk Engineering College, Sivakasi
[2]Professor, Department of IST, CEG, Anna University Chennai
[1]kanagaraj@mepcoeng.ac.in, [2] swamyns@annauniv.edu

Abstract

Workflow is a set of tasks arranged according to the data and control dependency. Fault tolerant workflow scheduling is difficult to achieve due to its complex architecture and dynamic resource requirements. However the emergence of cloud has provided new hope for ensuring fault tolerance in workflow scheduling. The elasticity in hiring and releasing resources in the cloud helps to reduce the node failures when scheduling workflows. However workflows with task and data dependencies introduce various timing and consistency problems. To enable fault tolerance, the state of an executing program is saved on a stable storage for recovering the system even after failure. The time at which the state is stored is called as check point. Adding numerous check points will improve the reliability, but affects the performance. On the other hand the less number of check points will need more time to recover from failure. In this paper, an Improved Check Pointing Technique (ICPT) is proposed. The ICPT determines the optimal check point interval based on the mean time between failure and mean down time. It reduces the check pointing overhead by 33% and identifies a system with maximum availability for storing the check pointed image. Experimental results shows that the proposed system ensures high fault tolerance during workflow scheduling and reduces the check pointing overhead when compared to the existing techniques

Keywords: *Check Pointing, Fault tolerance, Failure; Recovery, Workflow Scheduling*

1. Introduction

The emerging computing technologies are made up of gigabit networks and high-speed microprocessors. These computing environments consist of processors from several servers and communication between sites need transmission along several intermediate hops. This creates an important problem called as resource failure that needs to be handled effectively. Among all the failure management techniques, check pointing is an important technique that stores the state of a task at regular intervals. The check pointed values can be used to recover the system after failure and also prevents the loss of work that has been completed earlier.

Check points can be calculated using a variety of techniques. The frequency of the check points is of great concern when check pointing lengthy applications. When check pointing parallel applications, parallelism helps to improve the execution speed, however increases the chances for failure. In a cloud computing environment check pointing helps each VM to resume computing after failure. The reliability of the system need to be considered while calculating check points. Based on the system's reliability, the number of check points can be increased or decreased.

The process of completing a failed task is called as recovery. Based on the nature of the application a forward or a backward recovery scheme can be applied. It is necessary to build a system with high availability by reducing the recovery time. Failure diagnosis is the process of finding a breakdown and identifying the failed component. This can be achieved by sending heart beat message between the systems. If any of the system does not send or receive heart beat message then the probability of failure is more. When executing real time scientific workflows in the cloud, it requires highly fault tolerant resources, hence workflow scheduling in the cloud requires more focus on providing a fault free environment.

Check pointing and recovery techniques are very important to ensure the availability of a distributed environment. Check pointing is the process of periodically saving the state of an executing program to stable storage, from which the system can recover after a failure. The files containing the saved data are called as check pointed files. Identifying a suitable system for storing the check pointed file is also an important problem. Recent research also focuses on storing the check pointed image in the node memory. Workflows are special kind of parallel applications.

In a workflow, the number of parallel tasks may vary at different time intervals. So, applying the existing approach for check pointing parallel programs for workflows may not yield the desired results. As the resource requirements for workflows are dynamic the wise decision is to use the cloud resources for scheduling them. Using cloud computing for scheduling workflows using virtual machines also provides the following additional benefits.

1. Virtual machines are more fault tolerant than traditional machines.
2. Failure recovery is quick using live VM migration other state of the art cloud specific features.
3. Storing check pointing image and retrieving the saved images are simple in the cloud environment.

Hence, in this research an improved check pointing technique that harnesses the benefits of cloud resources for scheduling workflows is presented.

2. Literature Survey

The classification and survey of fault tolerant workflow management systems in the cloud and distributed computing environments are presented in [1]. In [2] the authors provide a fault-tolerance technique in dynamic systems that can help system designers to estimate the number of processors. Zhu et al. [3] proposed a new scheduling algorithm that can tolerate one node failures for real-time tasks in multi cloud and cluster environment. The adaptive check point interval placement algorithm [4] meets all tasks deadline. The check point intervals are adjusted to minimize the impact of stresses and permanent faults on the running hosts.

The work presented in [5] was aimed to minimize the workflow cost with the deadline constraint in the presence of internal and external failures. The Fault Tolerant Workflow Scheduling algorithm (FTWS) [6] provides fault tolerance by using replication and resubmission of tasks, based on the task priority. The replication of tasks depends on a heuristic metric which is calculated by finding the tradeoff between the replication factor and resubmission factor.

The heuristic metric is considered to avoid resource wastage. Tasks are prioritized based on the criticality of the task which was calculated by using parameters like out degree, earliest deadline and high resubmission impact. A recursive list-scheduling algorithm that exploits the M-SPG structure to assign sub-graphs to individual processors, and uses dynamic programming to decide how to check point these sub-graphs was proposed in [7].

The effect of spatial and temporal parameters for dynamic fault-tolerant workflow scheduling (DFTWS) [8] is useful to assign appropriate virtual machine for each task according to the task urgency and budget quota in the phase of initial resource allocation.

The effect of spatial and temporal redundancy was studied by Anghel et al. [9]. Since the occurrences of internal faults are usually unpredictable in computer systems, fault tolerance must be considered when devising workflow scheduling algorithms. Hwang et al. [10] present a fault tolerant mechanism which extends the primary backup model to cloud computing system.

Parallel execution is running a task on multiple resources simultaneously to guarantee a viable result, which results in a high spatial cost. Whereas, temporal redundancy relaxes time constraint to provide more time for re-executing the failed task on the original resources [11].

Chen et al. [12] propose three clustering strategies of fault tolerant to improve the QoS of workflow and construct a real-time workflow fault-tolerant model that extends the traditional primary-backup model based on many cloud computing characteristics, and the task allocation and message transmission mechanism are developed to ensure task faults can be done in the process of workflow execution. The advantages and disadvantages in dynamic scheduling of workflows was presented in [13].

Walters and Chaudhary [14] have presented a detailed survey of the different techniques used in application level check pointing that is very helpful for check pointing parallel tasks. The performance of synchronous check pointing in a distributed computing environment with and without load redistribution was presented in [15][17].

A detailed analysis of the existing workflow scheduling schemes was presented in [18]. A technique to provide elastic resource provisioning based on the budget and deadline constraints of workflows [19] was suitable for scientific workflow

However, these fault-tolerant scheduling algorithms [20] cannot be directly applied to cloud computing environment or workflow scheduling problem as the fault-tolerant methods mentioned above only consider the non repairable system.

Due to the recent advancements we can harness the benefit of repairable systems in the form of VMs in the cloud. This paper proposes a novel idea used to provide a fault tolerant in the cloud using repairable VMs. The major contributions of this work are

1. To provide a fault tolerant environment for scheduling workflows in the cloud

2. To calculate the check point interval based on the MTBF (Mean Time Between Failure) and MDT(Mean Down Time) of the VMs used for executing the tasks

3. To select a system with more availability for storing the check pointed image based on the MTTF (Mean Time To Failure) and MTTR (Mean Time To Recover) of the available VMs.

3. Improved Check Pointing Technique (ICPT)

The improved check pointing technique proposed in this paper is aimed to calculate the efficient check pointing interval and to identify a system with more availability to store the check pointed image. This minimizes the number of check points and speeds up the workflow execution.

The ICPT uses the MTBF (Mean Time Between Failure) and MDT (Mean Down Time) of all the virtual machines for calculating the check point interval. The mean time between failures is calculated using equation (1).

$$MTBF = \frac{\sum(Start\ of\ Down\ time - Start\ of\ up\ time)}{Number\ of\ Failures} \qquad (1)$$

From equation (2), it is observed that the mean time between failures is obtained by summing the difference between the up time and down time of the system and dividing it by the number of failures. Figure 2 shows the process of calculating the MTBF of a system.

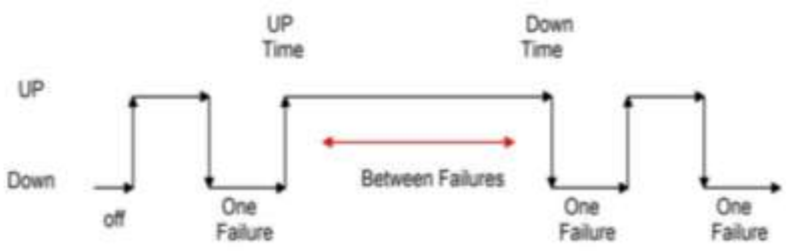

Figure 2 Mean time between failure of a system

For workflow applications the the MTBF can be calculated by adding the average MTBF of all the VMs used for executing the workflow and dividing it by the total number of VMs, as shown in equation 2.

$$AMTBF = \frac{\sum_{i=1}^{n} MTBFi}{n} \tag{2}$$

MTBF is defined by the arithmetic mean value of the reliability function $R(t)$, which can be expressed as the expected value of the density function $f(t)$ of time until failure.

$$MTBF = \int_0^\infty R(t)dt = \int_0^\infty t\, f(t)dt \tag{3}$$

The MTBF for the parallel system with two parallel repairable VMs can be calculated as per equation (4).

$$MTBF(c1||c2)=\frac{MTBF(c1)*MTBF(c2)}{MTBF(c1)+MTBF(c2)} \tag{4}$$

Similarly the mean down tome of the VMs can be calculated by adding the average MDT of all the VMs used for executing the workflow and dividing it by the total number of VMs, as shown in equation 5.

$$MDT = \frac{\sum(Start\ of\ Up\ time - Start\ of\ Down\ time)}{Number\ of\ Failures} \tag{5}$$

The MDT of n virtual machines used for executing the workflow can be calculated as per equation (6).

$$MDT(c1|| \dots ||cn)=\sum_{k=1}^{n} \frac{1}{MDT(c_k)} \tag{6}$$

The MTTF and MTTR are used to calculate the availability of the system by selecting a system that has long mean time to failure and has short mean time to repair as shown in equation (7).

$$System\ Availability = MTTF(MTTF + MTTR) \tag{7}$$

The entire process of the ICPT proposed in this research is shown in Figure 3.

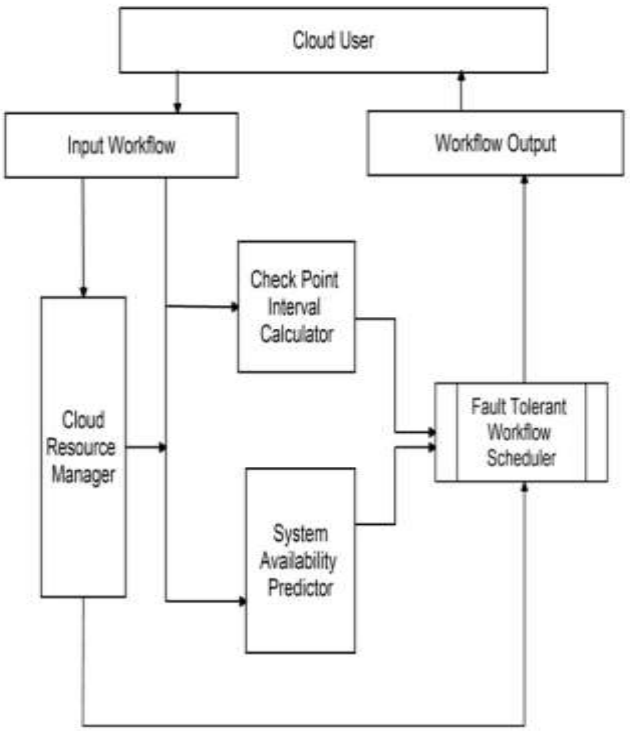
Cloud User
Input Workflow
Workflow Output
Check Point Interval Calculator
Cloud Resource Manager
Fault Tolerant Workflow Scheduler
System Availability Predictor

Figure 3 Architecture of Fault Tolerant Workflow Scheduler using ICPT

4. Implementation, Results and Discussion

To demonstrate the proposed ICPT, consider a workflow with 15 nodes as shown in Figure 4. The execution time of the tasks is shown within the circle. The task execution starts at tasks1 and completes at task15. In between there are 13 nodes that are arranged in different order. The parallel nodes at different levels. We have used the structure aware resource estimation technique proposed in our earlier research [16] to calculate the optimal number of VMs required for executing the workflow. Hence it is desirable to have 4 VMs for executing the workflow without affecting the deadline. The starting time and ending time of each task is calculated based on the task dependencies. It means that a task can start its execution only when all the dependent tasks have completed their execution. The implementation is done using CloudSim. Considering the task dependencies, the start time and end time of the tasks are calculated and presented in Table 1.

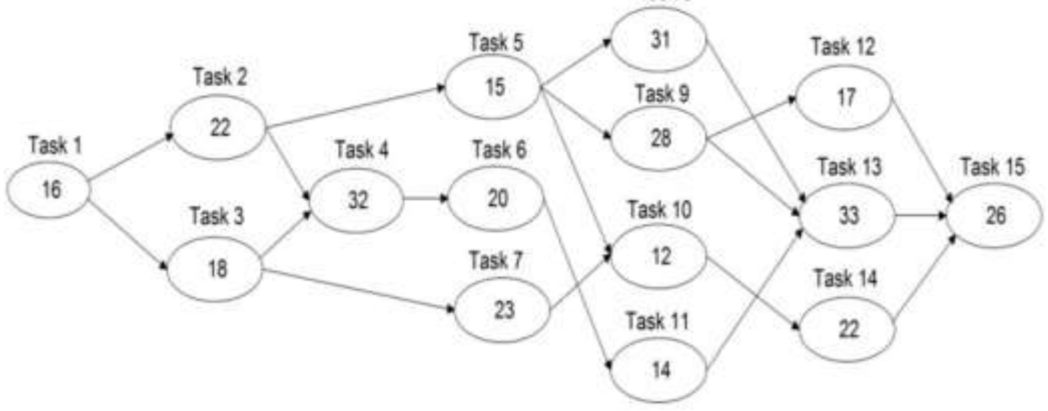

Figure 4 Workflow with 15 nodes

As the tasks are arranged in parallel, the number of virtual machines used should be equal to the maximum number of parallel tasks. In this workflow, the maximum number of parallel tasks is 4. So, four VMs are used to execute the tasks, without affecting the deadline. For the workflow in Figure 4, the task execution with different VMs is represented in Table 1.

Figure 3 shows how the tasks are assigned to different VMs for parallel execution. It is arranged in such a way that the tasks do not wait for a resource. Though the tasks can be executed without delay, there are other parameters that affect the execution. The most important factor among them is the VM failure. VM failures should be taken in to account while executing tasks in a workflow, as no system is 100% perfect.

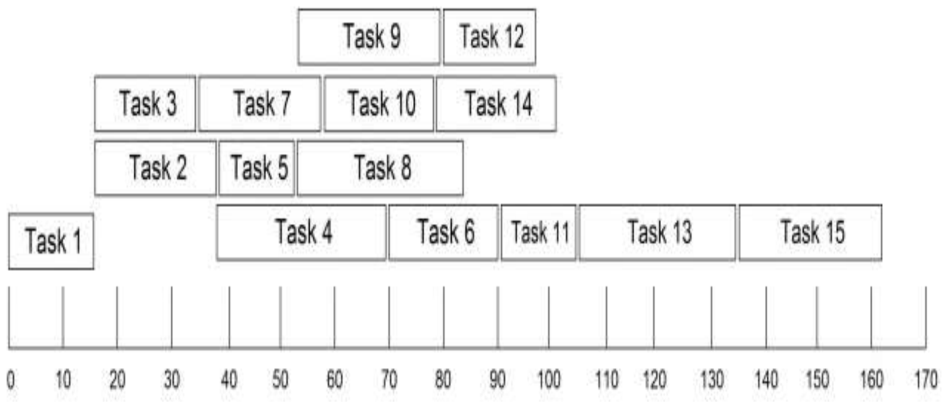

Figure 3 Sequence of Task Execution in the sample workflow

Table 1. Start Time and End Time of tasks in the sample workflow

Tasks	**T1**	**T2**	**T3**	**T4**	**T5**	**T6**	**T7**	**T8**	**T9**	**T10**	**T11**	**T12**	**T13**	**T14**	**T15**
Start Time	-	16	16	38	38	70	34	53	53	57	90	81	104	79	137
Finish Time	16	38	34	70	53	90	57	84	81	79	104	98	137	101	163

There are several methods available for calculating the failure, the most common factor is the MTTR, MTTF, MTBF and MTD. In order to recover from these failures the obvious technique is to have check points. The check point interval for the sample workflow is 25 minutes, calculated using the Optimal

Check Point Interval (OCPT). The formula to calculate the optimal check point interval is shown in equation (8).

$$Optimal\ Checkpoint\ Interval = \frac{\sqrt{MTTF * t_c}}{h} \qquad (8)$$

Here, t_c is the time at which check pointing is initiated and h is the average percentage of normal operation in the interval before failure. Figure 4 depicts the checkpoint locations and the number of checkpoints for the sample workflow using OCPT.

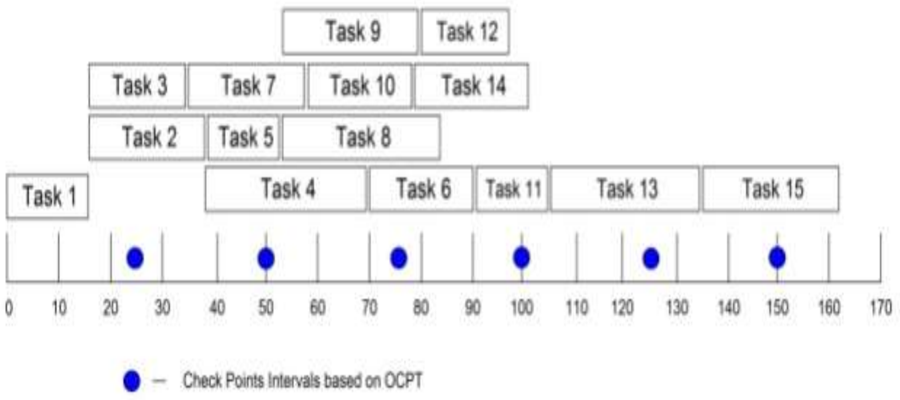

Figure 4 Check Point Interval using OCPT

4.1 Check point interval calculation using ICPT

The ICPT calculates the check pointing interval based on the mean time between failure and the mean down time. The MTBF and MDT of the virtual machines used for executing the workflow are calculated and presented in Table 2.

Table 2 MTBF and MDT of the sample workflow

VMs used for Execution	Average Annual Hours Used	Average Number of Failures	MTBF (Minutes)	MDT (Minutes)
VM1	4380	15	116	34
VM2	5000	12	124	24
VM3	4250	16	116	38
VM4	3900	13	120	33

Using the values in Table 2 the mean down time of the virtual machines is calculated.

Mean Down Time = $MDT(VM1\big||VM2|\big|VM3||VM4) = \left(\sum_{k=1}^{4} \frac{1}{MDT(C_k))}\right)$ = 34

Using these values, the check point interval is 34 Mins. The check pointing locations of the workflow using the improved check pointing technique is shown in Figure 5.

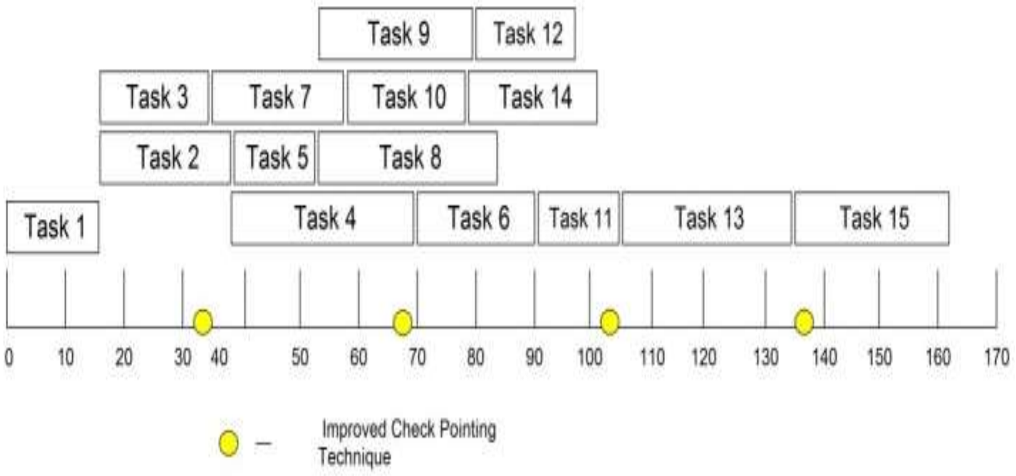

Figure 5 Check Points based on the interval calculated using ICPT

4.2 Comparison of performance of ICPT and OCPT

The check pointing intervals calculated using the optimal check pointing technique and the improved check pointing technique are applied for the sample workflow. The experiment is carried out using the popular simulator the CloudSim and the results obtained are plotted in Figure 6.

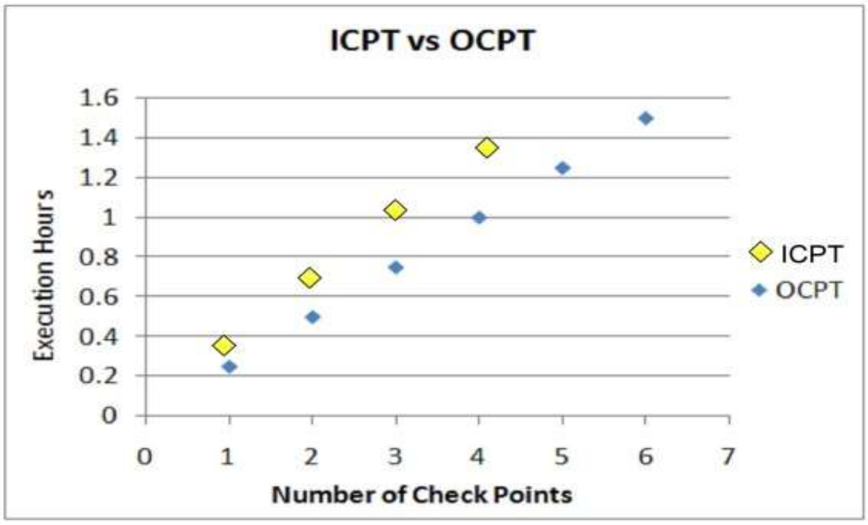

Figure 6 Comparison of check pointing intervals using OCPT and ICPT

The number of check points as per OCPT is 6 and the number of check points as per ICPT is 4. The total percentage of reduction in check points is 33%. It will be very useful to save the overall cost involved in executing the workflow. Moreover it will also have tremendous savings in execution cost when applied to large scientific workflows like montage, cybershake, ligo, epigenomics and sipht.

4.3 Predicting System Availability using ICPT

The system availability is an important factor to be considered for storing the check pointed file. Particularly in parallel execution there is a compelling need to select a system with more availability. In this work, the system availability is calculated based on the MTTF and MTTR of the VMs used of execution and the VM with more availability is used for storing the check pointed image file.

Table 3 Availability percentage of virtual machines

VMs used for Execution	Average Annual Hours Used	MTTF per Thousand Nodes	MTTR in Hrs	System Availability %
VM1	4380	1	0.7	99.8
VM2	5000	1	0.5	99.9
VM3	4250	2	0.5	98.6
VM4	3900	1	0.9	99.5

From the above Table 3, it is understood that the VM2 has high availability when compared to other VMs used for execution. Hence it is wise to use VM2 for storing the check pointing images.

5. Conclusion and Future Enhancements

The ICPT proposed in this paper is a novel idea for fault tolerant workflow scheduling in the cloud. The system calculates the mean time between failure and the mean down time of the virtual machines used in executing a workflow. This reduces the frequency of check pointing interval by 33% and contributes for considerable savings in the cost. Also it increases the system availability by storing the check pointed image in a system that has more availability. The effectiveness of ICPT is proved by comparing it with the optimal check pointing technique. Hence it is recommended to apply ICPT when scheduling workflows in the cloud.

References

1. D. Poola, M.A. Salehi, K. Ramamohanarao, R. Buyya, "A taxonomy and survey of fault-tolerant workflow management systems in cloud and distributed computing environments. In Software Architecture for Big Data and the Cloud", Elsevier, Amsterdam, The Netherlands, 2017, pp. 285–320.
2. Ghosh, S, Melhem, R, Mossé, D, "Fault-tolerance through scheduling of a periodic tasks in hard real-time multiprocessor systems", IEEE Transactions on Parallel and Distributed Systems, 1997, no. 8, 272–284.
3. Zhu, M.M, Cao, F, Wu, C.Q, "High-throughput scientific workflow scheduling under deadline constraint in clouds", Journal of Communication, 2014, no. 9, 312–321.
4. Mohamad, B. Bandan, , S. Bhattacharjee, D. K, Pradhan, J. Mathew, "Adaptive Check point Interval Algorithm considering Task Deadline and Lifetime Reliability for Real-Time System", Procedia Computer Science, 2015, vol. 70, pp. 821-828.
5. L. Zhongjin, Y. Jiacheng, H Haiyang, J. Chen, Hu.H, Ge. J and V. Chang , "Fault-Tolerant Scheduling for Scientific Workflow with Task Replication Method in Cloud", In Proceedings of the 3rd International Conference on Internet of Things, Big Data and Security, 2018, pp. 95-104.
6. S.K. Jayadivya, S.J.Nirmala, M.S. Bhanu, "Fault tolerant workflow scheduling based on replication and resubmission of tasks in Cloud Computing", International Journal on Computer Science and Engineering, 2012, vol. 4, pp. 996-1004.
7. H. Li, Canon, L.C, Casanova, H, Robert. Y, and Fr ́ed ́ eric Vivien, "Check pointing Workflows for Fail-Stop Errors", IEEE Transactions on Computers, 2018, pp.1-16.
8. N. Wu , D. Zuo and Z. Zhang, "Dynamic Fault-Tolerant Workflow Scheduling with Hybrid Spatial-Temporal Re-Execution in Clouds", 2019, Vol. 10, no. 169, pp. 1-18.
9. L. Anghel, D. Alexandrescu, M. Nicolaidis, "Evaluation of a soft error tolerance technique based on time and/or space redundancy. In Proceedings of the 13th Symposium on Integrated Circuits and Systems Design", Manaus, Brazil, 18–24 September 2000; pp. 237–242.
10. S. Hwang, C. Kesselman, "Grid workflow: A flexible failure handling framework for the grid", In Proceedings of the 12th IEEE International Symposium on High Performance Distributed Computing, Seattle, WA, USA, 22–24 June 2003, pp. 126–137.
11. Y. Gao, S.K. Gupta, Y. Wang, M. Pedram, "An energy-aware fault tolerant scheduling framework for soft error resilient cloud computing systems", In Proceedings of the Conference on Design, Automation & Test in Europe. European Design and Automation Association, Dresden, Germany, 24–28 March 2014, p. 94.

12. H. Chen, F.Z.Wang, N.Helian, “Entropy4Cloud: Using Entropy-Based Complexity to Optimize Cloud Service Resource Management”. IEEE Transaction. Intel. 2018, 2, 13–24.
13. E.N. Alkhanak, S.P.Lee, R.Rezaei, R.M.Parizi, “Cost optimization approaches for scientific workflow scheduling in cloud and grid computing: A review”, classifications, and open issues”, J. Syst. Softw. 2016, 113, 1–26.
14. J.P. Walters, and V. Chaudhary, “Application-Level Check pointing Techniques for Parallel Programs”, ICDCIT 2006, pp. 221-234.
15. K.F. Wong and M. Franklin, “Check pointing in Distributed Computing Systems, Journal of Parallel And Distributed Computing”, 1996, vol. 35, pp. 67–75
16. K. Kanagaraj, and S.Swamynathan, “Structure aware resource estimation for effective scheduling and execution of data intensive workflows in cloud”. Future Generation Computer Systems, 2018, 79, 878–891.
17. G. Manimaran, C.S.R. Murthy, “A fault-tolerant dynamic scheduling algorithm for multiprocessor real-time systems and its analysis”. IEEE Trans. Parallel Distrib. Syst. 1998, 9, 1137–1152.
18. M. Masdari, , S.ValiKardan, Z.Shahi, S.I. Azar, “Towards workflow scheduling in cloud computing: A comprehensive analysis”, J. Netw. Comput. Appl. 2016, 66, 64–82.
19. C. Lin, S.Lu, “Scheduling scientific workflows elastically for cloud computing”. In Proceedings of the 2011 IEEE 4th International Conference on Cloud Computing, Washington, DC, USA, 4–9 July 2011; pp. 746–747.
20. T.Herault, Y. Robert, A.Bouteiller, D.Arnold, K.B. Ferreira, G.Bosilca, J. Dongarra, “Optimal Cooperative Check pointing for Shared High-Performance Computing Platforms” , Research Report, 2017, pp. 1-15.