International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

WBFAS: Workflow based Failure-Aware Scheduling in Grid Computing

Manjeet Singh*[1], Dr. Javalkar Dinesh Kumar[2]

Submitted: 29/10/2022 **Revised:** 15/12/2022 **Accepted:** 01/01/2023

Abstract: Scheduling is a difficult problem in general because it is an NP-complete problem; this is true whether it is being done in Grid or in any other environment. When tasks are dependent on one another the problem becomes more complex. NP-complete problem does not have a predetermined heuristic to describe them. It's possible that a particular heuristic will function well in some circumstances but not in others, and this makes the scheduling more crucial and critical. With the goal that the application performance will be improved and the resulting throughput will be optimized, a workflow based failure aware scheduling approach (WBFAS) is proposed in this research to solve scheduling problem for dependent task in large scale system like grid computing. The workflow of dependent task is represented by directed acyclic graph (DAG). The WBFAS method is based on incremental checkpoint fault tolerant mechanism and failure information of resources. The result analysis shows that proposed method WBFAS reduces the makespan and number of failures of the system while increasing the reliability and system performance.

Keywords: *Directed Acyclic Graph (DAG), Fault Tolerance, Grid Computing, Reliability, Scheduling, Workflow.*

1. Introduction

As the scientific problem becomes more complex in the context of modern computing technologies, an organisation needs more computational resources like more processing power and storage space etc. Distributed computing takes on a new form with grid computing, which creates a seamless connection between all of the systems, databases, and users. The development of new technologies has made it possible to use resources in a decentralised setting to address the growing number of issues that arise in the fields of science, engineering, and research [1]. Grid designs offer a middleware technology that may be utilised for a variety of purposes; including resource allocation, task scheduling, authorization, data management, and security. Grid is an integration of many sorts of resources, and it is considered to be an ideal infrastructure because it has a variety of resources at once, including processing units, storage units, and communication units. In general, grids can be divided into the two categories: computational grid and data grid [2]. Computational grid fulfils the processing requirements posed by difficult scientific issues, high-performance computing. A computational grid is a network of interconnected nodes that enables the processing of large-scale activities, improves resource utilisation, and satisfies the necessity for quick access to resources on demand by providing computational capacity. Data grid is the storage component of a grid environment. It performs the function of a massive data storage system and is responsible for the storing, sharing, and management of a large quantity of dispersed data [2].

There are various grids, such as departmental grids like Folding@Home and Global Grids that can be accessed via the internet. Some grids, like Folding@Home and Global Grids, are used to solve problems for specific groups. Compute Grids, such as smart grid, are used solely for the purpose of providing access to computational resources. In contrast to Compute Grids, which provide access to computational resources, Utility Grids provide access to resources. Extraprise Grids, such as Amazon.com, are established between companies, customers, etc [1]-[3].

In broad terms, scheduling refers to the process of making decisions regarding the distribution of jobs among resources. The most important function of a grid is to facilitate quick and easy access to resources that are located in different parts of the world. It is tough to create a schedule due to variability and dynamic nature of resources. The jobs are distributed among all of the available processors by the scheduler. Scheduling is an NP-complete issue; researchers are attempting to solve it by employing heuristics in the hopes of obtaining a solution that is somewhat close to the optimal answer [1], [2].

The high heterogeneity of the resources, the nature of the applications and the high demand for them, the distance covered by the network, and the large volumes of data that need to be shipped around as required are all factors that contribute to the complexity of the features that are enforced by grid architecture. When the number of hosts in a grid goes from ten to thousands, fault tolerance becomes the most important concern [4]. Tasks are broken down into a number of smaller subtasks and then scheduled according to the resources that are readily available in an environment.

[1] *Research Scholar, Department of Computer Science & Engineering Lingaya's Vidyapeeth, Faridabad, India*
ORCID ID : 0000-0002-0421-0668
[2]*Assistant Professor, Department of Electronics & Communication Engineering Lingaya's Vidyapeeth, Faridabad, India*
ORCID ID : 0000-0002-8168-3426
** Corresponding Author Email: 19phcs05w@lingayasvidyapeeth.edu.in*

In the current scenario of grid computing, task scheduling is a critical issue that needs to be addressed. It is necessary to have an effective task scheduling algorithm in order to make efficient use of the available resources and reduce the amount of time needed to finish everything. The grid scheduling problem requires optimization of a number of different objectives, such as completion time, work priority, resource utilization, QoS (Quality of Service) metrics, prices, dependability variables, and the resource requirements of the task, amongst other things. During the scheduling process, many task scheduling algorithms do not take into account the possibility that a task or resource would fail. Although the makespan is improved by certain work scheduling method, in spite of the occurrence of failures at individual grid nodes, it is possible to devise an effective scheduling method that is based on the failure information and performance parameters of resources [3], [5].

To solve scheduling problem in grid, a workflow based failure-aware scheduling (WBFAS) approach is proposed and this technique is extremely helpful in grid environments since there is the potential for any node to fail as a result of a number of different circumstances.

2. Literature Review

In this section, a review of already existing different methods of job scheduling have been done. The description is summarized in Table 1.

Table 1. Comparison of various existing scheduling algorithm

Author	***Description***
R. Garg et al. [6]	It was suggested to use an approach for dependent task scheduling that is fault resilient. Weibull failure distribution is the method employed, and the checkpoint rollback resolution of problems is used to address failure.
R. Garg et al. [7]	Developed an approach for the dependent job scheduling of the computing grid. Dependent task were modeled using a DAG, and the availability of resources was dynamic in nature. The simulation and analysis that uses dynamically chosen task graphs as well as task graphs directly relating to real world problems shows that the proposed method is able to deal failure and optimize performance.
Y. Zhang et al. [8]	It suggests some new approaches for integrating fault tolerance approaches with the dependent task scheduling algorithms and provides recommendation with HEFT and a DAG for scheduling, along with an over-provisioning mechanism and checkpointing methods.
A. Iosup et al. [9]	An examination of the resource constraints characteristics of Grid'5000 is performed, along with an assessment of availability trace. It provides the failure information and value of various parameters for Weibull distribution.
Z. Yu et al. [10]	Proposed a method for using failure prediction to schedule workflows that are aware of failures.
L. Yu et al. [11]	For grid computing systems, a new communication inclusion generational scheduling (CIGS) method that is based on DAG has been developed which is found to be effective.
M. Hemamalini et al. [12]	Examined a number of different scheduling algorithms. Using a multi-constrained graph, it discussed process scheduling in order to reduce the amount of data movement. In addition to that, it employs the concepts of weight vectors and ranks. Additionally, priority scheduling for dependent tasks based on sets of parallel tasks with the highest priority value was considered.
C. Chandrasekar et al. [13]	The study highlights the intricacy of the scheduling challenge and demonstrates the relevance of the approach for the development of effective grid schedulers.

The workflow based failure-aware scheduling (WBFAS) approach proposed in the next section is different from the above methods and found to be efficient as discussed in details in result analysis section.

3. Methodology

Failures are unavoidable in a grid environment due to the heterogeneity of the resources, and as a result, they consume a significant portion of the execution time. Therefore, the idea is to determine the anticipated amount of time that will be lost during the execution due to failure and recovery from failure. This information about wasted time is used to recalculate the resource computing capacity, and subsequent scheduling is carried out in such a way that we can minimize the wasted time caused by failures and improve the overall performance of the system.

The total amount of time spent taking checkpoints, recalculating portions of jobs that have failed, and recovering from previously saved checkpoints is the amount of time that is considered to be wasted. According to the findings of a number of studies, the pattern of errors that occur in a grid computing system is appropriate to Weibull distribution [14]-[18]. The equation for the anticipated amount of time lost for a Weibull distribution failure and an incremental checkpointing mechanism is given as Eq. 1 [19]:

$$Wasted\ Time = \int_0^{\infty} \left[\frac{O_F + mO_I}{m+1} \int_0^T n(t).dt + \frac{k}{n(T)} + (R_F + mR_I) \right] f(t).dt \quad (1)$$

Where,

- O_F and O_I denotes the time required for saving a full and incremental checkpoint respectively.
- R_F and R_I denote the time required for recovery from full and incremental checkpoint respectively.
- f(t) is a PDF (probability density function)
- k is a coefficient of recomputing time
- m is the number of incremental checkpoint between two full checkpoints
- n(t) is checkpoint function, given by Eq. 2 [19]

$$n(t) = \sqrt{\frac{(m+1)k}{O_F+mO_I} \cdot \frac{f(t)}{1-F(t)}} \quad (2)$$

$$f(t) = \left(\frac{\beta}{\alpha}\right) \cdot \left(\frac{t}{\alpha}\right)^{\beta-1} \cdot e^{-(t/\alpha)^{\beta}} \quad (3)$$

$$F(t) = 1 - e^{-(t/\alpha)^{\beta}} \quad (4)$$

Where F(t) is the cumulative distribution function and α, β respectively, are the scale and shape parameters [19]. The Eq. 2 can be rewritten as Eq. 5 when we use Eq. 3 and Eq. 4.

$$n(t) = \sqrt{\frac{(m+1)k}{O_F+mO_I}} \cdot \left(\frac{t}{\alpha}\right)^{\frac{\beta-1}{2}} \cdot \sqrt{\frac{\beta}{\alpha}} \quad (5)$$

The Figure 1 below show the flowchart of the proposed workflow based failure-aware scheduling (WBFAS) approach.

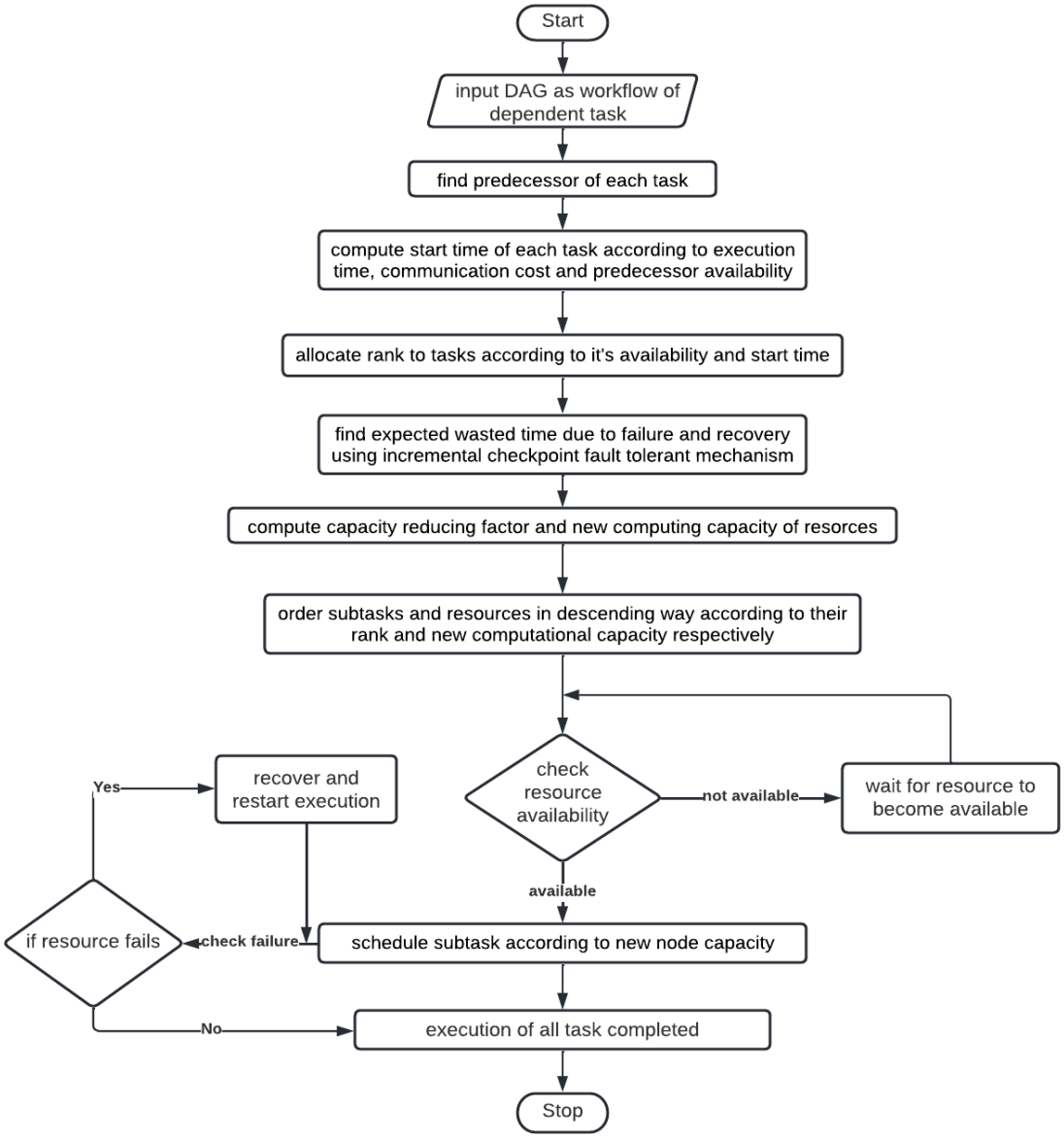

Fig 1. Flowchart for Workflow based Failure-Aware Scheduling Approach (WBFAS)

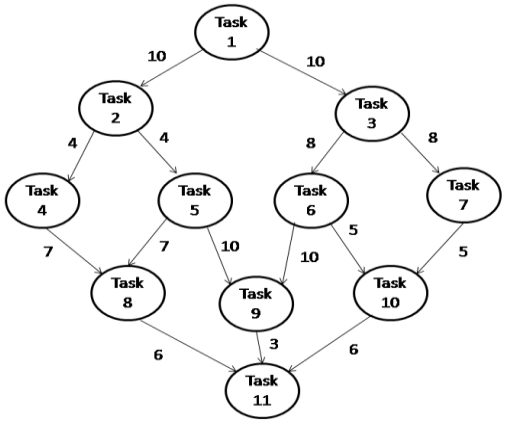

Fig 2. Simple Directed Acyclic Graph (DAG)

In workflow based failure-aware scheduling approach (WBFAS) the workflow of the dependent task is represented with the help of directed acyclic graph (DAG). A sample DAG is given above in Figure 2. The subtasks that make up the job are represented by the vertices of the graph. The dependency of a task is represented by the direction in which an edge extends through one node to another node in a DAG. For instance, the presence of an edge connecting nodes v1 and v2 indicates that node v2 is reliant on node v1. This means that the task v2 cannot begin its execution till the task v1 has completed its own execution and made its result accessible to task v2. In a DAG, the information sharing cost between pair of vertices is represented by the weight of the edge that connects them.

The process for the proposed methodology is stated as follows:

Step 1: Start by finding successor of each task in DAG based on workflow.

Step 2: Determine the predecessor subtask for each subtask based on the successor.

Step 3: Determine the average execution time for each subtask by measuring how long each subtask takes to complete on each resource.

Step 4: Determine the execution start time of a subtask based on its predecessor execution time and communication costs.

Step 5: Based on the execution start time of each subtask, assign a rank to it. Exit node will be having the lowest rank and entry node will be having maximum rank.

Step 6: Calculate the system's anticipated downtime caused by failure, recovery, and fault-tolerant mechanisms using Eq. 1.

Step 7: Determine the capacity reducing factor based on the anticipated system wasted time.

Step 8: Based on the capacity reducing factor, determine the reduced effective computing capacity of the resources.

Step 9: Sort tasks and resources according to decreasing of their rank and recomputed capacity respectively.

Step 10: Scheduling of tasks in accordance with rank and new capacity.

Step 11: Verify resource availability for all unscheduled and unexecuted tasks

```
while (workflow execution not over)
    if (resource available)
        schedule task
            if, (resource failure occurs during execution)
                recover from resource failure and restart execution;
            end if;
    else
        wait for the resource to become available
    end if;
end while;
```

Step 12: Finish;

4. Result Analysis

A grid model is simulated containing twenty computational resources. With a rising failure rate into consideration, the shape parameter can range from 1.8 to 3.6. The value of the scale parameter is set to equal 20. Both the checkpoint storage cost and the recovery time are equal to 2 minutes and 0.5 minutes respectively for full and incremental checkpoint. The value used for the re-computing time coefficients is 0.5. Grid applications run with a varying number of dependent tasks whose workflow is represented by DAG, typically falling somewhere in the range [20, 100]. The parameter values are referred from [9], [17], [18].

In order to verify that the workflow based failure-aware scheduling (WBFAS) algorithm is effective, we examine its performance in relation to that of speed-only scheduling approach (SOSA) using a variety of evaluation parameters. Only resource performance factors are taken into consideration by the SOSA algorithm when it is scheduling tasks. In order to test how well the suggested method performs, the performance metrics that are mentioned

below are utilized [20].

Performance Ratio: The performance ratio (PR) is the ratio of makespan of SOSA and makespan of WBFAS.

PIR: It stands for performance improvement rate. It provides a breakdown of the percentage by which the recommended method (WBFAS) performs better than any other algorithm that is currently in use (SOSA).

$$PIR=((makespan\ (SOSA)-makespan\ (WBFAS))/makespan(WBFAS)\ )*100 \quad (6)$$

Throughput: The term "throughput" refers to the number of tasks that are finished within a specified amount of time.

Failure Ratio: It is the ratio of number of accidents (failures) that occurred during scheduling with WBFAS to the total number of accidents that occurred while using other procedure.

The performance of WBFAS is analyzed with SOSA over PR, FR, throughput, and PIR in Table 2 and Figures 3 to Figure 8.
The analytical values of various results of performance parameters are given in Table 2 which we can relate with graphs.

Table 2. Simulation Results for WBFAS and SOSA

Number of Task	*Makespan (SOSA)*	*Makespan (WBFAS)*	*Performance Ratio (PR)*	*Performance Improvement Rate (PIR)*	*Throughput (SOSA)*	*Throughput (WBFAS)*	*NOF (SOSA)*	*NOF (WBFAS)*	*Failure Ratio (WBFAS)*
20	1177.00	841.44	1.3988	39.8798	0.0581	0.0873	1522.90	1151.60	0.7562
40	1151.50	1054.10	1.0924	9.2401	0.0538	0.0749	3804.30	3113.80	0.8185
60	1535.90	1457.10	1.0541	5.4080	0.0348	0.043	13234.00	11530.00	0.8712
80	2397.80	2082.30	1.1515	15.1515	0.025	0.0282	32370.00	30606.00	0.9455
100	3135.00	2659.70	1.1787	17.8704	0.0231	0.0257	52077.00	47007.00	0.9026

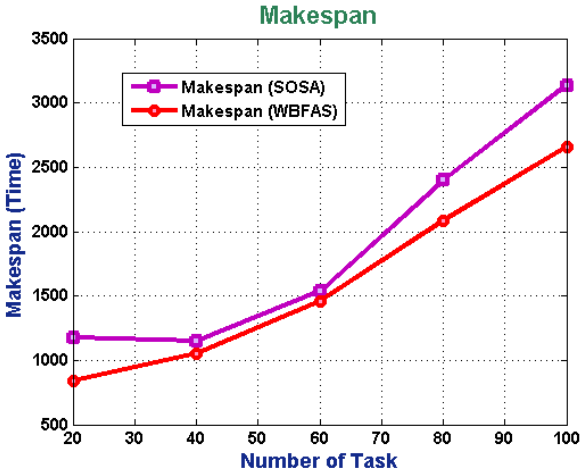

Fig 3. Performance Comparison (Makespan)

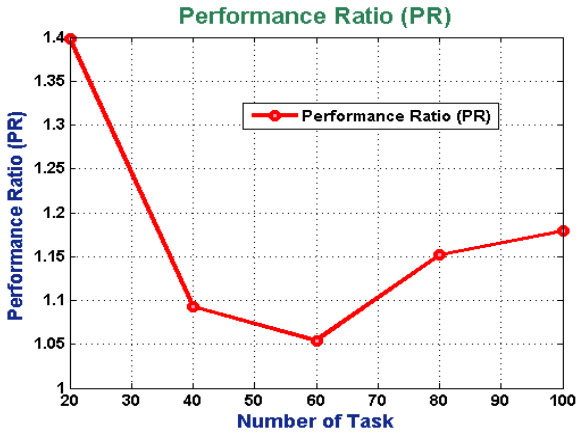

Fig 4. Performance Ratio (PR)

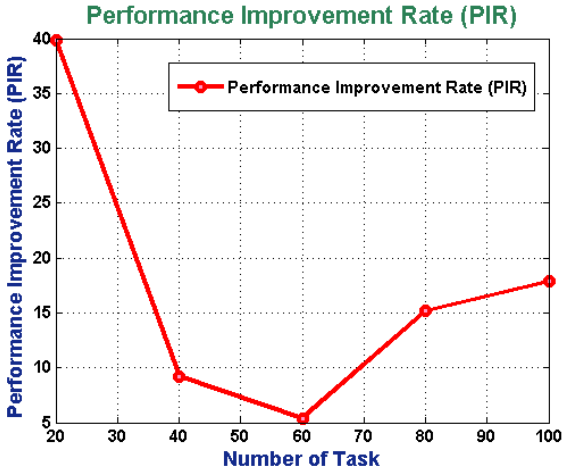

Fig 5. Performance Improvement Rate (PIR)

Figure 3 investigate the execution time (makespan) taken by both the algorithm for performing job, and graph shows that makespan of WBFAS is always less than SOSA and hence performance ratio continuously remains more than 1 (see Figure 4). Both makespan and PR are direct indicators performance improvement by the WBFAS.

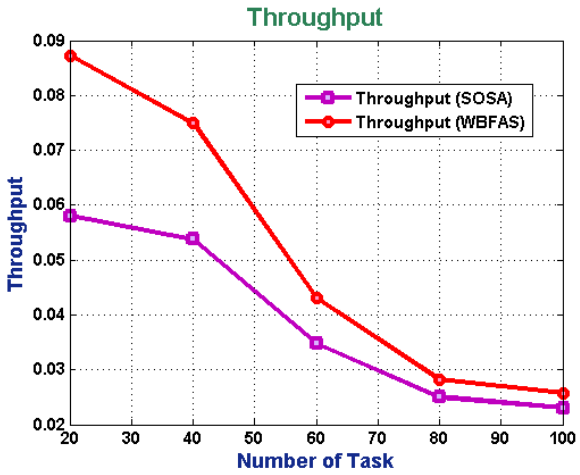

Fig 6. Throughput

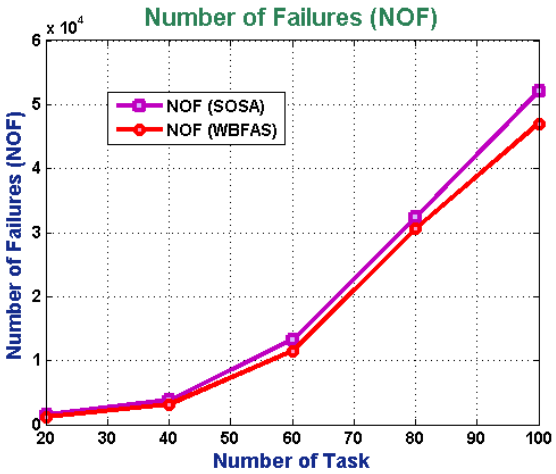

Fig 7. Number of Failures (NOF)

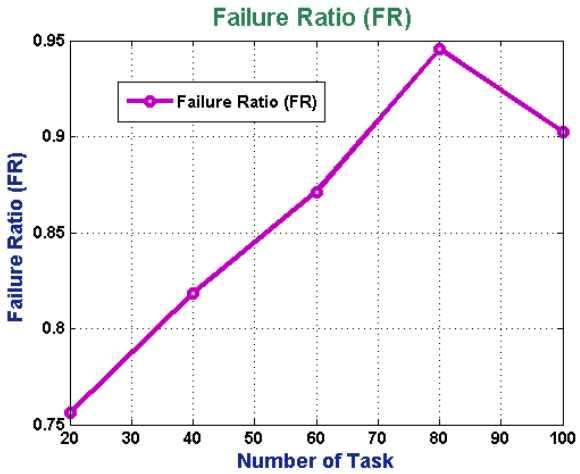

Fig 8. Failure Ratio (FR)

Figure 5 asses the PIR. For instance for 80 task PIR is around 15, it means that WBFAS improved the system performance by 15 % by decreasing the execution time and similarly for 100 task around 18% performance improvement is recorded.

Figure 6 examine the throughput of the system and result depicts that the throughput of WBFAS is always higher than SOSA. It means that WBFAS executes more number of jobs in the same time duration.

Figure 7 compares the number of failures (NOF) and graph shows that WBFAS recoded comparatively lesser NOF than SOSA. Similarly, in Figure 8 failure ratio (FR) always comes out to be less than 1. Lesser NOF means WBFAS increases the reliability of the system.

Hence, simulation results and graphs depicts that WBFAS increases system performance and reliability by reducing execution time and number of failures.

5. Conclusion

To solve the NP-complete scheduling problem in grid computing for dependent task, a workflow based failure-aware scheduling (WBFAS) approach is proposed in this research. The workflow of dependant task is represented with the help of directed acyclic graph (DAG) as explained in methodology section. The proposed approach uses incremental checkpoint approach for fault tolerance and use failure information of nodes for making the scheduling decision. The dependant tasks were assigned a rank based on workflow and then resources allocation was done according to their new calculated computational capacity. The WBFAS is compared with SOSA and simulation results depicts that WBFAS improves the system performance by reducing the makespan and increase system reliability by reducing the number of failures and failure ratio.

Author contributions

Manjeet Singh: Literature, methodology, implementation, result analysis, preparing and editing draft have been done by first author, who is a Ph.D. Scholar. **Dr. Javalkar Dinesh Kumar:** The review and editing of write-up, supervision and administration, has been done by second author, who is the Ph.D. supervisor of the first author.

Conflicts of interest

The authors state that there are no conflicts of interest.

References

[1] M. Baker, R. Buyya, and D. Laforenza, “Grids and Grid technologies for wide-area distributed computing”, Software – Practice and Experience. Vol. 32, No. 15, 2002.

[2] Manjot Kaur Bhatia, “Task Scheduling in Grid Computing: A Review”, Advances in Computational Sciences and Technology ISSN 0973-6107 10(6) (2017) 1707-1714.

[3] H. B. Prajapati, V. A. Shah, “Scheduling in Grid Computing Environment”, 2014 Fourth International Conference on Advanced Computing & Communication Technologies, ISBN:978-1-4799-4910-6, DOI: 10.1109/ACCT.2014.32, (2014).

[4] S. Haider and B. Nazir, “Fault tolerance in computational grids: perspectives, challenges, and issues”, Springer Plus, Vol. 5, pp. 1-20, 2016.

[5] R. Garg and A. K. Singh, “Fault Tolerance in Grid Computing: State of the Art and Open Issues”, International Journal of Computer Science & Engineering Survey (IJCSES), Vol. 2, No. 1, pp. 88-97, 2011.

[6] R. Garg and A. K. Singh, “Fault Tolerant Task Scheduling on Computational Grid Using Checkpointing Under Transient Faults”, Springer,

Arab J Sci Eng, Vol. 39, pp. 8775–8791, 2014.
[7] R. Garg and A. K. Singh. “Adaptive workflow scheduling in grid computing based on dynamic resource availability”, Engineering Science and Technology, an International Journal, Vol. 18, pp. 256-269, 2015.
[8] Yang Zhang, Anirban Mandal, Charles Koelbel and Keith Cooper, "Combined Fault Tolerance and Scheduling Techniques for Workflow Applications on Computational Grids", 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, pp 244-251, 2009, ISBN: 978-0-7695-3622-4/09, DOI 10.1109/CCGRID.2009.59.
[9] A. Iosup, M. Jan, O. Sonmez and D. H. J. Epema, “On the Dynamic Resource Availabilty in Grids”, IEEE 8th Grid Computing Conference, pp. 26-33, 2007.
[10] Zhifeng Yu, Chenjia Wang and Weisong Shi, “Failure-aware workflow scheduling in cluster environments”, Cluster Comput, Vol. 13, pp. 421–434, 2010. DOI 10.1007/s10586-010-0126-7.
[11] Liang Yu, Gang Zhou, Yifei Pu, “An Improved Task Scheduling Algorithm in Grid Computing Environment”, Int. J. Communications, Network and System Sciences, Vol. 4, pp. 227-231, 2011. DOI:10.4236/ijcns.2011.44027.
[12] M. Hemamalini, M.V. Srinath, “State of ART: Task Scheduling Algorithms in Heterogeneous Grid Computing Environment”, Elysium Journal of Engineering Research & Management, Vol. 1, Issue 1, pp. 15-21, 2014.
[13] C. Chandrasekar, V. Manuprasad, “A Review on Scheduling Algorithms for Resource Management in Data Grids”, International Journal of Scientific & Engineering Research, Vol. 6, Issue 5, pp. 1865-1873, 2015.
[14] P. Jiang, Y. Xing, X. Jia, and B. Guo, “Weibull Failure Probability Estimation Based on Zero-Failure Data”, Hindawi Publishing Corporation, Mathematical Problems in Engineering Volume , pp. 1-8, 2015.
[15] Lulu Zhang , Guang Jin, and Yang You, “Reliability Assessment for Very Few Failure Data and Weibull Distribution”, Mathematical Problems in Engineering, Hindawi, Volume 2019, Article ID 8947905, pp. 1-9, 2019. https://doi.org/10.1155/2019/8947905.
[16] Cappello F. et al., “Modeling and tolerating heterogeneous failures in large parallel systems”, In: Proceedings of the SC’2011 International Conference for High Performance Computing, Networking, Storage and Analysis, ACM Press (2011).
[17] Yudan Liu, Raja Nassar, Chokchai Leangsuksun, Nichamon Naksinehaboon, Mihaela Paun, and Stephen L. Scott, “An optimal checkpoint/restart model for a large scale high performance computing system”, In: IEEE International Symposium on Parallel and Distributed Processing (IPDPS 2008), pp. 1–9 (2009).
[18] Bianca Schroeder and Garth A. Gibson, “A large-scale study of failures in high performance computing system”, IEEE Trans. Dependable Secur. Comput. 7(4), 337–350 (2010).
[19] M. Paun, N. Naksinehaboon, and R. Nassar, “Incremental Checkpoint Scheme for Weibull Distribution”, International Journal of Foundations of Computer Science, Oct. 2009.
[20] Manjeet Singh and Javalkar Dinesh Kumar, “Designing and Implementation of Failure-Aware Based Approach for Task Scheduling in Grid Computing”, IJEER 10(3), pp- 651-658, 2022. DOI: 10.37391/IJEER.100339.