International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 10 Issue: 11
DOI: https://doi.org/10.17762/ijritcc.v10i11.5778
***Article Received**: 21 August 2022 **Revised**: 03 October 2022 **Accepted**: 18 October 2022*

DBRS: Directed Acyclic Graph based Reliable Scheduling Approach in Large Scale Computing

Manjeet Singh[*], Dr. Javalkar Dinesh Kumar[2]
[1]Department of Computer Sciences & Engineering,
Lingaya's Vidyapeeth
Faridabad, India
19phcs05w@lingayasvidyapeeth.edu.in
[2]Department of Electronics & Communication Engineering
Lingaya's Vidyapeeth
Faridabad, India

Abstract— In large scale environments, scheduling presents a significant challenge because it is an NP-hard problem. There are basically two types of task in execution- dependent task and independent task. The execution of dependent task must follow a strict order because output of one activity is typically the input of another. In this paper, a reliable fault tolerant approach is proposed for scheduling of dependent task in large scale computing environments. The workflow of dependent task is represented with the help of a DAG (directed acyclic graph). The proposed methodology is evaluated over various parameters by applying it in a large scale computing environment- 'grid computing'. Grid computing is a high performance computing for solving complex, large and data intensive problems in various fields. The result analysis shows that the proposed DAG based reliable scheduling (DBRS) approach increases the performance of system by decreasing the makespan, number of failures and increasing performance improvement ratio (PIR).

Keywords- Large Scale Computing, Reliability; Scheduling; Workflow; DAG- Directed Acyclic Graph; Fault- Tolerance;

I. INTRODUCTION

The term "grid computing" refers to a type of high-performance computing environment that makes it easier to meet the needs of large-scale computations. It encompasses a variety of research challenges, including resource management, work scheduling, information management, and security difficulties. Scheduling of tasks is an essential component of parallel computing, as well as distributed computing and grid computing. In grid computing, the primary objective of task scheduling is to maximize system throughput and performance while simultaneously satisfying the resource requirements of the work [1].

A programme known as Grid scheduler is accountable for controlling the allocation and execution of tasks on appropriate machines [2]. Computing on a grid can be useful in a wide variety of contexts, including the fields of medicine, meteorology, engineering, and research, amongst others. When it comes to research, the most important aspects of grid computing are task scheduling and ensuring fault tolerance [3].

The workflow of dependent task is represented by DAG in grid computing. A workflow is a series of tasks that must be completed in the specified order to achieve the desired end result. In most cases, the term "dependency" refers to the presence of a precedence order within the tasks. This means that a task cannot begin, and in some cases cannot even advance, until its predecessors have been completed. Dependency has a substantial effect on how fault tolerance is achieved [4]. A graph is a collection of vertexes joined by edges. Each node in the graph is in a specific order according to a topological sorting. Each edge runs from a previous edge to a subsequent edge. Topological ordering of graph is another name for DAG. Scheduling, circuit design, and Bayesian networks are some examples of applications of DAG. Figure 1 below shows an example of directed acyclic graph. The vertices of DAG represent the task. The directed edge between two vertices v1→v2 shows that v2 is dependent on v1. The weight on each edge depicts the communication cost between vertices.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 10 Issue: 11
DOI: https://doi.org/10.17762/ijritcc.v10i11.5778
Article Received: 21 August 2022 Revised: 03 October 2022 Accepted: 18 October 2022

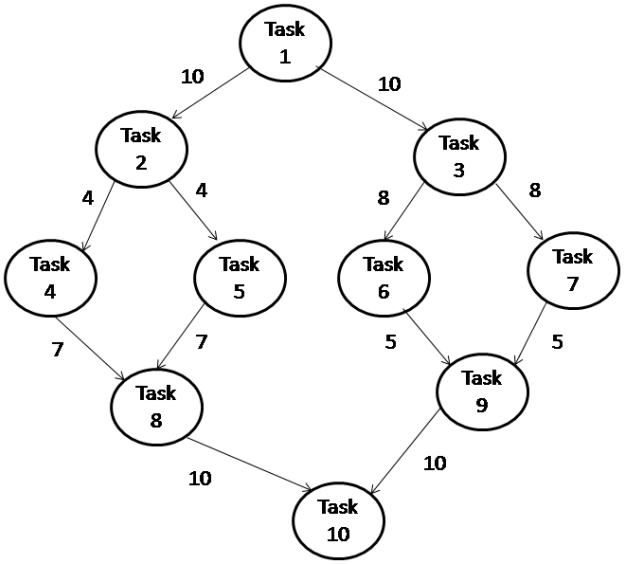

Figure 1. Dependent Task as Directed Acyclic Graph (DAG)

II. Related Work

This section reviewed the numerous tasks scheduling techniques that have been developed by various academicians and researchers. An algorithm for fault-tolerant scheduling of dependent tasks in computational grids was proposed by R. Garg et al. [5]. The method used is Weibull failure distribution and checkpoint rollback recovery method is used to handle failure. R. Garg et al. [6] designed a dependent task scheduling algorithm for computational grid. The workflow was represented with the help of DAG and resource availability was dynamic in nature. The computational analysis show that this method is capable of dealing with fluctuations in available resources and offers complete optimal performance. The simulations were carried out using task graphs generated at random as well as task graphs relating to real world problems.

Y. Zhang et al. [7] make some suggestions for new methods that integrate the fault tolerance strategies with the workflow scheduling algorithms already in place. It designed a HEFT and DAG with over-provisioning algorithm for scheduling with checkpointing methods.

A. Iosup et al. [8] perform an analysis of resource availability of Grid'5000, consisting of more than 2,500 processors. When availability is taken into account, there is an almost 5 percentage point increase in the average utilization of the trace that was studied. A model for the availability of grid resources is proposed based on the findings.

Zhifeng Yu et al. [9] introduced FLAW, a technique for failure-aware workflow scheduling using failure prediction. L. Yu et al. [10] a revised communication inclusion generational scheduling (CIGS) algorithms based on DAG has been demonstrated and found to be effective for grid computing environments.

Grid makes it possible to share, pick, and aggregate geographically dispersed "autonomous" resources. However, users must have the appropriate access permissions in order to access resources on a remote server. The usage of a password authentication technique is one of the most straightforward and practical security mechanisms. Consequently, the security issue is crucial for grid computing. The technique suggested by C. C. Lee et al. [11] is extremely straightforward and effective, it use only a one-way hash function and the server's private key.

H. Sing et al. [12] proposed an efficient resource scheduling algorithm. Prior to dispatching a resource, the scheduling procedure involves a methodical low for determining work-load requirements. User tasks are portrayed as an erratic workload that cloud users send. The objective of solution evaluation is to get the minimum execution time and cost. To save money and effort, it used pheromone-based heuristic data. It was inspired by how scattered ants communicate with one another, utilizing pheromones to determine the quickest, most efficient route.

R. Changan et al. [13] on the basis of adaptive genetic optimization presented a quick information scheduling solution for extensive logistics supply chains. The GA is utilized to find a solution. Experiments demonstrate that the suggested approach can successfully increase scheduling effectiveness and address the issues that currently exist.

M. K. Gourisaria et al. [14] proposed EPTS - an algorithm for energy-saving of resource and distributes the amount of energy consumed by tasks in an equal manner. This is accomplished by pre-empting jobs that have high energy demands with tasks that have lower energy demands.

III. Proposed Model

The proposed scheduling method DBRS uses the DAG workflow, performance and failure parameter of resources for making the scheduling decision. It is considered that communication links may be relied upon completely. The reliability is ensured with the help of full checkpoint fault tolerant mechanism. The time between failures of different resources is modeled as Weibull failure distribution [15, 16, 17-19]. Scale parameter (α) and shape parameter (β) are the two parameters that make up the Weibull failure distribution. When $\beta = 1$, it means that the failure rate is constant over time, $\beta > 1$, it means that it rises over time, and $\beta < 1$ it means that it falls over time. The time between failures in this study is predicated on the Weibull failure distribution with increasing hazard rate.

As the failures are inevitable in grid due to heterogeneity of resources, they consume a big chunk of execution time. So, the concept is to find out the expected wasted time during execution due to failure and recovery from failure. This wasted time information is used to recalculate the resource computing capacity and later scheduling is done such that we

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 10 Issue: 11
DOI: https://doi.org/10.17762/ijritcc.v10i11.5778
***Article Received**: 21 August 2022 **Revised**: 03 October 2022 **Accepted**: 18 October 2022*

can minimize the wasted time due to failures and improve the system performance. The wasted time is calculated with the help of Eq. 1.

$$Wasted\ Time\ during\ Execution\ of\ a\ Task = (Time\ required\ for\ taking\ checkpoint + Time\ require\ for\ re-computation\ of\ portion\ of\ failed\ job + Time\ required\ for\ recovery\ from\ saved\ checkpoint) \quad (1)$$

The expected wasted time for Weibull distribution and full checkpointing mechanism is given by Eq. 2 [20]:

$$Expected\ Wasted\ Time = \int_0^{\infty} \left[O_F \int_0^t n(t).dt + \frac{k}{n(t)} + R_F\right].f(t).dt \quad (2)$$

Where, O_F -denotes the time required for saving checkpoint
R_F -denotes the time required for recovery
f(t) -is a PDF (probability density function)
k -is a coefficient of recomputing time
n(t) -is checkpoint function, given by Eq. 3 [20]

$$n(t) = \sqrt{\frac{k}{O_F} \cdot \frac{f(t)}{1-F(t)}} \quad (3)$$

Where, F(t) is CFD (cumulative distribution function) .

$$f(t) = \left(\frac{\beta}{\alpha}\right) \cdot \left(\frac{t}{\alpha}\right)^{\beta-1} \cdot e^{-(t/\alpha)^{\beta}} \quad (4)$$

$$F(t) = 1 - e^{-(t/\alpha)^{\beta}} \quad (5)$$

Using Eq. 4 and Eq. 5, Eq. 3 can be written as Eq. 6.

$$n(t) = \sqrt{\frac{k}{O_F}} \cdot \left(\frac{t}{\alpha}\right)^{\frac{\beta-1}{2}} \cdot \sqrt{\frac{\beta}{\alpha}} \quad (6)$$

The Figure 2 below show the flowchart of the proposed DAG based reliable scheduling (DBRS) approach.

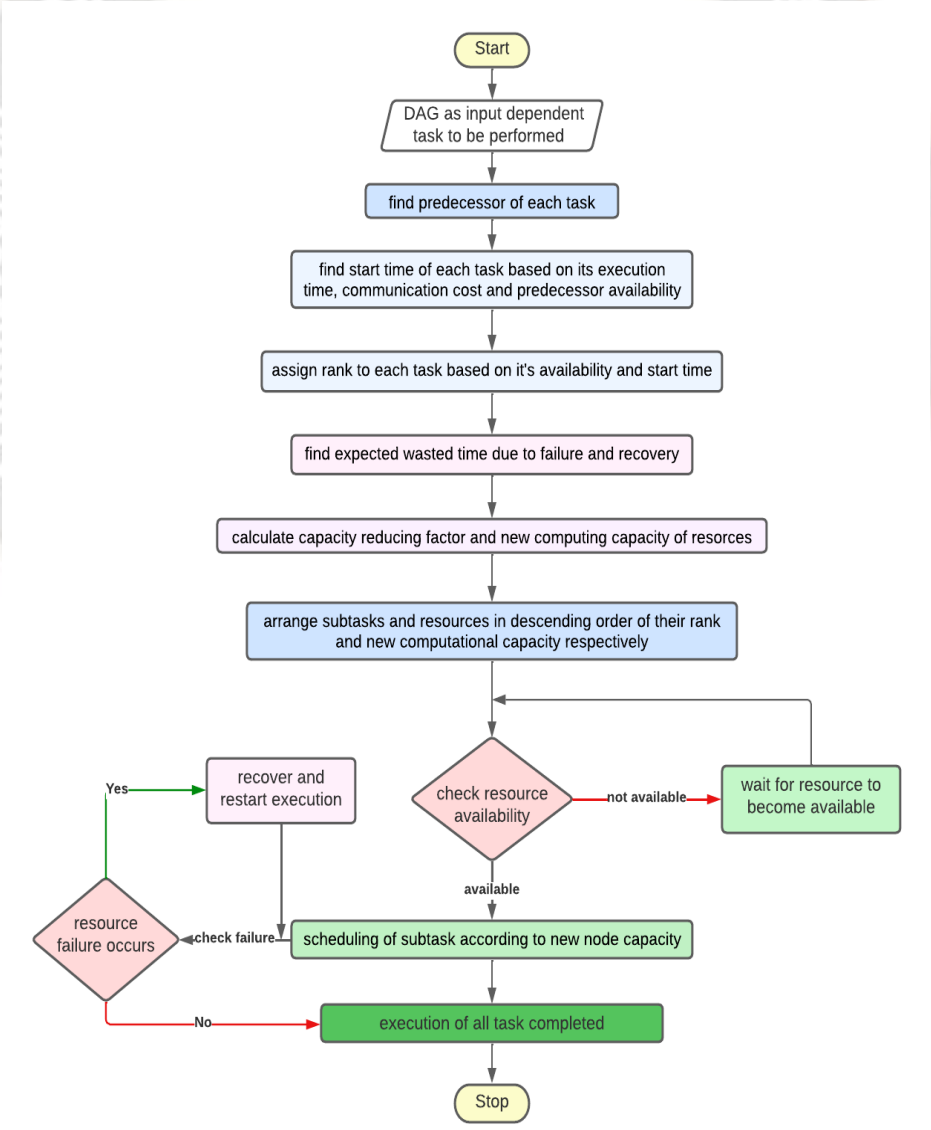

Figure 2. Flowchart for DAG based Reliable Scheduling Approach

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 10 Issue: 11
DOI: https://doi.org/10.17762/ijritcc.v10i11.5778
***Article Received**: 21 August 2022 **Revised**: 03 October 2022 **Accepted**: 18 October 2022*

The proposed DAG based reliable scheduling approach (DBRS), schedule the dependent task in large scale computing environment like grid computing. The job to be performed is given as input to the system in the form of a directed acyclic graph (DAG). The nodes/vertex of the graph indicates the subtasks among the job. The direction of an edge from one node to another node in DAG represents the dependency of task. For example if there is an edge from node v1 to v2, it indicates that v2 is dependent on v1 i.e. the task v2 can't start its execution until the task v1 finishes its execution and its result is available to task v2. The weight of an edge in DAG represents the communication cost between two vertices. Values within the vertex represent the size of subtask.

The procedures for the suggested method are outlined as follows:

Step 1: Input the job to be performed as DAG with workload of vertices and communication cost of edges.

Step 2: Based on successor of each vertex in DAG, find out predecessor subtask of each subtask.

Step 3: Calculate the average execution time of each subtask by computing the execution time of each subtask on each resource.

Step 4: Find arrival/start time of a subtask based on its execution time, communication cost and predecessor availability.

Step 5: Assign rank to each subtask based on its arrival time.

Step 6: Find expected wasted time of the system due to failure, recovery and fault tolerant mechanism.

Step 7: Calculate capacity reducing factor based on expected wasted time of the system using Eq. 2.

Step 8: Calculate actual effective computing capacity of resources based on capacity reducing factor.

Step 9: Arrange task in descending order of their rank.

Step 10: Arrange resources in decreasing order of their computational capacity.

Step 11: Schedule task according to rank and new capacity.

Step 12: Check resource availability

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

Step 13: End;

IV. Result and Discussion

To check effectiveness of proposed algorithm DBRS is compared with other algorithm over various evaluation parameters. A model is created with eight computing nodes/resources. We took into account the shape parameter (β) with increasing failure rate, which ranges from 1.8 to 3.6. The scale parameter's value is α =20. The recovery time and checkpoint storage cost are 2 minutes. The value of re-computing time coefficients is 0.5. Grid application run with a variable number of dependent tasks (DAG) ranging from [5, 25]. Various parameters are referred from [8, 17-18].

The performance of the proposed DAG-based reliable scheduling (DBRS) algorithm is compared to that of the Speed-only approach (SOSA). The SOSA algorithm only takes into account resource performance characteristics for scheduling tasks. Performance of DBRS is assessed using the metrics listed below [20].

PIR: Performance Improvement Rate reveals the percentage difference between the DBRS method and the competing SOSA method.

$$PIR(\%)=\left(\frac{Makespan\ (SOSA)-Makespan\ (DBRS)}{Makespan\ (DBRS)}\right)\times 100 \qquad (7)$$

Failure Ratio (FR): It is the ratio of overall crashes using the suggested way to overall crashes using the current method. The proposed failure-aware scheduling strategy will work better if the value of FR is smaller than 1.

Throughput: That's the number of tasks finished in a certain amount of time. It demonstrates the number of jobs that were completed or processed in the allotted amount of time.

Table 1 below lists the various efficiency parameters for failure ratio, throughput, and PIR.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 10 Issue: 11
DOI: https://doi.org/10.17762/ijritcc.v10i11.5778
***Article Received**: 21 August 2022 **Revised**: 03 October 2022 **Accepted**: 18 October 2022*

TABLE I. SIMULATION RESULTS FOR DBRS AND SOSA OVER VARIOUS EVALUATION PARAMETERS

No. of Task	Makespan (SOSA)	Makespan (DBRS)	PIR (DBRS)	Throughput (SOSA)	Throughput (DBRS)	NOF (SOSA)	NOF (DBRS)	Failure Ratio (DBRS)
5	375.2548	358.1493	4.78	0.0447	0.077	238.4006	179.878	0.7545
10	437.2227	362.9187	20.47	0.0536	0.0711	441.169	352.7856	0.7997
15	616.855	601.1254	2.62	0.0467	0.0622	887.1631	660.7692	0.7448
20	978.6657	828.1063	18.18	0.0403	0.052	1578.90	1104.80	0.6997
25	1253.80	1057.10	18.61	0.0354	0.045	2606.70	1705.80	0.6544

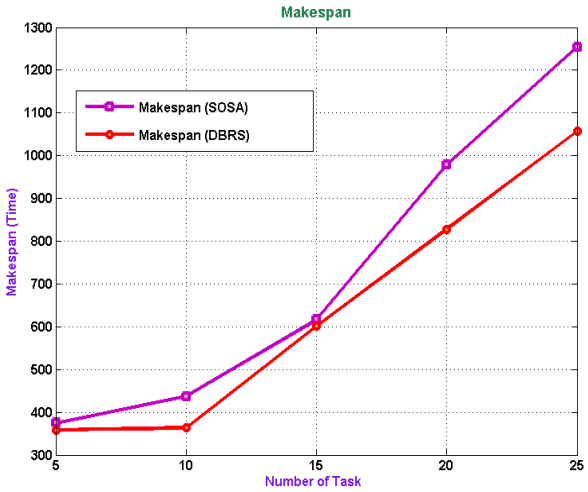

Figure 3. Makespan Comparison

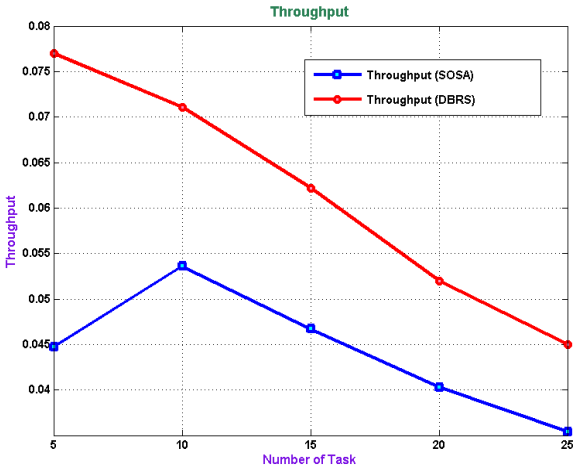

Figure 5. Throughput

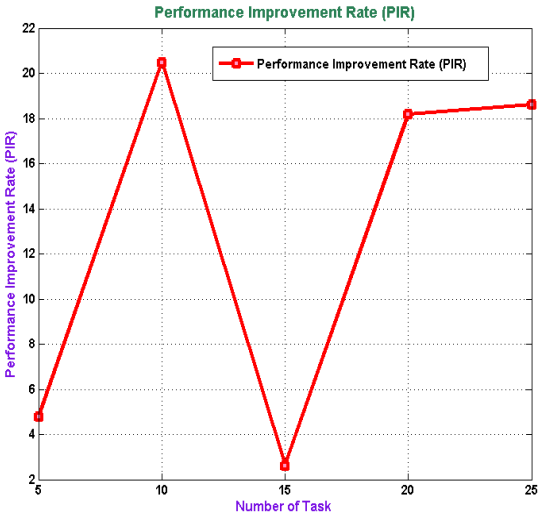

Figure 4. Performance Improvement Rate (PIR)

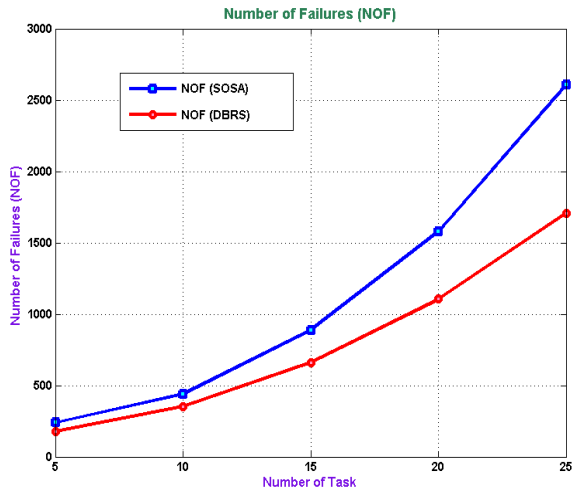

Figure 6. Number of Failures (NOF) Comparison

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 10 Issue: 11
DOI: https://doi.org/10.17762/ijritcc.v10i11.5778
***Article Received**: 21 August 2022 **Revised**: 03 October 2022 **Accepted**: 18 October 2022*

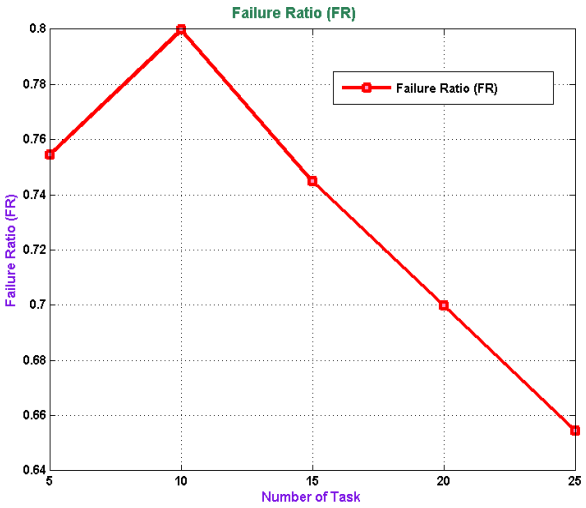

Figure 7. Failure Ratio

Table 1, Figure 3 and Figure 4 examine the makespan and PIR of the proposed approach DBRS over SOSA. From the mathematical results and graph it can be seen very clearly that DBRS always achieve lesser makespan, which is a direct indicator for performance improvement of the system. Lesser makespan means DBRS take less time to execute the job and hence is a faster method. For instance in Figure 4, for 10 tasks PIR is around 20, it means that DBRS is 20% faster than the SOSA method.

Figure 5 investigate throughput of DBRS over SOSA. Simulation values and graph pointed out that throughput of DBRS always higher than SOSA, which means that DBRS compute more number of tasks in the same timeframe and hence improve the system performance.

Figure 6 and Figure 7 asses the number of failures (NOF) and failure ratio (FR). The comparison of analytical data from Table 1 and these figures shows that NOF in DBRS are less than SOSA and hence FR continuously comes out to be less than 1. Reduced number of failures and FR<1, indicates that the proposed method DBRS improved reliable of the system.

V. Conclusion

Researchers have been interested in the scheduling of dependent tasks application since the inception of Grid computing because of its NP-Complete nature. Scheduling in a grid environment determines how to assign tasks to the resources that are available. When tasks have dependencies and resources are heterogeneous, grid scheduling become more difficult. Minimizing make-span is the primary goal of dependent task scheduling. In this research paper, a DAG-based reliable scheduling model is proposed with the goal to increasing the system's reliability while simultaneously reducing the amount of time required for its execution. The findings of the simulation indicate that the fault-tolerant scheduling algorithm DBRS increased the system performance by around 20% by reducing makespan and decreased the amount of failures and the failure rate, which increased the system's reliability. DBRS also improved the overall throughput of the system.

References

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

[2] 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).

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

[4] 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.

[5] 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.

[6] 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.

[7] 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

[8] 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.

[9] 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.

[10] 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.

[11] Cheng-Chi Lee, Hsien-Ju Ko & Shun-Der Chen, An improved simple user authentication scheme for grid computing, Journal of Discrete Mathematical Sciences and Cryptography, Vol. 15(2-3), pp. 113-124, 2012. DOI: 10.1080/09720529.2012.10698368

[12] Ren Changan, Jinguo Zhao & Liping Chen, A fast information scheduling algorithm for large scale logistics supply chain, Journal of Discrete Mathematical Sciences

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 10 Issue: 11
DOI: https://doi.org/10.17762/ijritcc.v10i11.5778
***Article Received**: 21 August 2022 **Revised**: 03 October 2022 **Accepted**: 18 October 2022*

and Cryptography, Vol. 20(6-7), pp. 1459-1463, 2017. DOI:10.1080/09720529.2017.1392463

[13] Harvinder Singh, Anshu Bhasin & Parag Kaveri, SECURE : Efficient resource scheduling by swarm in cloud computing, Journal of Discrete Mathematical Sciences and Cryptography, Vol. 22(2), pp. 127-137, 2019. DOI: 10.1080/09720529.2019.1576334.

[14] Mahendra Kumar Gourisaria, Pabitra Mohan Khilar & Sudhansu Shekhar Patra, EPTS: Energy-saving pre-emptive task scheduling for homogeneous cloud systems, Journal of Discrete Mathematical Sciences and Cryptography, Vol. 24(8), pp. 2415-2441, 2021. DOI: 10.1080/09720529.2021.2016191.

[15] 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.

[16] Lulu Zhang , Guang Jin, and Yang You, "Reliability Assessment for Very Few Failure Data and Weibull Distribution", Mathematical Problems in Engineering, Hindawi, Vol. 2019, pp. 1-9, 2019. https://doi.org/10.1155/2019/8947905.

[17] Cappello, F.: 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)

[18] Liu, Y.; Nassar, N.; Leangsuksun, C.; Nichamon, N.; Paun, M.; Scott, S.: 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)

[19] Schroeder, B.;Gibson, G.A.:Alarge-scale study of failures in highperformance computing system. IEEE Trans. Dependable Secur. Comput. 7(4), 337–350 (2010).

[20] Manjeet Singh and Javalkar Dinesh Kumar (2022), Designing and Implementation of Failure-Aware Based Approach for Task Scheduling in Grid Computing. IJEER 10(3), 651-658. DOI: 10.37391/IJEER.100339.