Bulletin of Electrical Engineering and Informatics
Vol. 13, No. 1, February 2024, pp. 396~404
ISSN: 2302-9285, DOI: 10.11591/eei.v13i1.5413

Customizing the minimum number of replicas for achieving fault tolerance in a cloud/grid environment

Mahdi S. Almhanna[1], Tariq A. Murshedi[1], Firas Sabah Al-Turaihi[1], Rafah M. Almuttairi[2]
[1]Department of Information Networks, College of Information Technology, University of Babylon, Babylon, Iraq
[2]Collage of Information Technology Engineering, Al-Zahraa University for Women, Karbala, Iraq

Article Info

Article history:

Received Dec 4, 2022
Revised May 25, 2023
Accepted Jun 4, 2023

Keywords:

Cloud computing
Distributed systems
Fault tolerant
Grid computing
Replication

ABSTRACT

Networks consist of numerous resources; it is crucial not to overlook fault tolerance and consider it during planning. This is because errors during implementation can result in wasted time and effort, thereby squandering these resources. One solution to address this issue effectively is to implement the task on multiple resources to minimize the occurrence of failed tasks. However, employing an unspecified or fixed number of resources can lead to the depletion of network resources and the overall failure of the network. Replication plays a pivotal role in enhancing data availability in distributed systems. By storing data in multiple locations, users can still access it even if some copies are unavailable due to site failure. Many replication-based algorithms utilize a predetermined number of iterations per function, which may consume excessive network resources, even if the ongoing task does not require such abundant resources. This paper proposes task replication as a viable mechanism for an efficient and fault-tolerant scheduling system. We introduce an algorithm that dynamically selects the optimal and minimal number of replicas based on the network's failure history. This approach aims to minimize the failure rate during task execution.

Corresponding Author:

Mahdi S. Almhanna
Department of Information Networks, College of Information Technology, University of Babylon
Babylon, Iraq
Email: mahdi.almhanna@uobabylon.edu.iq

1. INTRODUCTION

The utilization of computing enables organizations to integrate geographically dispersed resources from different administrative regions into a unified system. This consolidation facilitates the resolution of large-scale problems across scientific, human, and social domains [1]–[5]. These resources encompass diverse components such as computers, storage, peripherals, and applications. A middleware layer must be implemented to deliver essential services to users due to the heterogeneous and dynamically changing nature of network resources.

Network computing environments are prone to various failures and outages, primarily attributed to the abnormal characteristics of the network infrastructure. Failures can manifest in different forms [6], [7], including computer failures, connection issues, network bottlenecks, excessive power consumption, software malfunctions due to workload, potential software deletions, and other factors stemming from the diversity of networks and applications. Ensuring fault tolerance is crucial for maintaining continuous network operation and delivering reliable services.

In essence, reliable applications within the network should be capable of automatically mitigating failures with minimal losses, without significantly compromising performance and quality of service (QoS)

[8], [9]. Put simply, the network should possess the ability to minimize and overcome failures, allowing uninterrupted functionality. Fault tolerance in the network empowers it to continue operating even in the presence of significant mistakes or failures, without disrupting overall functionality.

Also, handling failure may be through scheduling strategies in resource scheduling. If it is prior to scheduling resources, it is called proactive orientation, otherwise, it is called post-active [10], [11]. This approach is easier in terms of implementation than the previous one (proactive) because it uses job monitoring techniques, while the other method works through probabilities and this requires more information about network resources. If the method used is proactive, such as replication, then all decisions made to handle failure must be made before the task is started, thus reducing the probability of failure and increasing productivity. Table 1 illustrated some examples of fault tolerance.

Table 1. Examples of fault tolerance

System	Fault	Tolerance method
Storage server	Storage device	Mirroring technique
Networking	Network card	Replica (backup) card
Computer systems	Power	Ups
Distributed system	Overloading	Migration
Program languages	Ambiguity data	Exception handling

In grids, all proposed mechanisms that deal with the fault-tolerant are divided into three categories [12]. The first category is called task replication [9], [13]. In this category, the same task is repeated to implement it on many independent resources to protect the task from one failure point. The second is called checkpoint redundancy [14]. The status of the running function is saved to stabilize stable. This condition can be used later if any error occurs to persistent storage at different points instead of re-execution, in other words, the implementation is resumed starting from the last stored point and not from the beginning. The last is called adaptive. Both checkpoint and symmetrical copies are used to achieve the task. The adaptive approach greatly improves the performance of the distributed system. It achieves throughput and fault tolerance with optimal parameters. The first category has been considered in this paper, to establish an error-resistant proactive scheduling system and specify the minimum number of replicas needed for each task.

- Related work

Detecting faults and predicting them before they occur is a primary goal during system design such as preventing or avoiding the error that causes the deadlock problem as well as recovery strategies. These strategies are implemented through replication strategy or through some improvement or both. Failure at one source is more likely than failure at multiple sources simultaneously. Therefore, re-implementing jobs from the beginning can be avoided using the replication. Thus, there is no waste of effort or time in the process progression, since a failure in one of these sources does not lead to a network breakdown and it can continue to provide services.

During the development of work on the implementation of the task, the system stores all the information and data for the task at that point in the work, and the system continues to make a backup for all the cases of the task at each new development and within a specific strategy. This point is called checkpoint. If the work fails somewhere, the system will work on retrieving the stored information and resume work from that point and not from the beginning, which saves effort and time. The main problem here is the possibility of repeating by mistake, repeatedly or continuously, which causes a bigger problem. In this research, the replication mechanism will be used by limiting the number of resources that will be exploited in replicas to the lowest possible extent in order to take advantage of time and reduce system expenses.

Abawajy [15] suggested a distributing algorithm for scheduling that combines both scheduling and repetition functionality. The idea behind this algorithm is to divide the network into a group of small networks so that each small network consists of a group of sites and each website has a scheduler for that site. Each scheduling manager, in turn, backs up another manager's scheduling. This algorithm supports each user with a fixed specific number of replicas.

Srinivasa *et al.* [16] with each gesture within the network propose a middleware to replicate, there is a copy of the task in each node, and through the TCP/IP protocol the communication between these nodes takes place. Jiang and Zhou [17] suggested a fault-tolerant algorithm for scheduling jobs by matching the resource's trust level and the user's security, the number of copies will be determined according to the security level of the network, which is variable. Chtepen *et al.* [18] introduced a heuristic schedule that relies on replicating functions and rearranging unsuccessful tasks using real-time network state information rather than relying on scheduled job data.

- The aim of the proposed work

The network resources are dynamic and heterogeneous, which means the possibility of failures in the network environment is high, thus more time is required to carry out these functions [19], which causes the network's failure. The fact that system will consume more time to search for other resources suitable for carrying out these tasks in the event of the failure of the available resources. Most of the algorithms that depend on symmetrical copies use a fixed number of identical versions [20]. This means excessive use of resources for the same task, causing the network to be busy with the minimum number of tasks to be performed. In this work, we suggested an algorithm, to determine the number of replicas that are not fixed for all jobs offered but are variable according to the type of tasks to be performed.

2. METHOD

2.1. Backup resources selection

The backup resources will be selected upon indicating the replicas needed for the jobs. Consequently, even if one resource fails, the network can still complete the job by utilizing a surrogate resource. The selection of these resources in the research paper under consideration is based on factors such as response time and resource load. The resource information server (RIS) is responsible for storing the historical data of resource loads, which is defined as (1):

$$LHj = \frac{TC}{Sj} \tag{1}$$

where LH is load history, TC is the instructions acts completed by resource j, and Sj is the rapidity of the resource j measured in seconds for a million instruction operations.

2.2. Grid scheduling system

Architectural engineering for the grid scheduler (GS) assumes a group of units that include: the user interface enables users to send their tasks to the network, then the tasks received from users are directed to the available network resources [21]–[25], one of these resources is called the RIS which collects information about the ability of other resources such as the size of the memory and information about the central processing unit. Through this information, the appropriate decision is taken about the scheduling by GS. GS also provides another reliable service about the fault-tolerant structure through the adoption of another component called the fault handler (FH) and during it, cases of failure that may occur in the system are handled. As shown in Figure 1, the assumed scheduling of resources spread in many different geographical regions is managed by a central management unit. It is well known that the behavior of resources towards failure is different between one resource and another, therefore there must be a processor with the ability to deal with errors in the event of occurrence. If the result is outside the expectations, this means that the failure has happened, so, the information is stored about this failure in resource services (RIS) to benefit from it when implementing the next task.

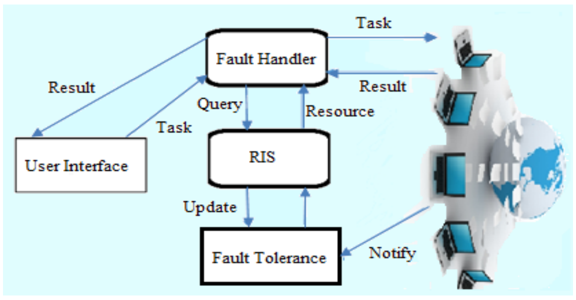


Figure 1. Proposed system architecture

2.3. Proposed system

By applying proactive scheduling, the proposed system tries to avoid failures. Also, assuming such failure, the system will reduce the effect of this failure. This is done by creating many replicas on a different set of resources and implementing them at the same time. Therefore, if one of the resources fails, this will not affect the implementation of the tasks in the rest of the other resources and remain in the event of continuous implementation without delay. Upon the completion of the implementation of any replica, all other versions are terminated, and the network resources are edited from it. The minimum number of replicas needed for each task to be completed is determined by the system depending on the knowledge of the inclination of those resources to fail. As a result, this will enable the system to reduce the effects of failure on the network. After that, the system is based on choosing a good group of resources for the purpose of carrying out the task on it by relying on the time of response to these sources, which is the sum of the necessary transferring time for the task from the scheduler to the source, the time of waiting in the queue, the time of implementation and the time to transfer the result from the source to the scheduler.

The job scheduling process involves selecting a job from the job queue, considering the user service quality desired. Then, the server of resource information is consulted to obtain a suitable list of resources that meet the user's QoS criteria [26]. The role of the RIS is to provide a resource list along with their estimated response times for task completion. Subsequently, the scheduler arranges this list based on the response times of each server. The highest-ranked server is selected as an essential server for executing the function. However, there is a possibility that this primary resource may encounter a failure in carrying out the task. To address this, during the replication phase, the system will elect certain resources from the available roster to act as duplicates of the task. These resources are known as backup or reserve resources. Improving performance can be improved in two ways:

a. By carrying out a job on more than one resource at one time, the time to complete the implementation of the task on the first resource is the time of response. It is not possible to fix the response time fixedly, maybe a variable according to the loads and requests for this server at the time, as well as on the condition of the network and the capacity of the server, thus implementing the task on multiple resources that may improve the regime's response time.
b. The implementation of repeated functions can help deal with failure. It is sufficient to complete the implementation of a single replica to finish the task implementation, therefore the effect of failure can be reduced which may occur if the implementation is on only one replica.

The process of determining the number of replicas is very important, because the increase in the number may reduce the failure to end the task significantly, but at the same time, it will cause consumption of the system resources and the increase in response time (more time for response). On the other hand, if the number of replicas is inappropriate, this may cause the task not to implement. Therefore, must the choice in the number of identical replicas is proportional to the above both cases so that the possibility of implementing the task is high and the effect on the stability of the network is less than possible.

2.4. Adaptive job replication

The proposed algorithm determines the optimal number of replicas which is not constant for all the tasks but fits with the inclination of resources to fail. This proportion is expulsive, the higher the number of source failures, the greater the need for more replicas, and vice versa, the fewer source failures, the less need for more replicas. Consequently, the optimal number of replicas can be deduced by relying on the number of times the failure of the resources, which can be calculated based on the history. This number will be variable according to the type of job assigned to that resource.

Let's suppose Nf represents the count of resource failures in executing its assigned tasks and Ns denotes the count of successful completions by the resource. Whenever the resource fails to accomplish its mission, the value of Nf increments by one and the task originally assigned to this resource is reassigned to another appropriate resource within the network. Conversely, if the resource completes its mission, the value of Ns increases by one. Thus, the inclination to fail FTj for resources can be represented as (2):

$$FTj = \frac{Nf}{Ns+Nf} * 100\% \tag{2}$$

Thus, the possible success of the mission's implementation for resource j can be as (3):

$$Bj = 1 - FTj \tag{3}$$

Assuming that the resources R_1, R_2, ..., R_N are dedicated to task j, then the inclination rate for failure in these resources as in (4):

$$FTn = \frac{\sum_{j=1}^{N} FTj}{N} * 100\% \tag{4}$$

The number of replicas of the task, k, was determined to be commensurate with the value of FTn. The minimum number of replicas should be at least one, the number of resources available and suitable for the task should not exceed N. Accordingly, the highest limit of the number of replicas will be N. Thus, the possible success of the mission's implementation for all resources combined can be as (5):

$$Bn = 1 - FTn \tag{5}$$

2.5. The algorithm

Algorithm 1 named optimal resource allocation and backup strategy algorithm (ORABS) employed to decide the number of replicas for every submitted task involves a comparison between the values of "Bj" and "Bn" starting with the first resource on the dedicated list.

If "Bj ≥ Bn" an additional replica is added. The new list "Bt=Bt+Bj",

The algorithm stops if it is "Bt > Bj." The steps of the algorithm as shown:

Algorithm 1 : ORABS.

```
For each task submitted by the user
{
Receive tasks from the User;
From a Fault handler, request "FT" for all resources
From Resource Information Server, request "LH" for all resources.
Calculate FTj, Bj, FTn, and Bn, for all resources in the grid. such that, if Bj > Bn, then add resource j to the list otherwise no action.
Send packets to the list of servers in step 5 to calculate the RTT
Choose a list of servers with the highest response time from the Resource Information Server;
Arrange the resources in an upward manner and according to the time of responding to these sources;
Calculate the average of the probability of successful resources
Count the number of replicas of the tasks.
Defines backup resources.
}
```

3. CASE STUDY

Assume that we have 50 resources R1, R2……...R50. First, calculate the LH using (1) for each server and neglect the servers that have more than medium. CPU utilization ≤66% and memory utilization ≤62% in [24]. Second, calculate FTj, Bj, FTn, and Bn, for all resources in the grid. Such that, if Bj>Bn, then add resource j to the list otherwise no action, do the procedure to all the 50 resources (if none of them are neglected by the first step, otherwise just for those remaining on the list). Third, suppose the number of resources remaining in the list is 20 resources, R1, R2......... R20, send a ping to all resources in the list and calculate the RTT. Table 2 represents the ping amount of these 20 servers measured in milliseconds. The total values of RTT will be 3317 ns for all resources, so the average will be 3317/20=165.85 ms. Forth, excluding all resources that have more value than average. The result is shown in Table 3.

Table 2. RTT in ms for 20 servers

R,1	R,2	R,3	R,4	R,5	R,6	R,7	R,8	R,9	R,10	R,11	R,12	R,13	R,14	R,15	R,16	R,17	R,18	R,19	R,20
169	168	170	188	180	210	200	190	150	130	135	230	199	290	110	108	102	149	139	100

Table 3. RTT for remaining servers

R,9	R,10	R,11	R,15	R,16	R,17	R,18	R,19	R,20
150 ms	130 ms	135 ms	110 ms	108 ms	102 ms	149 ms	139 ms	100 ms

The tendency to failure is calculated using (2) for all the remaining resources, suppose the tendency to failure of these resources is as in Table 4. Calculate the success probability of the above resources using (5). The success probability of the resources is presented in Table 5. Then, calculate the average of the probability of success for them. The total values probability will be 8.21 for all remaining resources, so the average will be 8.21/9=0.91. Excluding all resources that have less value than average and arranging the table ascending. Therefore, the sources in Table 6 will represent the resources that we can use for the purpose of replication.

Table 4. FT for 20 servers

R,9	R,10	R,11	R,15	R,16	R,17	R,18	R,19	R,20
10%	5%	7%	4%	15%	20%	9%	5%	4%

Table 5. Success probability of 9 servers

R,9	R,10	R,11	R,15	R,16	R,17	R,18	R,19	R,20
0.9	0.95	0.93	0.96	0.85	0.8	0.91	0.95	0.96

Table 6. Success probability of 6 servers

R,20	R,15	R,19	R,10	R,11	R,18
0.96	0.96	0.95	0.95	0.93	0.91

It is noted that each of the resources shown in Table 6 represents a high probability of success in implementing the task. Accordingly, only three resources can be satisfied to represent the resources of the replication, and the first three are the best of these resources, due to the high possibility of carrying out this task. The possibility of achieving the task increases in the event of increased resources, but this option is not good, because it causes resource consumption, network fall, and instability of the network [27]. Figure 2 represents the response time of all resources before any of them are excluded, while Figure 3 represents those with a higher response time.

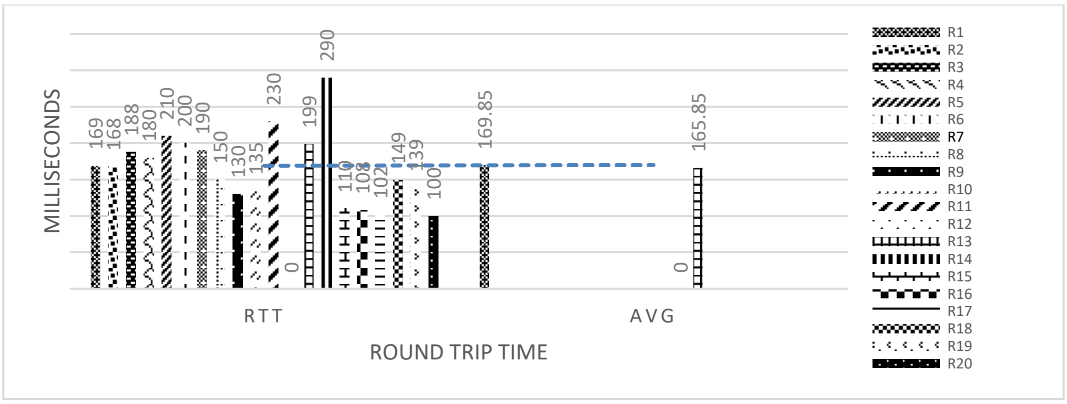


Figure 2. Round trip time for all resources

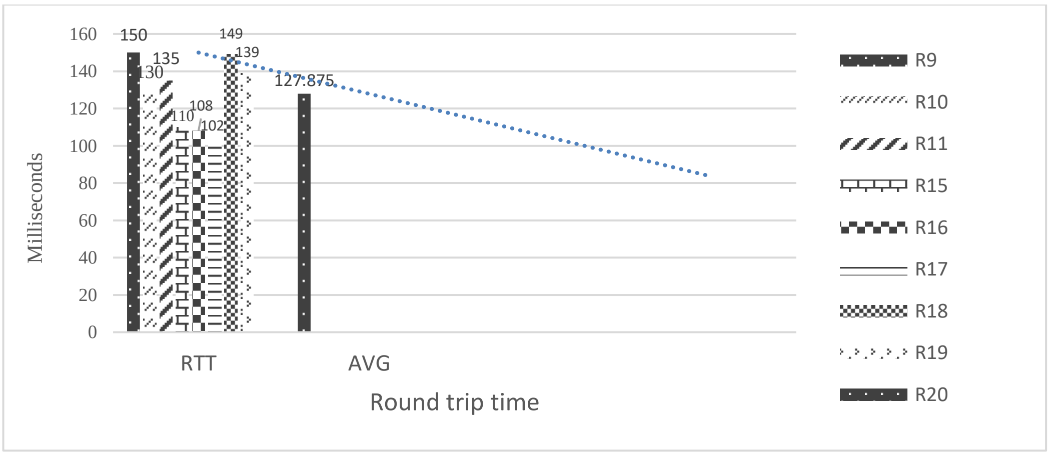


Figure 3. Excluding all resources that have more value than average

4. RESULTS AND DISCUSSION

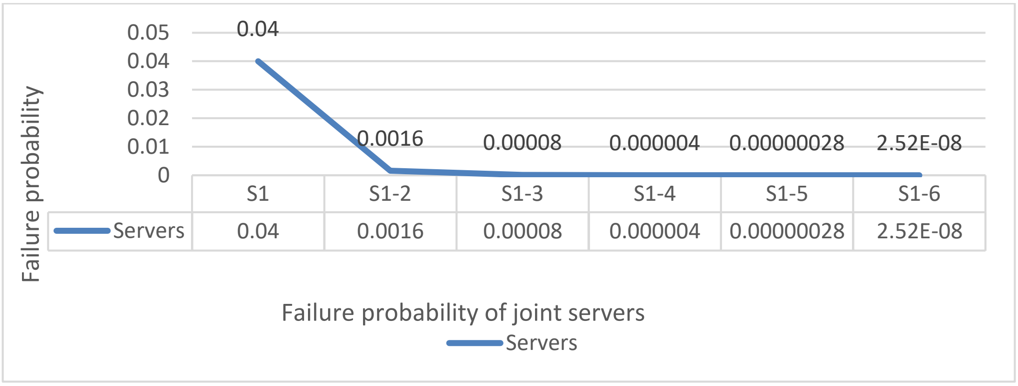
Independent events refer to occurrences that do not influence each other. When event Q is considered independent of event K, it means that the probability of event Q happening is unaffected by the occurrence of event K. If Q and K are independent events in a random experiment, the probability of both events occurring simultaneously, denoted as P(Q⋂K), can be calculated by multiplying their individual probabilities, represented as P(Q) and P(K) in (6):

$$P(Q \cap K) = P(Q) * P(K) \qquad (6)$$

In the case of multiple independent events, let's say Q_1, Q_2, ..., Qn, associated with a random experiment. The probability of all these events happening simultaneously, represented as P(Q1⋂Q2⋂Q3⋯⋂Qn), can be calculated by multiplying the individual probabilities of each event.

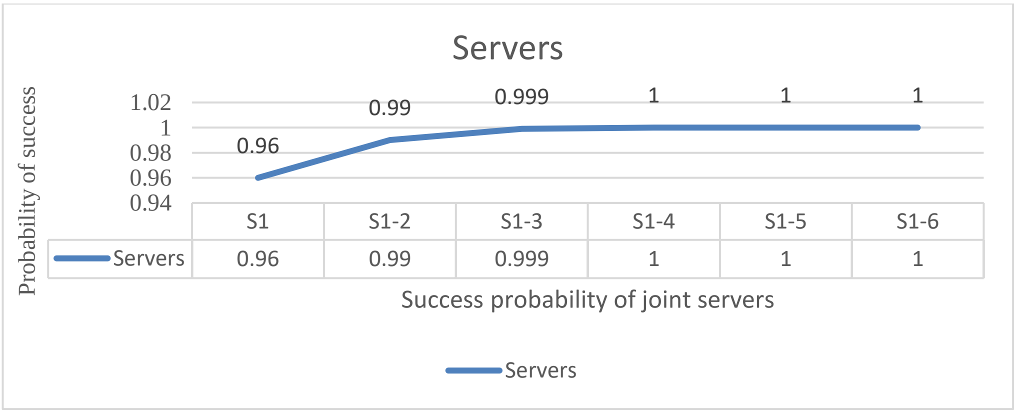
$$P(Q1 \cap Q2 \cap Q3 \cdots \cap Qn) = P(Q1) * P(Q2) * P(Q3) * \ldots * P(Qn) \qquad (7)$$

So, in the case of the first two servers are work, the probability of their failure together is (0.0016) using (6), since the probability of failure of each of them is equal to 0.04 (probability of success is 0.96 for each), which means that their failure rate together will be 0.04*0.04=0.0016. If the first 3 servers work together, the probability that all 3 servers will fail is 0.00008, therefore, their failure rate together will be 0.04*0.04*0.05=0.00008 using (7). From what has been explained above, we note that as the number of servers increases, this will reduce the probability of failure so that it approaches zero, as the failure rate in the case of using only two servers was 0.0016, and when adding another server, the failure rate decreased to 0.00008, and 0.000004 if using 4 servers, and 0.00000028, 0.0000000252 if we use five and six servers respectively. So, Figure 4 represents the failure probability of joint servers, while Figure 5 represents the probability of their success.



	S1	S1-2	S1-3	S1-4	S1-5	S1-6
Servers	0.04	0.0016	0.00008	0.000004	0.00000028	2.52E-08

Figure 4. Failure probability of joint servers



	S1	S1-2	S1-3	S1-4	S1-5	S1-6
Servers	0.96	0.99	0.999	1	1	1

Figure 5. Success probability of joint servers

By applying (5) we can deduce the probability of successful execution of the task for all servers. From Figure 5 it turns out that the best case is to choose only three servers for execution since the probability of execution on one of them will be very high and close to the confirmed execution of the task. Therefore, execution with a minimum number of resources, the result is not consuming system resources and ensuring execution with a high probability of task execution.

5. CONCLUSION

For the purpose of avoiding errors and re-executing tasks that cause loss of time and effort, most of the methods used a specific number of resources without taking into account the type and size of the task. Although the number of sources is large, the choice is ill-considered and the results of the implementation are highly uncertain. In this paper, we propose a fault-tolerant approach for scheduling functions in cloud/grid computing. This approach focuses on deliberately selecting a specific number of resources based on their known response times. By doing so, we achieve excellent performance, minimizing the impact on network resources, and ensuring uninterrupted performance. This deliberate selection reduces the number of resources required to execute tasks, thereby maximizing the likelihood of task completion and approaching near certainty. Executing tasks on carefully chosen resources with known response times yields highly favorable results. This approach optimizes performance, reduces resource utilization, facilitates seamless task completion with minimal disruption, and maintains network efficiency. The probability of successfully executing tasks becomes significantly high and nearing a level close to certainty.

ACKNOWLEDGEMENTS

The University of Babylon and the College of Information Technology deserve our heartfelt gratitude for their unwavering support of our staff.

REFERENCES

[1] M. Amoon, “Fault tolerance in grids using job replication,” *International Journal of Computing*, vol. 11, no. 2, pp. 115–121, 2014, doi: 10.47839/ijc.11.2.556.

[2] M. S. Almhanna, “Minimizing replica idle time,” in *2017 Annual Conference on New Trends in Information and Communications Technology Applications, NTICT 2017*, 2017, pp. 128–131. doi: 10.1109/NTICT.2017.7976134.

[3] R. M. Almuttairi, R. Wankar, A. Negi, R. R. Chillarige, and M. S. Almahna, “New replica selection technique for binding replica sites in data grids,” in *EPC-IQ01 2010 - 2010 1st International Conference on Energy, Power and Control*, 2010, pp. 187–194. doi: 10.37917/ijeee.6.2.16.

[4] K. H. Anun and M. S. Almhanna, “Web server load balancing based on number of client connections on docker swarm,” in *Proceedings of 2021 2nd Information Technology to Enhance E-Learning and other Application Conference, IT-ELA 2021*, 2021, pp. 70–75. doi: 10.1109/IT-ELA52201.2021.9773748.

[5] S. A. Abbas and M. S. Almhanna, “Distributed denial of service attacks detection system by machine learning based on dimensionality reduction,” *Journal of Physics: Conference Series*, vol. 1804, no. 1, pp. 1–12, 2021, doi: 10.1088/1742-6596/1804/1/012136.

[6] S. S. Sathya and K. S. Babu, “Survey of fault tolerant techniques for grid,” *Computer Science Review*, vol. 4, no. 2, pp. 101–120, 2010, doi: 10.1016/j.cosrev.2010.02.001.

[7] Q. Zheng and B. Veeravalli, “On the design of communication-aware fault-tolerant scheduling algorithms for precedence constrained tasks in grid computing systems with dedicated communication devices,” *Journal of Parallel and Distributed Computing*, vol. 69, no. 3, pp. 282–294, 2009, doi: 10.1016/j.jpdc.2008.11.007.

[8] S. P. Thiagarajah, M. Y. Alias, and W. N. Tan, “Qos controlled capacity offload optimization in heterogeneous networks,” *Bulletin of Electrical Engineering and Informatics*, vol. 9, no. 6, pp. 2667–2680, 2020, doi: 10.11591/eei.v9i6.2706.

[9] P. Kumari and P. Kaur, “A survey of fault tolerance in cloud computing,” *Journal of King Saud University - Computer and Information Sciences*, vol. 33, no. 10, pp. 1159–1176, 2021, doi: 10.1016/j.jksuci.2018.09.021.

[10] F. G. Khan, K. Qureshi, and B. Nazir, “Performance evaluation of fault tolerance techniques in grid computing system,” *Computers & Electrical Engineering*, vol. 36, no. 6, pp. 1110–1122, 2010, doi: 10.1016/j.compeleceng.2010.04.004.

[11] L. Yao, X. Wang, Q. Z. Sheng, S. Dustdar, and S. Zhang, “Recommendations on the Internet of Things: Requirements, Challenges, and Directions,” *IEEE Internet Computing*, vol. 23, no. 3, pp. 46–54, 2019, doi: 10.1109/MIC.2019.2909607.

[12] A. Litke, K. Tserpes, K. Dolkas, and T. Varvarigou, “A task replication and fair resource management scheme for fault tolerant grids,” *Lecture Notes in Computer Science*, vol. 3470, pp. 1022–1031, 2005, doi: 10.1007/11508380_104.

[13] C. Storm, “Fault Tolerance in Distributed Computing,” in *Specification and Analytical Evaluation of Heterogeneous Dynamic Quorum-Based Data Replication Schemes*, Wiesbaden: Vieweg+Teubner Verlag, 2012, pp. 13–79. doi: 10.1007/978-3-8348-2381-6_2.

[14] S. W. Kwak, K. H. You, and J. M. Yang, “Checkpoint management with double modular redundancy based on the probability of task completion,” *Journal of Computer Science and Technology*, vol. 27, no. 2, pp. 273–280, 2012, doi: 10.1007/s11390-012-1222-3.

[15] J. H. Abawajy, “Fault-tolerant scheduling policy for grid computing systems,” in *18th International Parallel and Distributed Processing Symposium, 2004. Proceedings.*, 2004, pp. 238–244. doi: 10.1109/IPDPS.2004.1303290.

[16] K. G. Srinivasa, G. M. Siddesh, and S. Cherian, “Fault-Tolerant Middleware for Grid Computing,” in *2010 IEEE 12th International Conference on High Performance Computing and Communications (HPCC)*, 2010, pp. 635–640. doi: 10.1109/HPCC.2010.60.

[17] C. Jiang and D. H. Zhou, “Fault detection and identification for uncertain linear time-delay systems,” *Computers and Chemical Engineering*, vol. 30, no. 2, pp. 228–242, 2005, doi: 10.1016/j.compchemeng.2005.08.012.

[18] M. Chtepen, B. Dhoedt, F. De Turek, P. Demeester, F. H. A. Claeys, and P. A. Vanrolleghem, “Evaluation of replication and rescheduling heuristics for grid systems with varying resource availability,” in *Proceedings of the IASTED International Conference on Parallel and Distributed Computing and Systems*, 2006, pp. 622–627.

[19] M. Amoon, “A fault-tolerant scheduling system for computational grids,” *Computers and Electrical Engineering*, vol. 38, no. 2, pp. 399–412, 2012, doi: 10.1016/j.compeleceng.2011.11.004.

[20] S. Song, K. Hwang, and Y. K. Kwok, “Risk-resilient heuristics and genetic algorithms for security-assured grid job scheduling,” *IEEE Transactions on Computers*, vol. 55, no. 6, pp. 703–719, 2006, doi: 10.1109/TC.2006.89.

[21] Y. Zhang and N. Lu, “Parameter selection for a centralized thermostatically controlled appliances load controller used for intra-

hour load balancing,” *IEEE Transactions on Smart Grid*, vol. 4, no. 4, pp. 2100–2108, 2013, doi: 10.1109/TSG.2013.2258950.
[22] A. Saoud and A. Recioui, “Hybrid algorithm for cloud-fog system based load balancing in smart grids,” *Bulletin of Electrical Engineering and Informatics*, vol. 11, no. 1, pp. 477–487, 2022, doi: 10.11591/eei.v11i1.3450.
[23] A. S. Kadhim and M. E. Manaa, “Hybrid load-balancing algorithm for distributed fog computing in internet of things environment,” *Bulletin of Electrical Engineering and Informatics*, vol. 11, no. 6, pp. 3462–3470, 2022, doi: 10.11591/eei.v11i6.4127.
[24] L. Tang and H. Chen, "Joint Pricing and Capacity Planning in the IaaS Cloud Market," in IEEE Transactions on Cloud Computing, vol. 5, no. 1, pp. 57-70, 1 Jan.-March 2017, doi: 10.1109/TCC.2014.2372811.
[25] W. S. W. Awang, M. M. Deris, O. F. Rana, M. Zarina, and A. N. M. Rose, "Affinity Replica Selection in Distributed Systems," *International Conference on Parallel Computing Technologies PaCT 2019: Parallel Computing Technologies,* Springer, Cham, 2019, vol 11657, pp. 385–399, doi: 10.1007/978-3-030-25636-4_30.
[26] Saibharath S., S. Mishra, and C. Hot, "Joint QoS and energy-efficient resource allocation and scheduling in 5G Network Slicing," *Computer Communications,* vol. 202, pp. 110–123, 2023, doi: 10.1016/j.comcom.2023.02.009.
[27] M. S. Almhanna, F. S. Al-Turaihi, and T. A. Murshedi, “Reducing waiting and idle time for a group of jobs in the grid computing,” *Bulletin of Electrical Engineering and Informatics*, vol. 12, no. 5, pp. 3115–3123, 2023, doi: 10.11591/eei.v12i5.4729.

BIOGRAPHIES OF AUTHORS

Mahdi S. Almhanna completed his Ph.D. research at Osmania University, India in December. 2012. He got his master’s degree in computer science from Osmania University, India in 2008. In 2007 he got a diploma in Arabic-English translation from Osmania University, Hyderabad, India. He has authored several international conference papers in the area of grid computing and he taught several subjects such as cloud computing, distributed systems, networks, and operating systems since February 2012 for undergraduate and postgraduate students in information and technology, Computer Engineering Colleges. His current research interest is in the area of data grid architecture, load balancing, and fault tolerance in distributed systems for grid resources, and replica election. He can be contacted at email: mahdi.almhanna@uobabylon.edu.iq.

Tariq A. Murshedi received the Ph.D. from Northeastern University, Shenyang, China. He is also an instructor at Cisco Networking Academy, Babylon University, Iraq. His research interests include routing protocols in mobile ad hoc networks and wireless networks. Currently, he is a lecturer in the Department of Information Networks at the University of Babylon, Babylon, Iraq. He can be contacted at email: tariq_alwan@itnet.uobabylon.edu.iq.

Firas Sabah Al-Turaihi is a lecturer at the University of Babylon, College of Information Technology, Department of Networks. He received his Ph.D. degree from Brunel University London, United Kingdom. He received his B.Sc. and M.Sc. degrees in Computer Science. His research interests cover communication networks. He can be contacted at email: firassabahalturaihi@uobabylon.edu.iq.

Rafah M. Almuttairi completed her Ph.D. research at the University of Hyderabad, India in April 2012. She received her master’s degree in computer science from the University of Baghdad, Iraq in October 2003. In 2007 she got a diploma in Arabic-English translation from Osmania University, Hyderabad, India. She has authored several international conference papers in the area of grid computing. During her work in the Ministry of Higher Education in Iraq, she has taught several subjects such as cloud computing, distributed systems, networks, operating systems, web-based applications, and interior design using 3DMax and Photoshop since Feb. 2004 for undergraduate and postgraduate students in Information and Technology, Fine Arts, Computer Engineering Colleges. Her current research interest is in the area of data grid architecture and fuzzy decision-making for grid resources and replica election and enhancing intrusion detection systems to search for hidden attack patterns using hadoop frameworks. She can be contacted at email: Rafah@uobabylon.edu.iq.