7-18-2023

Fault aware task scheduling in cloud using min-min and DBSCAN

S. M.F.D.Syed Mustapha
Zayed University

Punit Gupta
University College Dublin

Internet of Things and Cyber-Physical Systems 4 (2024) 68–76

Contents lists available at ScienceDirect

Internet of Things and Cyber-Physical Systems

journal homepage: www.keaipublishing.com/en/journals/internet-of-things-and-cyber-physical-systems

Fault aware task scheduling in cloud using min-min and DBSCAN

S.M.F D Syed Mustapha [a,*], Punit Gupta [b]

[a] College of Technological Innovation, Zayed University, Dubai, United Arab Emirates
[b] University College Dublin, Ireland

A B S T R A C T

Cloud computing leverages computing resources by managing these resources globally in a more efficient manner as compared to individual resource services. It requires us to deliver the resources in a heterogeneous environment and also in a highly dynamic nature. Hence, there is always a risk of resource allocation failure that can maximize the delay in task execution. Such adverse impact in the cloud environment also raises questions on quality of service (QoS). Resource management for cloud application and service have bigger challenges and many researchers have proposed several solutions but there is room for improvement. Clustering the resources clustering and mapping them according to task can also be an option to deal with such task failure or mismanaged resource allocation. Density-based spatial clustering of applications with noise (DBSCAN) is a stochastic approach-based algorithm which has the capability to cluster the resources in a cloud environment. The proposed algorithm considers high execution enabled powerful data centers with least fault probability during resource allocation which reduces the probability of fault and increases the tolerance. The simulation is cone using CloudsSim 5.0 tool kit. The results show 25% average improve in execution time, 6.5% improvement in number of task completed and 3.48% improvement in count of task failed as compared to ACO, PSO, BB-BC (Bib = g bang Big Crunch) and WHO(Whale optimization algorithm).

1. Introduction

Today we are living in the cloud computing era where cloud computing is the key to future computing in both domestic and global computing technologies. Rapid development of internet-based applications and expansion of internet usages have compelled the emergence of cloud computing technologies. Cloud computing provides services to the users based on pay per use which provides them dynamic and scalable virtual resources accessible through the internet on demand, and also its future development of traditional distributed, parallel and grid computing [1–5]. Cloud computing provides a lifeline for many new start-up's due to the availability of cost-effective and reliable resources at different times [6]. The cloud environment consists of five main key components, resource pooling, service-on-demand, available resource scalability, cost effectiveness, and most importantly is its availability [7].

Cloud Computing technology provides complete sets of computing needs of users with different features in terms of heterogeneity, on demand service, flexibility and many more. It has the capability to deal with the amount of data with respect to task scheduling and resource allocation policies [8]. Workflow and task scheduling also play a major role in cloud computing, which are the methods that provide the mapping of tasks to appropriate resources for execution. Efficient task scheduling is an essential part of cloud computing which is used to obtain high performance in a cloud environment. To improve the efficiency of cloud environment, researchers are working on this path where workflow technology [9,10] promises to provide solutions for these problems in cloud computing, mainly task scheduling and fault tolerance. The timing fault, which is a significant concern for fault tolerance, occurs when users cross the designated timeline or deadline for their tasks. Virtual Machine (VM) migration policies were also followed to deal with fault, in such situation data center broker migrate the virtual machine from one data canter to another to complete the task in minimal time bound [11]. One more thing that needs to be understood is that during the task execution, if the user's task requirement is big, then the time taken to execute the task will be high [12]. Cloud resources experience fluxes while delivering performance [13]. As user's task is expanding due to expansion of application or data, hence, an unavoidable growing of components will cause failure if not manageable [14]. Once the task fails, it affects the task scheduling and can lead to resource mismanagement. Hence, fault tolerance is a major concern in cloud computing such that mechanism building is required to deal with the fault. To deal with fault tolerance, it is the major responsibility of cloudlet scheduler to provide safety and safe delivery of user task even with the occurrence of failures in cloud environment [15]. Enhancing reliability for users, the current focus of cloud computing is to incorporate fault tolerance mechanisms into the cloud environment, specifically during task scheduling optimization and execution. Identifying faults poses a significant challenge, prompting numerous researchers to explore sustainable algorithms and effective

* Corresponding author.
E-mail addresses: smfdsm@yahoo.com, syed.duani@zu.ac.ae (S.M.FD.S. Mustapha).

https://doi.org/10.1016/j.iotcps.2023.07.003
Received 17 May 2023; Received in revised form 19 June 2023; Accepted 17 July 2023
Available online 18 July 2023

mechanisms for fault detection and handling. Fault detection plays a key role in fault tolerance which enables the task scheduler to sense the fault and execute the event to tolerate the fault and to improve the user's experience. This is such an NP-hard problem, probabilistic solution techniques are less exponential and have to deal with such hard problems [16]. Fault tolerance awareness is also being considered and many algorithms are being proposed by researchers, most of them are based on traditional algorithms such as ACO [17](Ant Colony Optimization), Modified ACO, PSO (Particle Swarm Optimization [18], GA (Genetic Algorithm), Min-Max, Min-Min [19] and there are many more algorithms that are being proposed by researchers. However, there is still some left to do in order to maximize the resource utilization and an optimistic approach is needed to deal with the task scheduling fault and to handle tolerance on it. The attainment of this goal relies on several crucial factors, such as employing advanced fault-aware algorithms [20–29], implementing a meticulous task scheduling approach, adopting a robust VM allocation policy, and taking into account other significant aspects during the design and proposal of a high-quality cloud environment.

Proposed traditional approaches are good but cloud environments require a different methodology to deal with fault tolerance and improvised task scheduling. Certain researchers suggest alternative clustering mechanisms to effectively group application or data elements, aiming to determine the optimal approach for handling fault tolerance. They explore various strategies to identify which elements should be clustered together and determine the most suitable approach for maintaining fault tolerance within the system. When striving for enhanced task scheduling and fault handling, the proposed algorithm should aim to maximize resource utilization, minimize the frequency of faults, and exhibit fault tolerance. It is imperative to explore improved task scheduling approaches and consider various other factors during the design process. Clustering of data of task, user's request and availability of VMs over the data center are also being considered. Certain approaches can be applied to maximize resource utilization and to improve efficiency by minimizing the fault. DBSCAN algorithm [30] is an advanced algorithm in the context of clustering and it is an unsupervised learning algorithm which has the capability to cluster the data (task information). Clusters are the formation of groups that can be mapped with suitable VM data, in other words, the user's tasks/requests for VM can be mapped with the suitable available VM. With this proposed solution task scheduling can also be optimized and fault handling can be improved, which leads to improvement in data center efficiency and resource utilization in a better manner. Traditional approaches like ACO, GA, PSO [17,18,31,32] etc. provide solution to handle the fault tolerance. These solutions focused task scheduling but not the efficiency in terms in-time and out-time, in other work completion of task and response time. Here proposed algorithm scheduling the user's task in cluster form with fittest VM, i.e., based on user's requirement, if task demands to highly configured VM then it will pick up from the highly configured VM pool(cluster) and associates with that tasks accordingly.

The work is divided into five sections. In section 2, some of the related works from task scheduling in cloud are discussed. In section 3, the proposed approach utilizing DBSCAN with the complete formulation is discussed. In section 4, experimental results are showcased with some comparative studies. In section 5, the conclusion and the outcomes of the proposed model in its improvement are discussed.

2. Related work:

Cloud service specialty is that it can be accessed from anywhere or any time on the basis of subscription, without user interaction tasks being scheduled and complete. Problems come out when the cloud environment is lacking with resource requests which creates a dilemma of how the task is going to be handled. This common situation is known as a fault in resource allocation or equalization and has to be handled without user noticing it. Many researchers have proposed solutions for fault aware and tolerance algorithms like in Ref. [33] that proposed fault power aware scheduling algorithm that minimizes the power consumption, request failure rate and minimize cost overall of the data center, but it is limited to power aware fault tolerance. Similarly fault tolerance and privacy security algorithm [34] in multi cloud environments where the authors use DeepSky and virtual storage concept to ensure security, which deal with SLA (service level agreement) and not with the task scheduling based fault tolerance. Other research works are done in a similar manner which provided survey reports over the cloud computing 's different areas such as hardware fault tolerance [20], software fault tolerance, proactive fault tolerance and currently proposed solutions for respective fault tolerance, and they lighten the solutions provided by the researchers. Here hardware fault tolerance can be minimized by using fault masking and dynamic recovery mechanism and can improve the output. Similarly, software fault tolerance can be handled with react fault tolerance technique with checkpoint and replication strategy and improvise the throughput. The author also introduced fault tolerance manager architecture and message passing interface which are being suggested by other researchers. An automatic fault tolerance approach for scientific workflow [21] that will handle such problems automatically. The proposed solution is having two phases, in the first phase heuristic hybrid scheduling is being proposed to schedule the workflow and in the second phase VM migration approach is being proposed. The proposed work shows that VM can be automatically migrated between the data centers whenever resource overutilization conditions occur. Fault tolerance can be handled with trust aware min-min scheduling and trust aware max-min algorithm to overcome the fault tolerance and to maximize the resource utilization [22]. Researchers are also considering adaptive framework-based solutions [23] to handle fault tolerance. Such methodology is applied by using replicas of VM and the checkpoint details to deal with fault. A Work Design Scheduling algorithm (WSSS) deals with Byzantine faults [24] that are not easily detected. Proposed work also includes check point optimization (TCC) to tolerate and eliminate the Byzantine fault. Here the WSSS algorithm helps to monitor server performance and also to detect Byzantine error prone regions which is related to workflow-based task scheduling and fault tolerance. Task scheduling with QoS (Quality of Service) is a major concern in cloud environment, fault tolerance and QoS maintain scheduling algorithm which is based on CNN (Content Addressable Network) model for mobile social cloud computing (MSCC) [25]. MSCC handles the task scheduling but the fault. Another way to schedule tasks in a more prominent way is to consider checkpoint strategies where checkpoint will be placed in several places to ensure the availability of appropriate resources. Such work is proposed [26] optimal checkpoint strategies to maximize the availability of a highly configured system and reduce the service time of the system in the terms of finishing time. Another mechanism proposed to stigmatize the fault tolerance using classification technique [27] which is based on Naive Bayes classification. This approach is a proactive based learning technique. It shows how artificial intelligence can provoke classification in cloud environments [28]. To handle fault in cloud computing adaptive fault tolerance mechanism being proposed by researcher [29] that provides a shade to an implementation of cloudlets and avoid network congestion and monitoring fault detection using migration method for adaptive approach. Targeting fault tolerance with avoidance of network congestion and applying VM migration. Proactive fault tolerance approach [35] in which clusters of VMs are made based on VM characteristics that deal with network resource consumption issues and energy being used while a physical machine is executing. The author compared the proposed approach with a traditional fault tolerance algorithm. Another survey report [36] lists out the challenges in scheduling techniques which show that good scheduling affects the QoS, Quality of Experience, latency, and performance of cloud services. Many researchers are working in this area but a highly optimized scheduling technique is still in demand. Correction of faulty application output and most of the algorithm result is erroneous and most of it contains error free data. Instead of rolling back all the operation in the segment being proposed the error localization [37]. This type of survey report shows that

only few focus on fault tolerance mechanisms to deal with fault, even many researchers also showed their work to predict the fault occurrence rate with their calculator to predict the fault probability and tolerance policy to maintain QoS of cloud service. A fault tolerant allocation (FTVMA) strategy also proposed where data of fail rate, execution time (start time + end time), VM (Virtual Machine) properties (Configuration of VM in terms of mips, core, space, connectivity etc.) are also being considered. In search for the solution of fault implementing fault aware scheduling policy to get the prediction of fault and then apply the tolerance solution. The solutions manifest through the availability of resources and a fault-aware job scheduling method proposed by Ref. [38]. This method incorporates a checkpoint mechanism that facilitates task migration in the event of resource failure. In this context, an integral aspect involves considering the failure history, including the user's satisfaction value, along with the proposal of a proactive fault tolerance method [39]. These parameters demonstrate their significance in formulating a robust methodology for cloud environments, effectively enhancing the performance of cloud resources while maintaining the quality of service (QoS). In this domain, traditional algorithms have also been employed to address faults within specific contexts. For instance, a suggested approach [40] adopts a greedy selection strategy where tasks are scheduled based on the quality of service (QoS) of the corresponding services. This entails monitoring QoS information across the cloud environment and processing task scheduling accordingly. However, a challenge arises as an additional process is needed to maintain records of the QoS for each data center and categorize tasks, which introduces complexity and requires additional parameters. This approach is reactive fault tolerance and mixes with resource allocation methodology which tolerates the fault by allowing the resources with respective QoS. To amplify the resource utilization and minimize the makespan of task Load Balanced improved Min-Min (LBIMM) algorithm also has been proposed [19] to allocate the task based on their resource budget constraints and balancing load. In the realm of cloud environments, a task scheduling approach with a focus on data-intensive tasks has been proposed by researcher [41]. This approach prioritizes the consideration of response time and task deadlines to minimize timespan, communication latency overhead, and maximize the hit rate. A task scheduling algorithm has been suggested that takes into account task deadlines when allocating resources during task scheduling in the cloud environment.

Certain algorithms adopt a holistic perspective of cloud resources, considering clusters of both unallocated and allocated resources. Task scheduling takes place based on specific requirements and timespan considerations. Notably, researchers like [42] have delved into this area and proposed a clustering-based fault tolerance mechanism. This mechanism focuses on real-time task scheduling to prevent faults and ensure effective fault tolerance. Furthermore, power consumption is monitored in relation to resource utilization. Additionally, an efficient approach utilizing fuzzy logic [43] has been introduced to handle fault tolerance in an effective manner. Usual error detection algorithm, fuzzy logic based, common error detection enabled with trained in cloud environment which can detect fault using analysis detection approach to predict the common error subsequently, initialization fault detection and keep the record of fault rate for better design of fuzzy logic to enrich the QoS of cloud computing environment. A dynamic clustering league championship algorithm (DCLCA) [30] has been proposed, aiming to cluster cloud resources based on a predefined set of criteria. This algorithm considers task scheduling and the allocation of virtual machines (VMs) with the necessary cloud resources. Clustering of cloud resources sounds like a novel approach but where we are going to apply also an important, place where it going to apply will decide whether it is good or bad. In this manner high execution power enabled data centers which are highly configured having low fault rate. Another major concern of fault probability which can be distinguished is the high and low fault probability of data centers. In this concept high probability of handling fault tolerance in efficient manner is high and also increase the QoS in both aspects as vendor and cloud service provider also add plus point to cloud resource utilization.

3. Proposed model

Cloud service is provided based on pay per use which is also required for maintaining QoS and fault handling as major concern of service. As discussed, many approaches have been proposed by many researchers to maintain the QoS and handle fault tolerance in an efficient manner. Some suggested fuzzy logic-based training to detect the common fault, some clustering league approach and many more researchers are working on this problem. The presence of fault issue plays a vital role and affects efficiency and QoS of the cloud environments. Researcher proposing different approaches and trying to minimize the fault and tolerate it but still it requires some improvement. Here if we change our view and filter the cloud resources with their capability, in this aspect user task can be scheduled to as per its requirement of resource which enhance the throughput also maximize the resource utilization in good manner. Proposed models based on clustering of cloud resource and their probability of low and high fault prediction. Let's take a dig upon the cluster of resources, in this we will collect the data of available resources, their space and other key elements details, with these details we can make cluster of them using sort of calculation on the probability of fault toleration and categorize the low and high fault occurrence. Keeping all these aspects to schedule the user's task which enhance the throughput and maximize the resource utilization with least probability of failure or fault of data centers. One more key aspect also being proposed here is having least probability of fault also make sure that task being schedule to highly configured and least fault probable data center. At the data center VM creation and VM allocation policy will be preferred with calculated probability. With the least number of fault rate user task scheduled and number of faults reduced.

The proposed algorithm is divided into two parts:

a) Clustering
b) Scheduling

3.1. Clustering

Scheduling task to highly configured VM and least fault probability can be clustered and user task scheduling can be implemented on top of that, to do so a best performing clustering algorithm is required which full fill our needs and deeds. Here proposed algorithm which is efficient and least time consuming to make desired cluster of input value is Density-based spatial clustering of applications with noise (DBSCAN) which is stochastic clustering algorithm that help to create the cluster based on density of elements and their distance with respected to closeness of item. In cloud environment data centers, their resources, VMs, and probability of fault occurrence. With this data we can also subcluster the data centers, resources, VMs, and probability of fault occurrences like data centers can b sub-clustered with their availability and their load of task, resource also sub clustered with respect to their configuration and also be subcategories with their high execution power, VMs also similarly sub clustered. These clusters distinguished the each and every resource which will be used to schedule the user task with their requirement of resource.

DBSCAN is described as stochastic clustering algorithm [30] which cluster the nodes based on their distance and divide them in cluster on the basis of distance from the initialize point we calculate or try to adjust the epsilon distance to get the main centroid point also known as core point which having two edges which are calculated with the help of distance that we initially put and calculate the radius of core points. The elements lying in that region will be consider in the cluster of that core point some elements or node lying at the edge of the region which draw its region with same procedure as core point. Here noisy data which doesn't meant to in any cluster discarded.

DBSCAN clustering algorithm can be divided in three segments, initially first part of the algorithm says that it needs to choose the epsilon

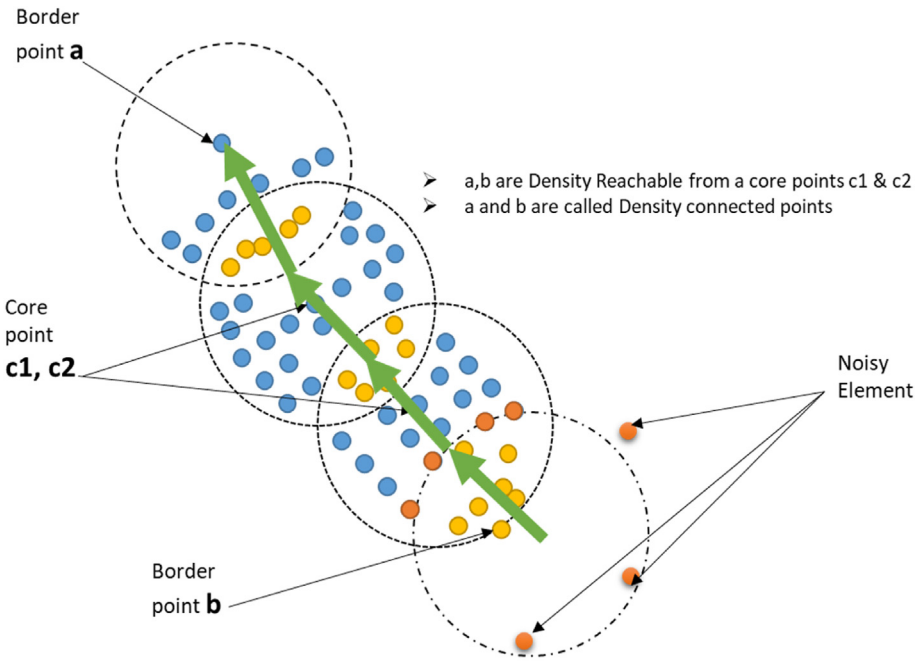

Fig. 1(a). DBSCAN basic structure with core points c1, c2, border points a and b [30].

value that is used to calculate the distance where territory of the core point will be marked from the core point, and with this marked region, it calculates the points lying within that region. Those points lying in this region are having similarity with their values. The second part of DBSCAN is to take selection of core or centroid point recursively till be get to the edge of the point or elements. The last phase of algorithm is to draw the border line of the points or in other words we can say finding the boarder of points or elements. Points which are not in range called noisy data which will be discarded.

As we can see from Fig. 1(a), it can be observed that core c1, c2 are connected with border point a, and b, also can be observed that a path can be drawn from border point a to b which show the reachability throw core points c1, c2. Edge point which lying at boarder of a, b call border point beyond the reachability point considered as noisy points, these noisy points discarded. Some point lying in both border and core point region having both capability of core point as well as boarder points.

$$N(p) = \{q\ \varepsilon\ D \mid \text{dist}\ (p, q)\} < = \varepsilon \quad \text{(i)}$$

Where N(p)- Neighbour of a point p in the data set D.

dist (p, q) - distance between two neighbour.
so the core point can be defined as if $|N(p)| >= \text{mPts}$.

A border point has fewer than mpts (minimum points) within it ε-Neighbourhood (N).

Now for direct density reachability from point b if-

1. $|N(b)| >= \text{mPts}, \text{i.e}, \text{b is a core point.}$
2. $\text{a}\ \varepsilon\ N(b),$ i.e, a is in the epsilon neighbourhood of b.

Also need to look for density reachable from a point b with respect to ε and mPts,

For a chain of points b1, b_2, b_3 … ….b_n, where $b_1 = b$, $b_n = a$ in such that b_{i+1} is direct density reachable from point b.

Cloud environments resources also can be considered as density of elements where VMs could be high configured or not. It is important to understand that highly configured virtual machines (VMs) with substantial computing power result in a lower failure probability for user tasks. When user tasks are scheduled on such highly configured VMs, the ratio of failure probability decreases significantly.

The scheduling of user tasks involves considering task requests and resource availability. User tasks can be viewed as individual elements, while virtual machines (VMs) can be seen as clusters of VMs with their specific specifications. When a user task arrives in the scheduler, it searches for the most suitable VM based on its core requirements. To achieve this, a proposed model based on DBSCAN is utilized, which handles fault tolerance by rescheduling tasks with the best fit VM. The selection of the best-fit VM is determined by choosing the VM with the least distance within the cluster. As we can see in the above figure noisy elements are those VM which are not being part of cluster and terminated immediately, searching for best fittest goes till the border point of cluster from cluster to cluster till the end point of border of cluster.

$$\text{VM'S} = (\text{vm}_1, \text{vm}_2, \text{vm}_3, \ldots\ldots\ldots\ldots\ldots\text{vm}_i)\ \text{[set of VMs over the cloud environments]}$$

$$\text{VM'}S_C = (\text{vvm}_1, \text{vm}_2, \text{vm}_3, \ldots\ldots\ldots\ldots\ldots\text{vm}_n)\ \text{[set of highly configured VMs over the cloud environments]}$$

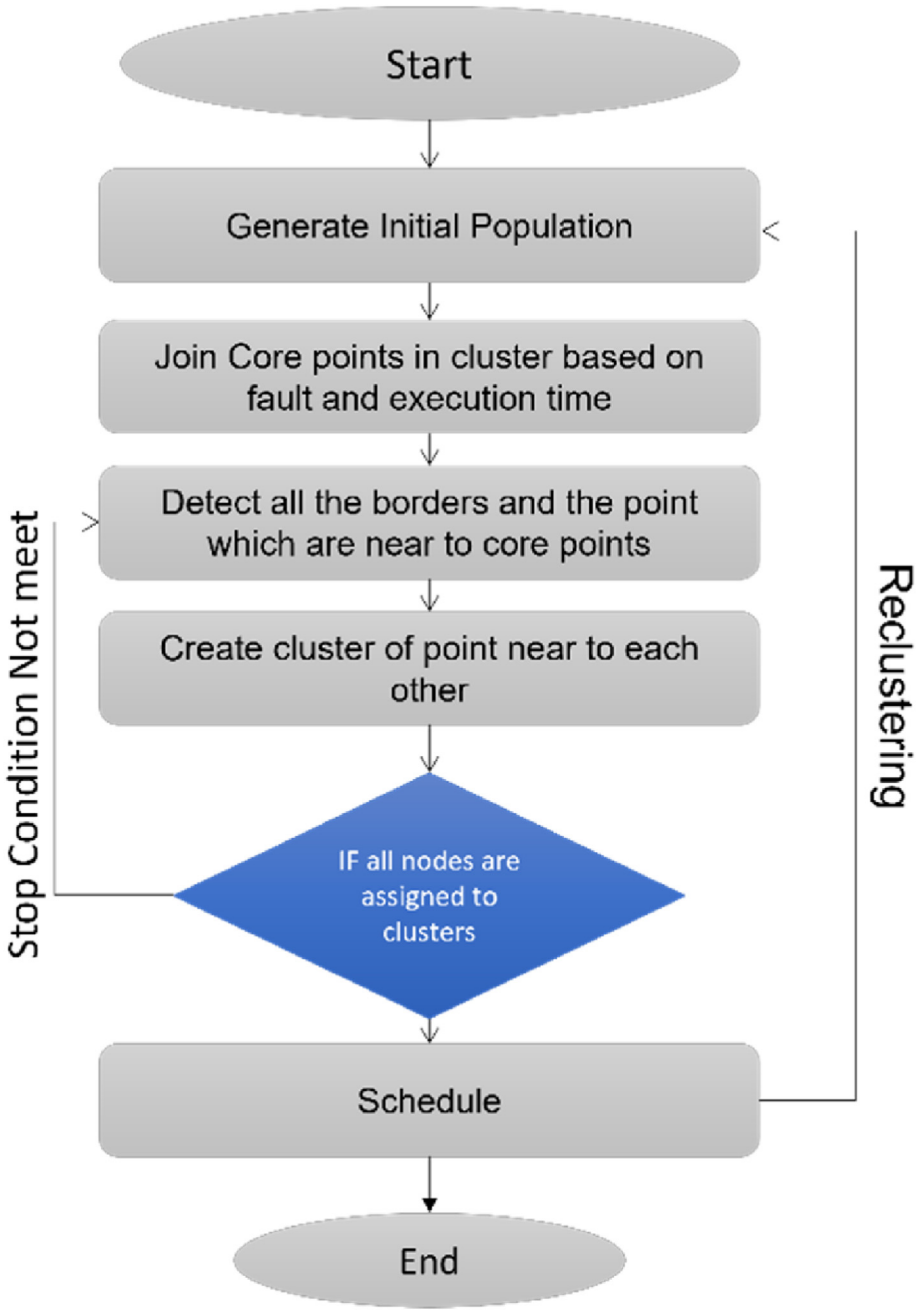

Fig. 1(b). Flow diagram of proposed task scheduling algorithm.

$$\text{tskT} = (t_1, t_2, t_3, \ldots\ldots\ldots\ldots\ldots t_i)\,[\text{set of tasks}]$$

$$\text{tskT} = (\text{treq}_1, \text{treq}_2, \text{treq}_3, \ldots\ldots\ldots\ldots\ldots \text{treq}_n)\,[\text{set of tasks which required high configured VMs}]$$

T(2)asks scheduling first as per user task arrived and mapping with best suitable strategies, when tasks has been scaled or required high configured VMs then need to look for density reachable approach of DBSCAN and associated with best fit VM's for tasks. As shown in Fig. 1(a), the blue dots show the VM's which having high configured VMs and yellow dots show ordinary VMs which are being part of cloud environments. Noisy elements are nothing but VMs which are either busy or not having enough configuration to satisfy the task, so all such noisy VMs are ignored while scheduling.

Now the tasks are going to be mapped with most suitable highly configured VMs even users request made with scalability inputs.

$$\text{User\_tskT} = \sum \text{VM'S}_c\ \{\text{from i to n}\}$$

Here the tasks are going to be associated with best suitable VMs.

Fitness function:

Table 1
Configuration Parameters of user tasks.

Parameters	Values
Task length	300 (small) 2000/3000(medium) 4000(high)
Input File Size	200 Byte
Output File Size	400 Byte
PE	1–2
Population(PSO/ACO)	100
Number of swarm/Ant	10
Fitness Function	Execution time

Table 2
Types of VM's.

	MIPS/Ram (mb)/PE
VM1	1000/512/1 Small machine
VM2	2000/1024/2 Medium machine
VM3	4000/2156/4 Large machine

Table 3
Parameter for existing algorithm.

DBSCAN	Min point: 10 Distance: euclidean Eps: 0.5
PSO	Number of Swarms: 100 Initial inertia weight: 0.9 Variable inertia weight: 0.2
ACO	Number of Ants:100 Evaporation rate ρ: 0.1 Pheromone factor α:1 Heuristic factor β: 1

$$Execution\_time_i = \frac{Task\ Length}{VM\_MIPS*core} \tag{1}$$

$$f(n) = \alpha * \frac{1}{Number\ of\ task\ failed} + \beta * Execution\ time \tag{2}$$

Where $\alpha + \beta = 1$.

3.2. Scheduling

In second phase the min-min algorithm is used to select a VM from the cluster of VM with least execution time and least faults in the cluster based on the fitness value shown in Equation (2). Fig. 1(b) shows the working of the proposed algorithm with clustering and scheduling using min-min algorithm. This allows the VM which is the least loaded in the cluster. Clusters are basically the group of VM's with the same computing power and failure probability. The clustering is repeated after equal interval of time to maintain the VM based on their performance in the cluster. So, migration of VM from one cluster to another cluster takes place in re-clustering phase.

4. Experiment & result

The simulation is carried out on Cloudsim 5.0. The simulation is done using 4 data centers with 2 host each task varying from 1000 to 10000 to study the performance in underloaded and over loaded condition. The simulation is done with scaling task load in order to study the performance of the proposed algorithm in under loaded and over loaded condition. To study the work with scaled resources the simulation is done with 5 VM's and 10 VM's. The work uses SWF(Standard Workload Format) files from parallel-workload repository. The work uses NASA iPSC workload log file. The details are shown in Table 1, Table 2 and Table 3 accordingly.

The results show that the proposed fault aware DBSCAN performs better than exiting ACO,PSO, BB-BC [44] and WHO [45] algorithms. The performance Matrix taken into consideration is the number of task failed, number of task completed and execution time.

Fig. 2(a) shows the comparative study of execution time i.e. the total time taken to execute the tasks taking into consideration 5 VM's. The experiment is done with scaling tasks to study the performance with increasing load. The result shows that the proposed model takes less time to complete tasks as compared to PSO and ACO.

Fig. 2(b) shows the comparative study of execution time i.e. the total time taken to execute the tasks taking into consideration 10 VM's. The experiment is done with scaling tasks to study the performance with increasing load. The result shows that the proposed model takes less time to complete tasks as compared to PSO and ACO. The study is important because even the proposed model completed most of the tasks and with least time.

Fig. 3(a). shows a comparative study of number of tasks completed for 5 VM's. The study shows that the proposed model completed more tasks as compared to exiting models. This is because the model is able to find a least faulty data center and VM with least failure probability.

where the experiment scales the task from 1000 to 5000 tasks.

Fig. 3(b) shows the comparative study of number of tasks complete for 10 VM's to study the performance with increasing infrastructure. The experiment is done with scaling tasks to study the performance with increasing load. The result shows that the proposed model completes more tasks as compared to PSO and ACO. As shown in Fig. 3(a) and (b)

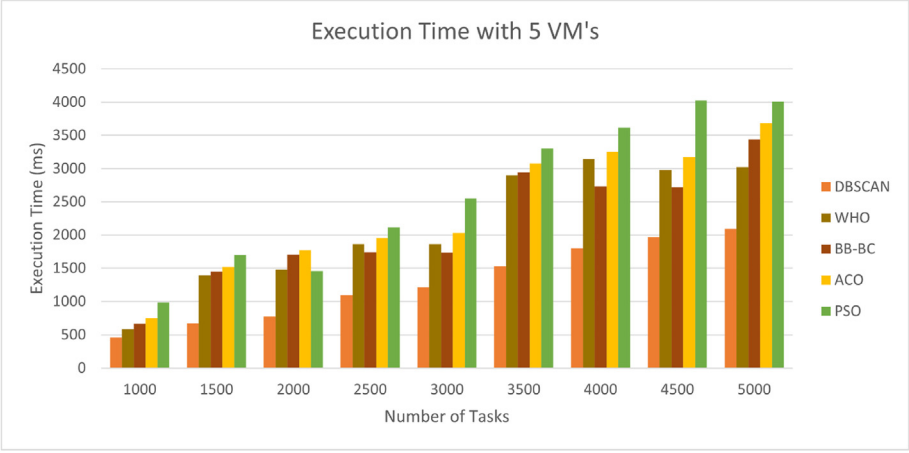

Fig. 2(a). Execution time comparisons for 5 VM's.

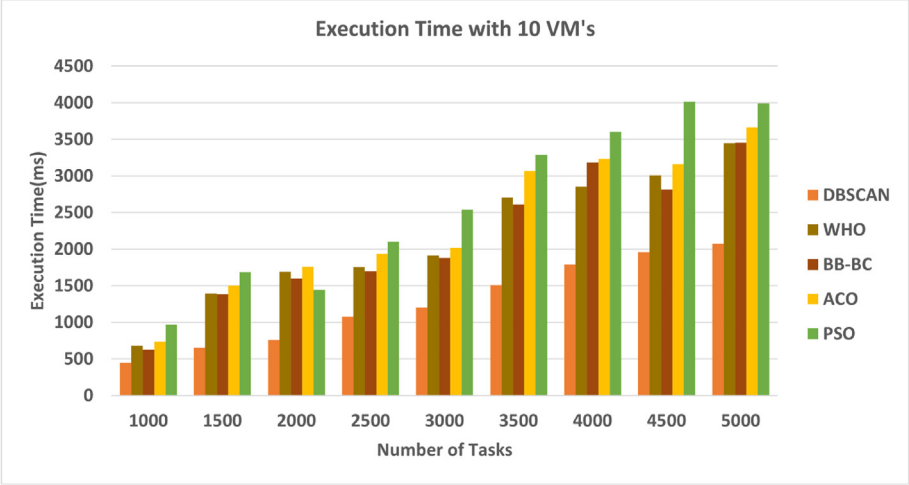

Fig. 2(b). Execution time comparisons for 10 VM's.

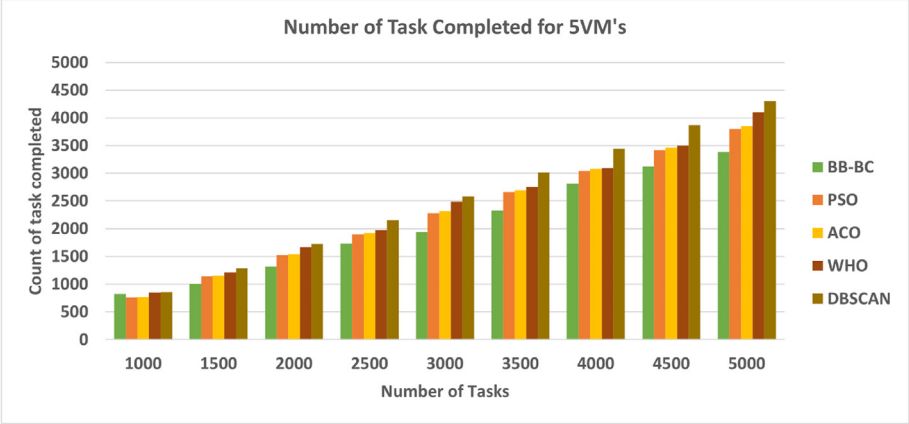

Fig. 3(a). Number of tasks Completed comparisons for 5 VM's.

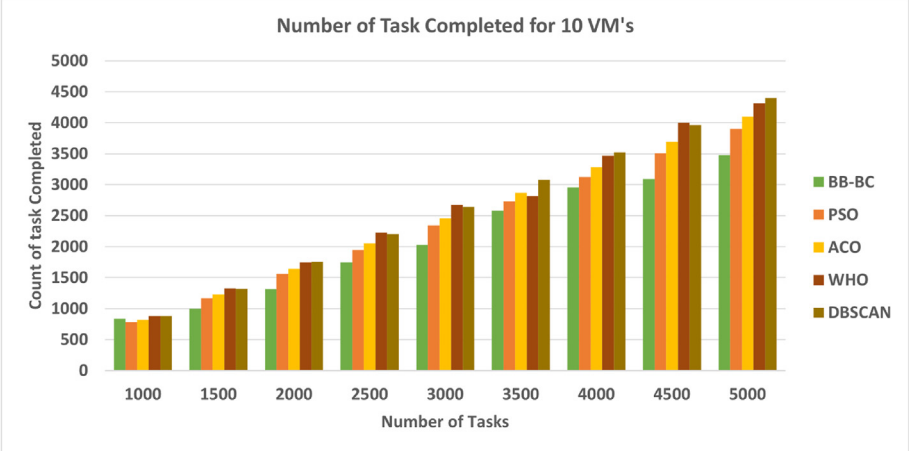

Fig. 3(b). Number of tasks Completed comparisons for 10VM's.

for task count 100 and 2000 the performance of proposed algorithm is poor due to small size of clusters and initial phase of the algorithm, but with increasing load the performance of the algorithm improves with increasing number of tasks.

Similarly Fig. 4(a) and (b) shows the comparative study of number of tasks that failed for 5 and 10 VM's. The experiment is done with scaling tasks to study the performance with increasing load. The result shows that with the proposed model less number of task are failed as compare to PSO and ACO exiting algorithms. The study is important because the algorithm is able to find the least failure probability infrastructure which improves the failure tolerance of the system. The proposed model improves the reliability of the system as the number of faults reduces. In the

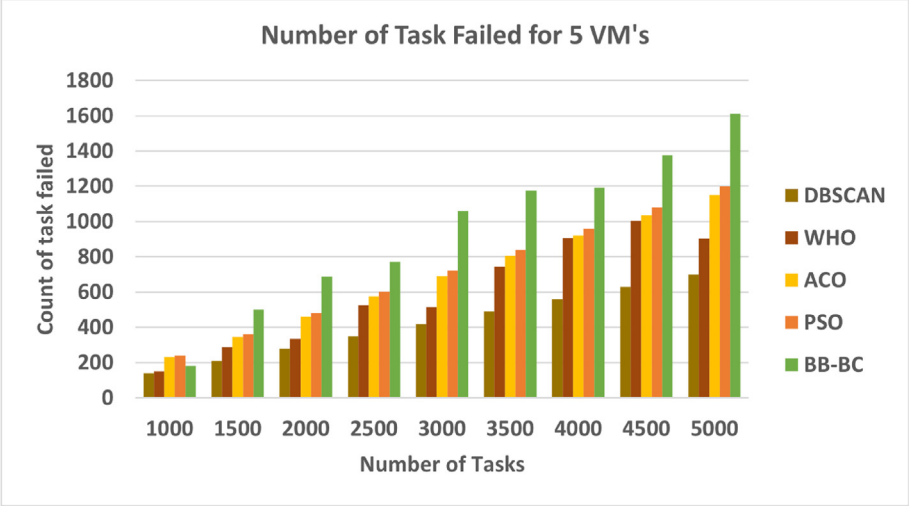

Fig. 4(a). Number of tasks failed comparisons for 5 VM's.

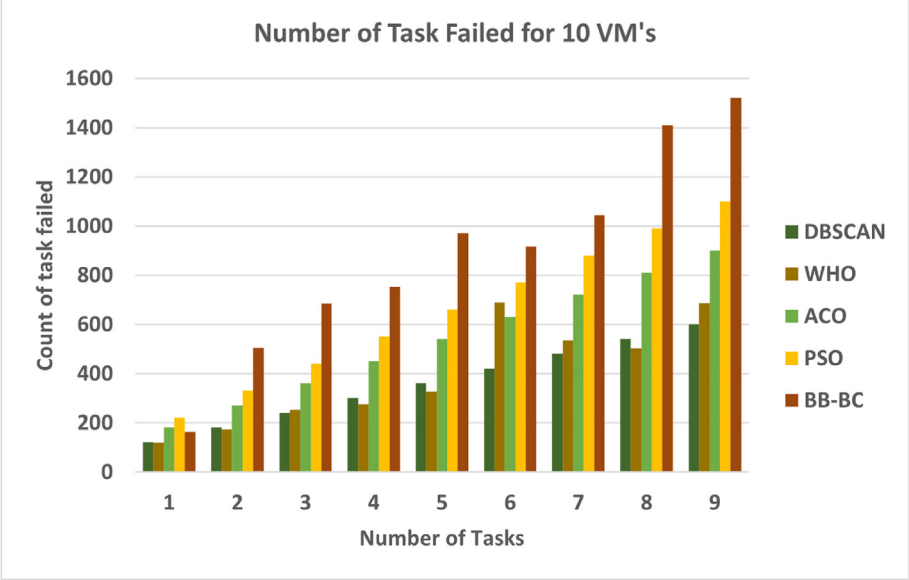

Fig. 4(b). Number of tasks failed comparisons for 10 VM's.

beginning when the cluster size is small, the proposed algorithm shows worst results but with increasing number of task, the number of task that failed reduces with time.

5. Conclusion

The result section demonstrates the effectiveness of the proposed algorithm for task scheduling in cloud infrastructure The result shows that the proposed algorithm outperforms exiting ACO, PSO, BB-BC [44] and WHO [45] algorithms for task scheduling in cloud with scaling load taking 5 virtual machines and 2 data centers. The proposed model performance is studied using number task that failed, completed and total execution time as performance parameters. The performance is studied using execution time, average start time, average finish time and total execution time as performance parameters. The result shows that the proposed algorithm performs better over the existing approaches algorithms in the context of execution time, number of tasks completed and failed. The evaluation is done by scaling the number of tasks from 100 to 10000 to study the performance in underloaded and overloaded conditions and the second evaluation with increasing the number of virtual machines i.e. scaling resources. Under both the conditions of underloaded, average and overloaded condition the proposed algorithm performs better than the existing model with a 25% improvement in execution time 6.5% improvement in number of tasks completed and 3.48% reduction in number of task failed. The work can further be extended with more objective functions like cost and power efficient algorithm. In future the algorithm can be merged with machine learning algorithm to further improve the performance of the algorithm.

Data availability

The dataset for simulation is supported by parallel http://workload.com for real-time analysis.

Funding

The research project is funded by Startup Research Grant R21043 from Zayed University, Dubai, United Arab Emirates.

Declaration of competing interest

The authors have no conflicts of interest to declare, and there is no financial interest to report.

References

[1] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R.H. Katz, A. Konwinski, G. Lee, D.A. Patterson, A. Rabkin, I. Stoica, M. Zaharia, A view of cloud computing, Commun. ACM 53 (4) (2010) 50–58.

[2] K. Chen, W.M. Zheng, Cloud computing: system instances and current research, J. Softw. 20 (5) (2009) 1337–1348.

[3] J.X. Zhang, X.M. Gu, C. Zheng, Survey of research progress on cloud computing, Appl. Res. Comput. 27 (2) (2010) 429–433.

[4] B.P. Rimal, E. Choi, A service-oriented taxonomical spectrum, cloudy challenges and opportunities of cloud computing, Int. J. Commun. Syst. 25 (2012) 796–819.

[5] I. Foster, Y. Zhao, I. Raicu, S. Lu, Cloud computing and grid computing 360-degree compared, Grid Comput. Environ. Workshop (GCE'08) (2008) 1–10, https://doi.org/10.1109/GCE.2008.4738445.

[6] R.K. Gupta, R.K. Pateriya, Balance resource utilization (BRU) approach for the dynamic load balancing in cloud environment by using AR prediction model, J. Organ. End User Comput. 29 (4) (2017) 24–50, https://doi.org/10.4018/JOEUC.2017100102.

[7] B. Hicham, B. Said, A. Touhafi, A. Ezzati, Deadline and Energy Aware Task Scheduling in Cloud Computing, 2018. Paper presented at the 2018 4th International Conference on Cloud Computing Technologies and Applications (Cloudtech).

[8] Z.C. Papazchos, H.D. Karatza, Scheduling of frequently communicating tasks, Int. J. Commun. Syst. 25 (2012) 146–157.

[9] H. Luo, Y. Fan, C. Wu, Over view of workflow technology, J. Softw. 11 (7) (2000) 899–907.

[10] N. Kaur, T.S. Aulakh, R.S. Cheema, Comparison of workflow scheduling algorithms in cloud computing, Int. J. Adv. Comput. Sci. Appl. 2 (10) (2011) 81–86.

[11] Z. Ahmad, A.I. Jehangiri, M. Iftikhar, A.I. Umer, Data-oriented scheduling with dynamic-clustering fault-tolerant technique for scientific workflows in clouds, Program. Comput. Software 45 (8) (2019) 506–516.

[12] B. Wu, K. Hao, X. Cai, T. Wang, An integrated algorithm for multiagent fault-tolerant scheduling based on MOEA, Future Generat. Comput. Syst. 94 (2019) 51–61.

[13] A.Y. Gital, A.S. Ismail, M. Chen, H. Chiroma, A framework for the design of cloud based collaborative virtual environment architecture, in: Proceedings of the International Multi Conference of Engineers and Computer Scientists, 2014.

[14] Y.-H. Moon, C.-H. Youn, Multihybrid job scheduling for fault-tolerant distributed computing in policy-constrained resource networks, Comput. Network. 82 (2015) 81–95.

[15] J. He, M. Dong, K. Ota, M. Fan, G. Wang, NetSecCC: a scalable and fault-tolerant architecture for cloud computing security, Peer-to-Peer Netw Appl 9 (1) (2014) 67–81.

[16] N.M. Nawi, A. Khan, M.Z. Rehman, H. Chiroma, T. Herawan, Weight optimization in recurrent neural networks with hybrid metaheuristic Cuckoo search techniques for data classification, Math. Probl Eng. (2015) 1–18.

[17] L. Zuo, L. Shu, S. Dong, C. Zhu, T. Hara, A multi-objective optimization scheduling method based on the ant colony algorithm in cloud computing, IEEE Access 3 (2015) 2687–2699.

[18] M. Farid, R. Latip, M. Hussin, N.A.W. Abdul Hamid, A survey on QoS requirements based on particle swarm optimization scheduling techniques for workflow scheduling in cloud computing, Symmetry 12 (4) (2020) 551.

[19] H. Chen, F. Wang, N. Helian, G. Akanmu, User-priority guided Min-Min scheduling algorithm for load balancing in cloud computing, PARCOMPTECH) (2013), https://doi.org/10.1109/ParCompTech.2013.6621389. Paper presented at the 2013 national conference on parallel computing technologies.

[20] Jasbir Kaur, Supriya Kinger, Analysis of different techniques used for fault tolerance, IJCSIT) Int. J. Comput. Sci. Inform. Technol. 5 (3) (2014) 4086–4090.

[21] Anju Bala, Inderveer Chana, Autonomic fault tolerant scheduling approach for scientific workflows in Cloud computing, Concurr. Eng. 23 (1) (2015) 27–39.

[22] Punit Gupta, et al., Trust Aware Workflow Scheduling in Scalable Cloud Environment, 2019.

[23] Mohammed Amoon, Adaptive framework for reliable cloud computing environment, IEEE Access 4 (2016) 9469–9478.

[24] Sathya Chinnathambi, et al., Scheduling and checkpointing optimization algorithm for byzantine fault tolerance in cloud clusters, Cluster Comput. 22 (6) (2019) 14637–14650.

[25] SookKyong Choi, KwangSik Chung, Heonchang Yu, Fault tolerance and QoS scheduling using CAN in mobile social cloud computing, Cluster Comput. 17 (3) (2014) 911–926.

[26] Punit Gupta, Satya Prakash Ghrera, Load and fault aware honey bee scheduling algorithm for cloud infrastructure, in: Proceedings of the 3rd International Conference on Frontiers of Intelligent Computing: Theory and Applications (FICTA) 2014, Springer, Cham, 2015.

[27] Bashir Mohammed, et al., Failover strategy for fault tolerance in cloud computing environment, Software Pract. Ex. 47 (9) (2017) 1243–1274.

[28] Deepak Kochhar, H. Jabanjalin, An approach for fault tolerance in cloud computing using machine learning technique, Int. J. Pure Appl. Math. 117 (22) (2017) 345–351.

[29] T. Tamilvizhi, B. Parvathavarthini, A novel method for adaptive fault tolerance during load balancing in cloud computing, Cluster Comput. 22 (5) (2019) 10425–10438.

[30] Derya Birant, Kut Alp, ST-DBSCAN: an algorithm for clustering spatial–temporal data, Data Knowl. Eng. 60 (1) (2007) 208–221.

[31] Z. Zhou, F. Li, H. Zhu, H. Xie, J.H. Abawajy, M.U. Chowdhury, An improved genetic algorithm using greedy strategy toward task scheduling optimization in cloud environments, Neural Comput. Appl. 32 (2020) 1531–1541.

[32] P.S. Rawat, P. Dimri, S. Kanrar, G.P. Saroha, Optimize task allocation in cloud environment based on big-bang big-crunch, Wireless Pers. Commun. 115 (2020) 1711–1754.

[33] Punit Gupta, S.P. Ghrera, Power and fault aware reliable resource allocation for cloud infrastructure, Proc. Comput. Sci. 78 (2016) 457–463.

[34] Maha Tebaa, Said EL Hajji, From single to multi-clouds computing privacy and fault tolerance, IERI Procedia 10 (2014) 112–118.

[35] Jialei Liu, et al., Using proactive fault-tolerance approach to enhance cloud service reliability, IEEE Trans. Cloud Comput. 6 (4) (2016) 1191–1202.

[36] Hassan Asghar, Eun-Sung Jung, A Survey on Scheduling Techniques in the Edge Cloud: Issues, Challenges and Future Directions, 2022 arXiv preprint arXiv: 2202.07799.

[37] Joseph Sloan, Rakesh Kumar, Greg Bronevetsky, An algorithmic approach to error localization and partial recomputation for low-overhead fault tolerance, in: 2013 43rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), IEEE, 2013.

[38] A. Latiff, M. Shafie, A checkpointed league championship algorithm-based cloud scheduling scheme with secure fault tolerance responsiveness, J Applied Soft Computing 61 (2017) 670–680, https://doi.org/10.1016/j.asoc.2017.08.048.

[39] V. Sathiyamoorthi, et al., Adaptive fault tolerant resource allocation scheme for cloud computing environments, J. Organ. End User Comput. 33 (5) (2021) 135–152.

[40] J. Li, L. Feng, S. Fang, An greedy-based job scheduling algorithm in cloud computing, J. Softw. 9 (4) (2014) 921–925, https://doi.org/10.4304/jsw.9.4.921-925.

[41] P. Suresh, P. Balasubramanie, User demand aware scheduling algorithm for data intensive tasks in grid environment, Eur. J. Sci. Res. 74 (4) (2012) 609–616.

[42] M. Khaldi, M. Rebbah, B. Meftah, O. Smail, Fault tolerance for a scientific workflow system in a cloud computing environment, Int. J. Comput. Appl. 42 (7) (2020) 705–714, https://doi.org/10.1080/12062 12X.2019.16476 51.

[43] M. Khaldi, M. Rebbah, B. Meftah, O. Smail, Fault tolerance for a scientific workflow system in a cloud computing environment, Int. J. Comput. Appl. 42 (7) (2020) 705–714, https://doi.org/10.1080/12062 12X.2019.16476 51.

[44] P. Gupta, D.K. Saini, P.S. Rawat, S. Bhagat, Hybrid Big Bang-Big Crunch based resource scheduling to improve QoS in cloud infrastructure, J. Intell. Fuzzy Syst. 43 (2) (2022) 1887–1895.

[45] P. Gupta, S. Bhagat, D.K. Saini, A. Kumar, M. Alahmadi, P.C. Sharma, Hybrid whale optimization algorithm for resource optimization in cloud E-healthcare applications, Comput. Mater. Continua (CMC) 71 (3) (2022).