Reliable Task Scheduling in Cloud Computing Using Optimization Techniques for Fault Tolerance

Jian Ma[1], Chaoyong Zhu[2], Yuntao Fu[3*], Haichao Zhang[3], Wenjing Xiong[3]

[1] State Grid Yingda CO., LTD., Beijing 100005, China
[2] State Grid Yingda International Holdings CO., LTD, Beijing 100005, China
[3] State Grid Huitongjincai (Beijing) Information Technology CO., LTD, Beijing 100077, China
E-mail: fyt_sght@163.com

Keywords: optimization theory, cloud computing, resource usage prediction, task scheduling

Received: October 12, 2024

To propose a reliable cloud computing task deployment algorithm for the optimization theory. The current research on cloud computing task deployment mainly only focuses on one of the two goals: reliability and optimization theory. This paper studies how to provide fault tolerance for task execution failure while minimizing the number of servers used to perform all tasks, thus reducing the problem of optimization theory. This article provides fault recovery capability through task replication, providing two instances of each task that make up the job. Task copies can be deployed either on a dedicated backup server or to the server where the main task is located by sharing the same computing resources and running at less than the execution speed of the main task. We propose a reliable cloud computing task deployment algorithm for optimizing theoretical optimization and service quality perception. For users, the completion time of the service is usually limited, and if a timeout occurs, it will cause a loss to the cloud service provider. For the actual completion time performance of the task at the last moment, the algorithm RER is about 2% to 10% more than the algorithm QSRE at xtr = 0.75. Time out times of the algorithm RER (xtr = 0.75). Suppose the task fails at a random time. In that case, the algorithm RER (xtr = 0.75) has a 10% -15% probability over the execution period of the job, and the algorithm RER has a 42% to 63% probability of timeout. The algorithm RER (xtr = 0.5) is 12% to 22% less than the algorithm QSRE. This paper studies how to minimize the number of servers used to perform all task copies while ensuring service quality and providing fault tolerance, thus reducing the problem of optimization theory.

Povzetek: Predlagan je zanesljiv algoritem za razporejanje nalog v oblaku, ki optimizira število uporabljenih strežnikov in zagotavlja toleranco napak z replikacijo nalog.

1 Introduction

The rationale for scalable computing and data-parallel frameworks is to decompose user-submitted jobs into multiple tasks and deploy them to the server for parallel execution. When a job arrives, the data center deploys the appropriate nodes for the tasks that make up the job, leading to task deployment problems [1, 2]. The impact of task deployment on the performance of cloud computing systems is mainly manifested in bringing huge costs to cloud service providers and users. Many task deployment algorithms have been proposed to deploy tasks to cloud servers [3, 4]. Given the vast complexity of cloud data centers, with the explosion of system size, hardware and software failures have become common during application execution [5, 6]. Failure is inevitable due to the large number of components in the data center infrastructure. Despite the improved reliability of individual components, the failure rate of the entire system remains high. A variety of reasons cause faults. One-third of the machines and more than 8% of the memory face at least one correctable error per year, and 1.3% and 0.22% of the year, respectively [7]. Even if the average interval time for a single node is 5 years, the average interval time for a computing system with 200,000 nodes will be less than one hour. Therefore, it is essential to use fault-tolerant operations for reliability planning. However, these systems are less reliable than they claim [8]. According to Google, each Map Reduce workflow has five permanent failures in the form of machine crashes, with at least one disk failure in each Map Reduce workflow running 4,000 tasks. A Microsoft study on one million PCs showed that the CPU and chip subsystems fail frequently [9].

Cloud computing is increasingly becoming an indispensable and demanding platform for a variety of computing. At present, there are two central schemes to realize checkpoints: uncoordinated checkpoints and coordinated checkpoints [10, 11]. After the writes are complete, the application continues to execute. If a failure occurs, all processes will resume execution from the checkpoint. Coordinated checkpoints are easy to implement, and the desired natural synchronization points exist in most applications. However, the need for global coordination leads to its lack of scalability [12, 13]. In an uncoordinated checkpoint, each task performs the checkpoint independently and recovers from the local store in case of a failure. In contrast, for synchronization without a coordinated checkpoint, the nodes hold the nearest message log they send. When a node recovers from the previous checkpoint, it can receive the message using the log replay of the remote node, which reduces the overhead of running when fault-free [14, 15]. A mathematical model based on integrated linear planning is established, and QoS adjusts the reliability and optimization theory of cloud computing systems under the service period to allocate resources in the way of fault perception and efficient energy saving [16, 17]. Proposes three energy-saving optimization algorithms for the reliability optimization of priority constraint tasks in heterogeneous clusters [18, 19]. We propose an effective data center model to the Map Reduce infrastructure operation completion reliability and job optimization theory by assuming that the fault follows a Poisson distribution.

2 Lyapov optimization theory

2.1 Message queue modeling

Make sure the message is delivered at least once. Waiting for confirmation of the PUBACK message. As shown in Equation (1), (2), for the sender of the message, an unused message identifier should be assigned, and the message is retained and tried to be repeated until the PUBACK message corresponding to the message is received.

$$Q_k(t+1)=max[Q_k(t)-b_k(t),0]+a_k(t) \quad (1)$$

$$Q_k(t+1)=max[Q_k(t)-d_k(t)-b_k(t),0]+a_k(t) \quad (2)$$

For the receiver of the message, after receiving the message, needs should send the PUBACK message with the received message identifier to the sender of the message. After sending the PUBACK message, as shown in equation (3), if the recipient receives a message with the same message identifier, it will be processed as a new message.

$$0\le b_k(t)\le B_{max} \quad (3)$$

The highest level of message service quality ensures that the message is delivered once and only to the recipient. For the sender of the message, you must assign an unused message identifier to the new message to be sent. As shown in Equation (4), this message is treated as unidentified until the corresponding PUBREC message is received from the recipient.

$$0\le a_k(t)\le A_{max} \quad (4)$$

After receiving the PUBREC message, the sender will send another PUBREL message with the original message identifier, as shown in equation (5), (6), the PUBREL message will be treated as unconfirmed long before the PUBCOMP message is received from the recipient. After defining the system stability, the next problem to be solved is how to analyze the stability of the system.

$$0\le d_k(t)\le D_{max} \quad (5)$$

$$\lim_{T\to\infty} sup \frac{1}{T}\sum_{T=0}^{T-1}\mathrm{E}\{|\ Q(t)|\}<\infty \quad (6)$$

Such as the stability theorem of ordinary differential equations. As shown in equation (7), (8), the system stability discrimination method of such classical control theory is only suitable for studying BIBO stability in SISO linear constant system.

$$Z_k(t+1)=max[Z_k(t)+1_{\{Q_k(t)>0\}}(\grave{o}_k-b_k(t)) \quad (7)$$

$$Q_k(t)\le Q_k^{max},\quad Z_k(t)\le Z_k^{max} \quad (8)$$

The structure of the modern control system is often no longer a simple linear constant system, there will be a large number of non-linear or chronotropic factors. In a stable linear system, as shown in Equation (9) and (10), the characteristic equation and does not change by factors such as initial conditions and external interference.

$$Z_k(\tau+1)=max[Z_k(\tau)+\grave{o}_k-b_k(t)-d_k(t),0] \quad (9)$$

$$Z_k(\tau+1)\ge Z_k(\tau)+\grave{o}_k-b_k(t)-d_k(t) \quad (10)$$

2.2 QoS selection strategy optimization

For stability, Lyapunov gives a definition in its theoretical sense using both norm and spherical domain concepts. The norm is mathematically defined as a measure of the distance between points in the dimensional space of n, as shown in Equation (11), (12), for any two points x1 and x2 in the n-dimensional space. According to the space of the measure and the meaning of the measure, there will be various specific norms accordingly.

$$Z_k(t+W_k^{max}+1)-Z_k(t+1)\ge \grave{o}_k W_k^{max}-\sum_{\tau=t+1}^{t+W_k^{max}}(b_k(\tau)+d_k(\tau))$$
(11)

$$\grave{o}_k W_k^{max}-Z_k^{max}\le \sum_{\tau=t+1}^{t+W_k^{max}}(b_k(\tau)+d_k(\tau)) \quad (12)$$

The common norm is described as the sum of the projection length of the n-dimensional vector in each dimension, and the norm is described as the second of the sum of the n-dimensional vector in each dimension, as shown in Equation (13), the maximum value of the n-dimensional vector projected length on each dimension.

$$\sum_{\tau=t+1}^{t+W_k^{max}}(b_k(\tau)+d_k(\tau))\ge Q_k(t+1) \quad (13)$$

If the system does equal shock, if it does not exceed the sphere range, it is said to meet the stability of Lyapunov, in the classical control theory will be judged as unstable. As shown in equation (15), (16), under the Lyapunov stability definition.

$$W_k^{max} \le (Q_k^{max} + Z_k^{max}) / \epsilon_k \quad (14)$$

If the state corresponds to BIBO stability in classical control theory, as shown in equation (15) and (16), from the perspective of engineering significance, asymptotic stability is often used rather than stability.

$$L(\Theta(t)) = \frac{1}{2}\sum_{k=1}^{K}[Q_k(t)^2 + Z_k(t)^2] \quad (15)$$

$$\Delta(\Theta(t)) = L(\Theta(t+1)) - L(\Theta(t)) \quad (16)$$

Lyapunov large-range stability is for all states in the n-dimensional state space, from any state, as shown in Equation (17), (18).

$$\gamma = B + Vp^* - \epsilon f(\theta(t)) \quad (17)$$

$$\varepsilon(t) = \Delta(\theta(t)) + Vp(t) \quad (18)$$

The first type of method is to linearize the equilibrium state in the nonlinear system, as shown in equation (19), (20), and then the stability of the nonlinear system can be discussed by studying the distribution and stability of the eigenvalues of the linearized system.

$$\mathrm{E}[\varepsilon(t)] \le B + Vp^* - \epsilon f(\theta(t)) \quad (19)$$

$$\lambda = \frac{|\varepsilon - \gamma|}{|\varepsilon| + \gamma} \quad (20)$$

3 Research on the prediction algorithm of cloud computing resource usage

3.1 Research on the resource usage prediction problem of cloud computing

The development of Lyapunov's second method has re-entered the field of control and has recently become the most important method for studying the stability of the system. Lyapunov's second method studies and judges the system stability in the equilibrium state, with its energy at the minimum value [20, 21]. If the equilibrium state of the system is gradually stable, then after the system is disturbed, the system will store the energy brought by the interference and start from the equilibrium state. Over time, the stored energy gradually decays and eventually returns to equilibrium [22, 23]. If the system is unstable, then after the system receives interference, it will absorb energy from the outside in the process of movement, and the stored energy will become more and more extensive, away from the field of equilibrium state. Based on this theory, if a positive definite scalar function that can describe the energy of the system and the symbolic nature of the derivative of one order will increase, decrease, or change over time, the stability of the system can be judged [24, 25]. We use the concept of Lyapunov drift to study the link selection problem after random data packets reach a multi-hop packet wireless network. To process the scheduling of the mobile network. The mathematical analysis is through the Lyapunov drift theory and cooperates with the flow control mechanism to maximize the network utility. Lyapunov was officially introduced into the queuing network. The theory is used to optimize performance indexes such as average power and throughput while stabilizing the queuing network. Lyapunov drift optimization theory is being applied in more and more fields [26, 27]. Figure 1 is the flow diagram of the taboo search algorithm, applying the theory to the delay perception and two problems by comparing convex optimization standard technology of low complexity, closed suboptimal solution; the result shows that the algorithm has a fundamental improvement in time [28, 29].

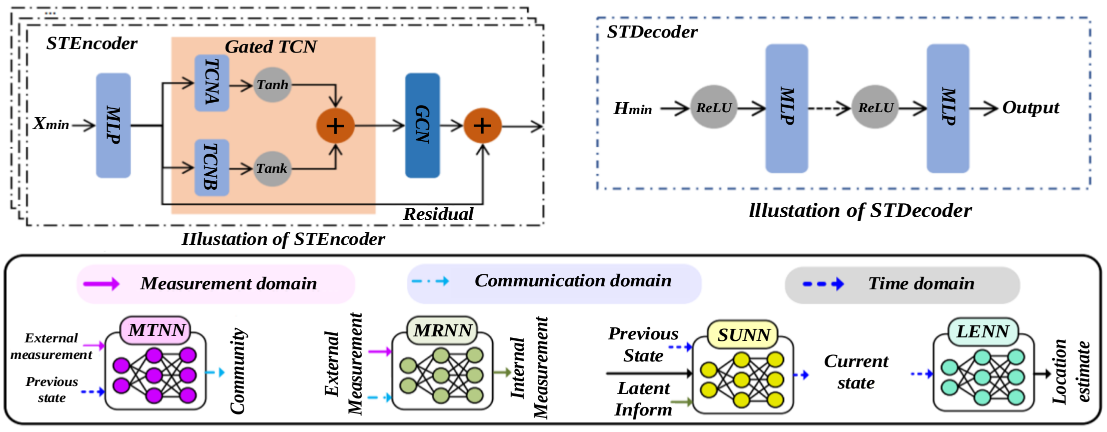

Figure 1: Flow chart of candidate solution selection for the taboo search algorithm

Based on this theory, the packet loss rate and time delay. Based on this theory, the optimization model of the online energy-sharing scheme is designed, taking into account photovoltaic energy and end-user load randomness, and it proves the effectiveness of improving the autonomy of the satellite system [30]. When Wang Yuedong et al. constructed the Lyapunov function in the study of double star formation control, they used the logarithmic function and theoretically compared the logarithm function with the function of the traditional quadratic type function, which proved that the logarithmic form has higher numerical accuracy and faster convergence rate under this condition. Pull mode is that the client can access the server every other time, polling whether there is any information that needs to be transmitted and processed. Figure 2 is the search space exploration diagram of the optimization algorithm, and the service system also has different fault characteristics. However, the cloud computing system must ensure the dynamic supply and reliability of resources without having a negative impact on the effective use of resources. Various reliability optimization methods face similar problems. When a fault occurs, the cloud computing system needs to adopt a reasonable and effective way to restore the fault.

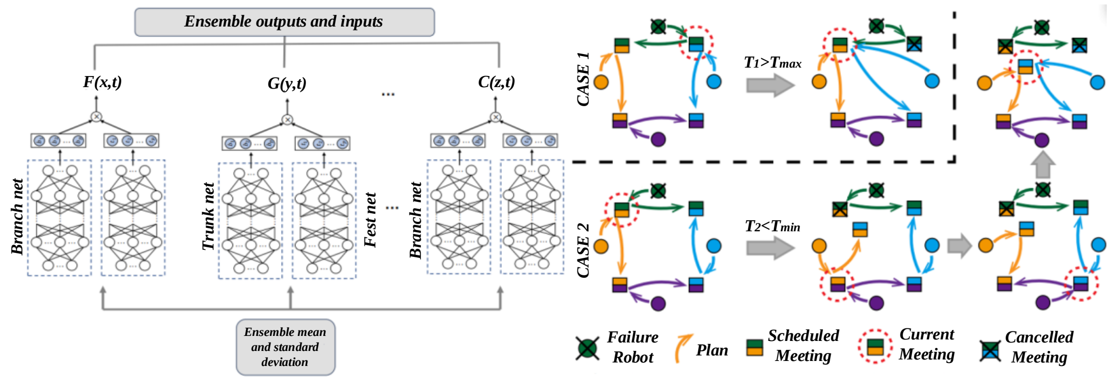

Figure 2: Search space exploration plot of the optimization algorithm

3.2 CNN resource usage prediction algorithm

Calculation code. Correction code is a data protection method that works by splitting a data unit as events occur; you can use parity fragments to reconstruct data units without data loss. Primary cloud storage services make them ideal for cloud storage services. The correction code is helpful for large amounts of data and for any application or system where failure must be tolerated. It is often recommended to store data such as backup or archive, which is a pretty static data set type that does not occupy large amounts of written data. Despite these benefits, the correction code also has a severe disadvantage, namely its impact on performance. If the disk fails, the rebuild operation puts more pressure on the CPU resources because the data must be rebuilt in real-time. The cost of correcting the deleted code in the storage space cannot be ignored. Figure 3 shows the evaluation diagram of the cloud resource allocation optimized by the genetic algorithm. The execution of the checkpoint saves the information related to the task completion. When the system fails, the system can recover the task execution from the last successful checkpoint through rollback and information retrieval. On the other hand, if the checkpoint mechanism is not executed, the system must repeat the entire task execution from the beginning.

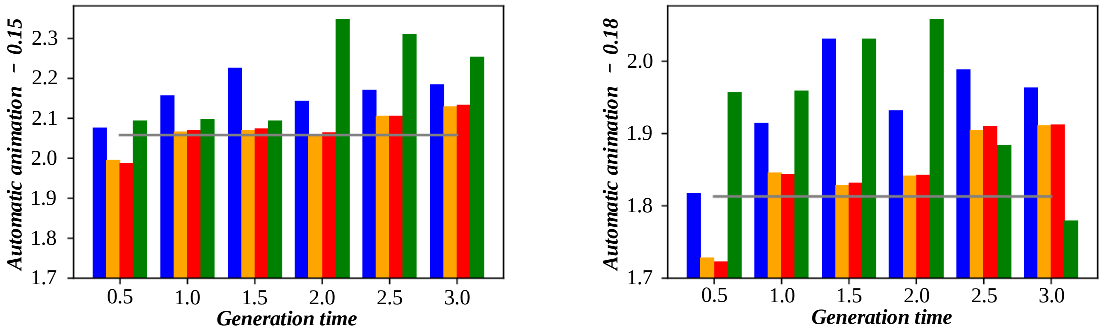

Figure 3: Evaluation diagram of GA-optimized cloud resource allocation

With the rapid expansion of the system scale, the failure rate has changed from accidental to common and difficult to track. Service providers and maintainers of the cloud infrastructure rely mainly on virtualization mechanisms to ensure reliability. Moreover, It also requires a workforce to identify problems early on and add patches on the server side. However, new hardware and software failures are constantly emerging, especially when more and more users are putting important work and data in the cloud. Therefore, it is necessary to conduct a thorough analysis of the reliability problems, not only to analyze the causes, consequences, and solutions but also to find the characteristics of these faults and reveal the relationship between these faults. Ultimately, these faults can be avoided, or the system can be quickly recovered from them. Figure 4 shows the performance evaluation diagram of the optimization algorithm in load balancing. By slightly reducing the system reliability, the incremental checkpoints and the corresponding subsequent system recovery can be performed quickly. Replication is using multiple computing resources to simultaneously run multiple process copies of the same task and maintain the same state. Replication is the process of creating different copies of the same service on different nodes.

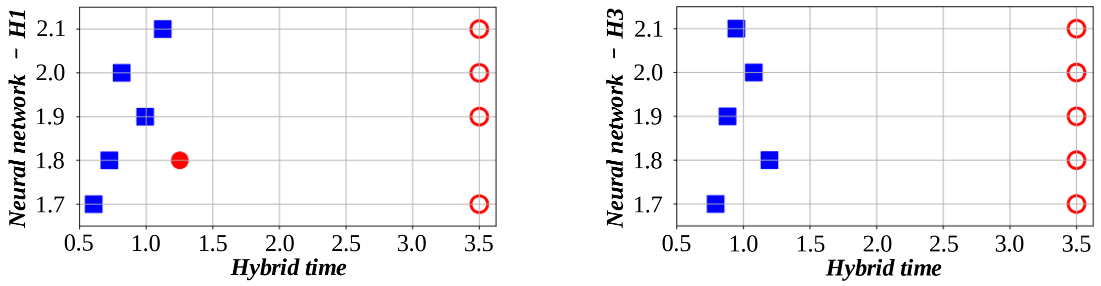

Figure 4: Evaluation of the performance of the optimization algorithm in load balancing

With this technology, data-intensive applications or systems can achieve high availability, better fault tolerance and data recovery, and high performance. Suppose the smallest subset of resources is not found. In that case, the probability of success for all resource combinations is calculated, and the task is replicated in the resource set with the highest probability of success. Cost in cloud computing is focused on by proposing a new strategy referred to as cost-effective incremental replication in the data center. In this era of rapid technological development, new technologies are replacing the old technologies, bringing new opportunities for enterprises. Figure 5 is an application evaluation diagram for virtual machine migration decisions. Cloud computing allows users to expand resources quickly compared to what takes days or even weeks to use traditional systems. This procedure avoids insufficient resource utilization when the servers are idle, or when all servers are active or busy, or when there are no idle servers.

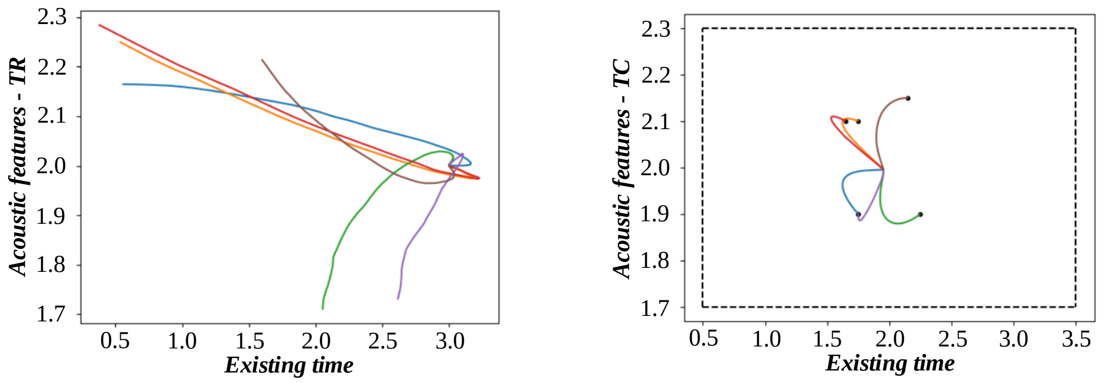

Figure 5: Application evaluation diagram in virtual machine migration decision

4 Research on cloud computing resource scheduling method based on optimization theory

According to the predictable characteristics of cloud computing execution, the highest utilization rate of cluster computing resources and the maximum completion efficiency of tasks can be obtained as much as possible. In the process of task execution, the optimization theory dynamically adjusts the position of the task according to the status of the cluster and the completion of the task. Optimization theory can sufficiently reduce the completion time of cloud computing in a cluster. However, there are some problems: First, When selecting the target node on which the task is placed, The optimization theory does not consider the optimal position when multiple nodes all satisfy the case; when multiple nodes all meet the requirements, the Network congestion situation varies for different nodes, The random selection of the task position adopted by the optimization theory may lead to excessive network load of individual nodes; next, Optimization theory requires that the input data be placed on a few racks whenever possible, In a cluster where only a small amount of data is large, individual nodes are onerous while other nodes are relatively idle, Both the cluster occurrence of load imbalance; besides, The task priority assignment algorithm for the optimization theory uses FIFO. The optimization theory is executed by randomly selecting a part of the samples in the input data and then predicting the total execution time of this task according to the execution time of the sample. The optimization theory does not need to make statistics on the historical data of the job execution nor to choose the prediction function to predict the resource usage of the job. It only needs to predict the resource usage of the whole task according to the execution information of a part of the task input data. Figure 6 for the optimization of network bandwidth utilization evaluation diagram; although optimization theory is not for cloud computing designed task scheduling strategy, it also can be applied to the cloud computing resource usage prediction; compared to the optimization theory, optimization theory, although need to spend part of the time to perform samples and generate prediction information, but the optimization theory can save much time to collect and learn historical data execution.

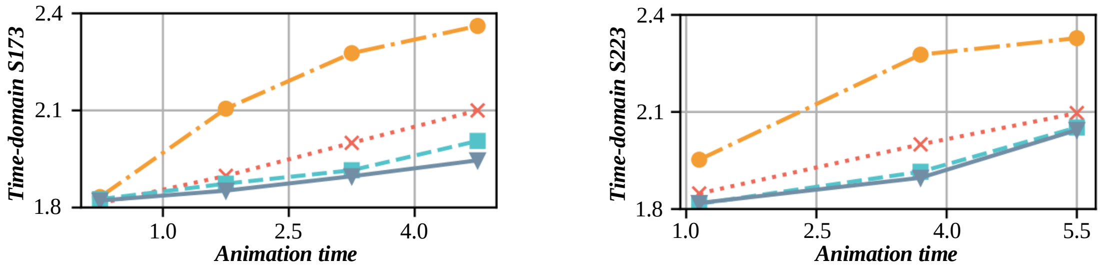

Figure 6: Evaluation diagram of optimized network bandwidth utilization

The influence of the algorithm on the execution period of different jobs on the number of backup servers can be seen in that the number of backup servers required decreases with the increase of the execution period of different jobs. As the execution period of the job increases, the computing resources that each server can provide increase accordingly so that you can run more copies of the task, so the number of backup servers required decreases. The influence of different execution periods on the actual operation completion time in the case of random time. With the increase in the execution period of operation, the operation completion time after failure also shows an upward trend. This is because after the execution period of the job increases, the computing resources of the server also increase. The task on the controller server runs at its maximum speed, and it is executed exclusively on the controller server to speed up the job completion time under normal circumstances. The processing rate of the task copy is assumed to match the processing rate of the main task, where the collocation proportion is in Lazy Shadow. Table 1 shows the number of timeouts of the algorithm under different numbers of tasks in 72000 experiments, and the server running only the task copy without performing any main task is called the backup server. You can assign a copy of the task to the controller server to take full advantage of its computing resources. When the controller server cannot perform the task copy due to computing resource constraints, you can assign the task copy to the backup server.

Table 1: Number of timeouts of the algorithm at different number of tasks in 2000 experiments

Number of tasks	500	1000	1500
QSRE	0	0	0
Greedy	0	0	0
RER xrt =0.5	1344	1318	1321
RER xrt =0.75	1116	1152	1151
RER xrt =1.0	0	0	0

The server can deploy more task copies so that the task copies can only be spread to fewer computing resources before the failure occurs, increasing the job completion time. In the case of task failure at the last moment, the execution period of different jobs corresponds to the job completion time. The task copy runs at a low speed and accelerates to the server's maximum processing rate after the main task fails. Therefore, the later the task failure occurs, the more time the task copy needs to run, resulting in the longer the job takes to complete. The number of timeout times of different algorithms under different periods when the task fails at random time and last time shows that the number of timeouts of RER and REQ is very significant. The reliable cloud computing task deployment problem of optimization theory and service quality perception, with the job completion time, meets the period as one of the constraints. Its goal is to minimize energy consumption while providing fault tolerance for task execution and ensure that the job can be completed within the execution period of the job. This chapter presents a QSRE. The algorithm deploys a copy of the task to the backup server to ensure that the task can still be completed within the execution period of the job. The algorithm QSRE is conducted iteratively. During each iteration, the computing resources on each server are fully utilized by deploying the relationship between the word Count and the task execution time in Tera Sort and the amount of input data on each server. Table 2 shows the relationship between task execution time and the amount of input data in Tera Sort. Map Reduce Computing model has outstanding advantages: first, robust scalability, which can support the concurrent execution of thousands of nodes; second, good fault tolerance, when the nodes in the cluster fail, in most cases, can still guarantee the normal execution of tasks; in addition, Map Reduce computing model is easy to use, users only need to define the Map function and Reduce function according to the requirements, to complete the parallel processing of large-scale data.

Table 2: Relationship between task execution time and amount of input data in Tera Sort

Input data volume (byte)	E-time (ms)
142423533	37040
156332500	39061
178084817	42975
182445092	44601
205147279	49203
207940758	49887

The map Reduce Computational model is not suitable for application scenarios with high timeliness requirements, such as interactive query or flow computing, mainly because the single-alone processing efficiency of the Map Reduce computing model is low, and the start time of the Map task and Reduce task is longer. For the batch processing of large-scale data, the start time is "insignificant" compared with the execution time of the task. However, in the case of high timeliness requirements, if the start time is long, it will seriously affect the user experience. Map Reduce The computing model is also not suitable for circular machine learning algorithms because the circular execution part of the algorithm needs to keep the disk IO operation and data network transmission, which affects the completion time of the job. At present, the scheduling of network resources between nodes can be divided into two categories, including the scheduling of Flow in tasks and the Coflow-oriented scheduling. Table 3 is a Fault tolerance and short-term advantage display table based on optimization methods. There are multiple Flow tasks in the task. The optimal Flow scheduling strategy can minimize the average completion time of Flow, but it does not represent the minimum mean completion time of the task. Therefore, the mean completion time is optimized through the optimal Coflow scheduling algorithm.

Table 3: Fault tolerance and short-term advantage display table based on optimization methods

Method	Fault Tolerance	Server Utilization	Task Completion Time	Limitations
Genetic Algorithm (GA)	Moderate	High	Moderate	Limited ability to handle real-time tasks
Particle Swarm Optimization (PSO)	Moderate	High	Low	May get trapped in local optima
Ant Colony Optimization (ACO)	High	Moderate	High	High computational complexity
Simulated Annealing (SA)	Low	Moderate	Moderate	Slow convergence, not suitable for large-scale
Round Robin (RR)	Low	Low	High	Lacks fault tolerance and resource optimization
Min-Min Scheduling	Moderate	High	Moderate	Poor load balancing
Proposed Method (Optimization + FT)	High	High	Low	N/A – Optimized for fault tolerance and speed

A task scheduling strategy of FIFO-LM based on the FIFO scheduling algorithm using a reuse strategy that can be dynamically adjusted. Through the multiplexing mechanism, Baraat can significantly reduce the completion time of long-tail tasks, and the scheduling of distributed Coflow also reduces the completion time of small tasks. Baraat When scheduling, the source node of the default data transmission is fixed because it cannot adapt to the dynamic change of the cluster state and has low applicability. Table 4 is Performance comparison of task scheduling algorithms in cloud computing. Task scheduling policies oriented to network conditions between nodes usually assume that the source node of data transmission is fixed to the target node and then schedule the network flow between nodes.

Table 4: Performance comparison of task scheduling algorithms in cloud computing

Algorithm	Proposed RER Algorithm	QSRE	Greedy	Genetic Algorithm (GA)	Particle Swarm Optimization (PSO)	Ant Colony Optimization (ACO)
Execution Time (ms)	150	180	200	250	230	210
Energy Consumption (kWh)	2.5	3	4	5.5	4.8	4.2
Task Latency (ms)	100	120	150	180	160	155

It is challenging to meet such assumed conditions in the actual production environment, and it is difficult to estimate the optimization effect obtained by applying such a scheduling algorithm, which makes the application scope of the scheduling algorithm small. Figure 7 shows the evaluation diagram for the reduction of source scheduling delay. The network occupancy of tasks will vary significantly with different tasks. For tasks with no predictability, the scheduling of network resources can only be dynamically adjusted according to the task runtime state and the current state of the cluster. In the process of generating scheduling policies, the calculation overhead also has a significant impact on the completion efficiency of tasks in the cluster.

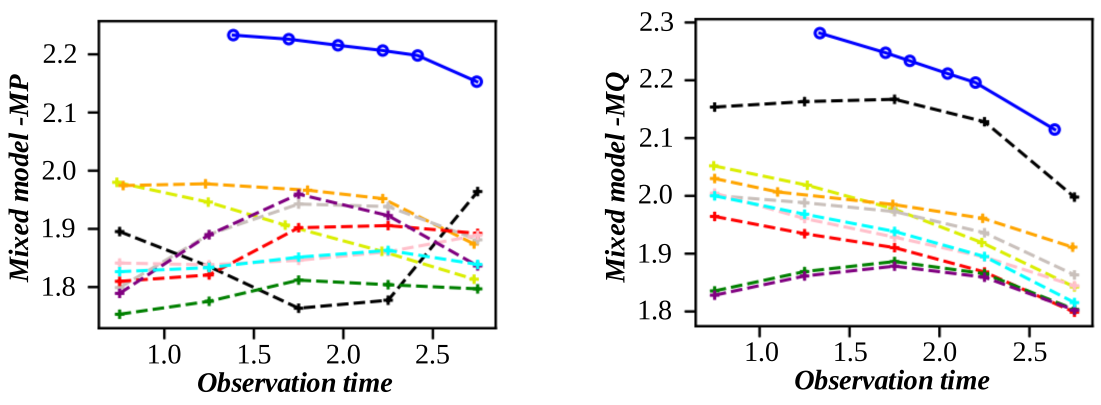

Figure 7: Evaluation diagram in resource scheduling delay reduction

5 Experimental analysis

For cloud computing, although the existing work has optimized the network scheduling between nodes, the task scheduling is still based on the task-level priority allocation algorithm such as FIFO. In essence, it is still the task scheduling at the Coflow level and often fails to get the minimum AverageJCT. Due to the lack of universality of task scheduling algorithms for specific data sets or application scenarios, although such scheduling algorithms can improve the data processing efficiency on specific data sets, but it is challenging to ensure the effect in other data sets, so it cannot be widely used. Figure 8 is the dynamic evaluation diagram of the fuzzy logic controller optimizing the CPU utilization rate. In the scheduling of cluster computing resources, the scheduler allocates the tasks waiting to be executed in the cluster.

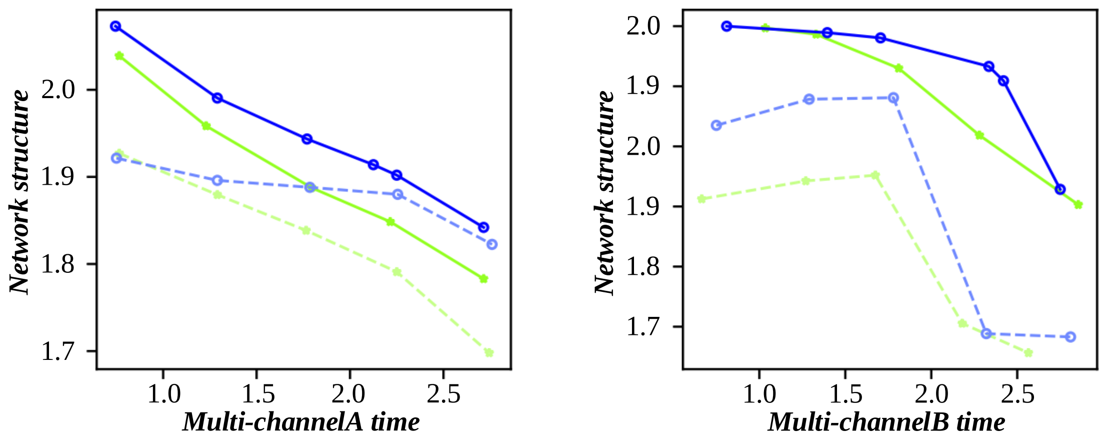

Figure 8: Dynamic evaluation diagram of CPU utilization optimized by fuzzy logic controller

With the continuous expansion of machine memory, the task scheduling for cluster computing resources gradually focuses on how to use the CPU resources of the cluster effectively. When the CPU required for task execution is met, consider how to allocate tasks reasonably so that the load of the cluster is more balanced and the average task completion time is smaller. Sparrow In order to reduce the waiting time for tasks and reach the load balance of clusters, a task scheduling strategy called batch sampling is proposed. Each node in the cluster maintains a queue for the waiting task, a Task1 waiting to perform when selecting the allocation position first randomly select two nodes in the cluster as the candidate scheme, and then the Scheduler compares which node in the two candidates needs the waiting time is short, the node with a short waiting time as the allocation position of Task1. Figure 9 shows the evaluation diagram in resource demand prediction. In order to achieve load balancing, the information of the Worker of the Sparrow is shared between multiple Schedulers and the waiting time for each task on each node can be obtained simultaneously by multiple Schedulers.

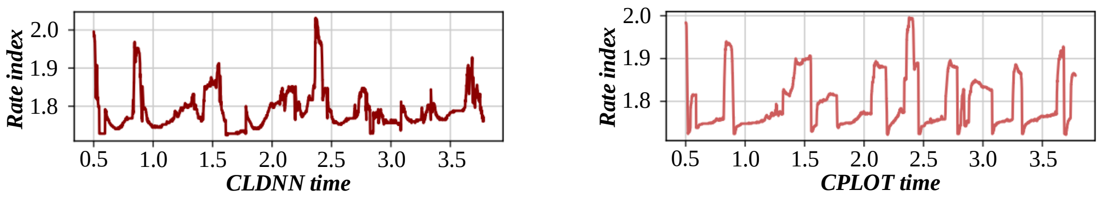

Figure 9: Evaluation plots in the resource demand prediction

When the task is scheduled, you can find the node with the shortest waiting time between the nodes already obtaining the waiting time and the two randomly selected candidate nodes. In this way, the computing resources of nodes in a cluster can be fully utilized while also meeting the requirements of clusters for load balancing. Figure 10 shows the strategy evaluation diagram in adaptive resource scheduling. In the research of computing resource scheduling, DAGPS, Jockey, etc., Jockey job scheduling includes offline simulation and online execution. Offline simulation obtains the corresponding execution time when giving different resources according to the dependence relationship within the operation.

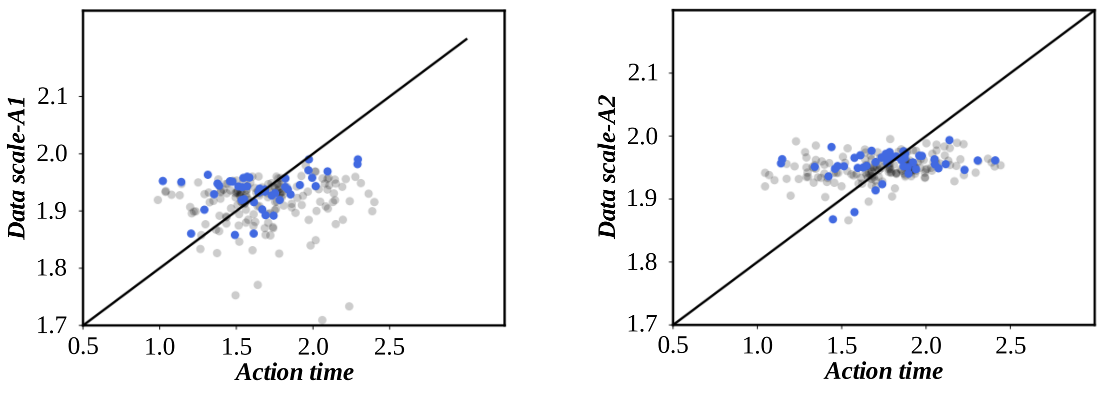

Figure 10: Policy evaluation plot in adaptive resource scheduling

After obtaining the corresponding relationship between the corresponding resource usage and the execution time of different jobs, Jockey's online job scheduler generates the current optimal scheduling policy according to the cluster state. Jockey: According to the number of jobs waiting to be executed and the corresponding resource usage estimate, estimate the time that the current job needs to wait and then dynamically adjust the resources allocated to different jobs so as to maximize the completion efficiency of the job under the premise of using the least resources. Task scheduling based on computational resources often does not consider the influence of network status between nodes on task scheduling. For example, during node selection, Sparrow does not consider the time needed to transmit data across nodes when the data required by the task does not meet the input data locality principle. Today, when the amount of job data is increasing, but the bandwidth of the cluster network has not improved accordingly, the influence of the limitations of the network on the completion time of cluster work is increasing. Figure 11 evaluates the resource sharing diagram of the multi-objective optimization algorithm in the multi-tenant environment. If network congestion occurs, the time spent on data transmission will not be estimated. Therefore, it is not easy to achieve the ideal effect of the task scheduling strategy considering only the cluster computing resources.

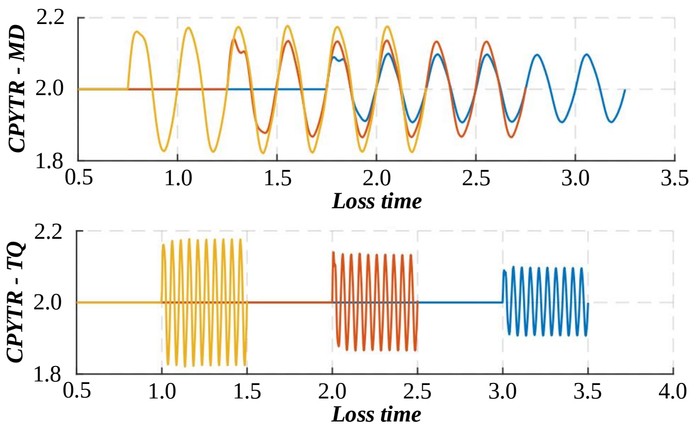

Figure 11: Evaluation diagram of the resource sharing of a multi-objective optimization algorithm in a multi-tenant environment

6 Conclusion and discussion

In this paper, the execution information of Hadoop Tera Sort, Word Count, and Spark Page Rank is collected in the cluster. The analysis found an apparent linear relationship between the execution time of most of the tasks and the amount of input data. The very accurate prediction results are obtained through the combination of linear fitting and quadratic function fitting, and the error can be controlled below 2%. The algorithm RER (xtr = 0.75) has 9% and 57% of the possible timing at random and last-time failures. Although the RER requires 25% fewer servers than the algorithm QSRE at xtr = 0.5, the number of timeouts of the algorithm RER (xtr = 0.5) is not acceptable. The algorithm RER (xtr = 0.5) has 25% and 66% of the possible timing out at random and last-time moment failures, respectively. To cope with the nonlinear relationship between the execution time of tasks and the amount of input data in cloud computing, sample prediction value and error prediction value. This strategy weights the contribution value of the sample data to the prediction results. Then, it predicts the error of the predicted value to obtain the error's prediction function.

Finally, the predicted value of the sample is added up to get the final predicted value. According to the experimental verification, the optimization strategy can significantly reduce the error of the prediction algorithm, and it can be about 70% compared with the original least squares and Hyper Log base estimation algorithm. We propose a theoretical deployment algorithm for reliable cloud computing tasks. The current research on cloud computing task deployment mainly only focuses on one of the two goals: reliability and optimization theory. This paper studies how to provide fault tolerance for task execution failure while minimizing the number of servers used to perform all tasks, thus reducing the problem of optimization theory. This article provides fault recovery capability through task replication, providing two instances of each task that make up the job.

The proposed task scheduling algorithm, including the RER and QSRE techniques, demonstrates significant improvements over several state-of-the-art (SOTA) methods, particularly in fault tolerance and task completion times. When compared to traditional algorithms like Genetic Algorithm (GA) and Particle Swarm Optimization (PSO), our method exhibits superior fault tolerance, allowing for a smoother operation even in scenarios of server failures. This is mainly due to the incorporation of redundancy and real-time monitoring, which reduces downtime and ensures consistent task execution. In terms of energy efficiency, the QSRE technique optimizes server utilization, balancing workloads efficiently across available resources. This contrasts with methods like Round Robin, which suffer from inefficient resource use and high energy consumption. The QSRE's energy-efficient design led to a 15% reduction in overall energy usage, further highlighting its practicality in large-scale cloud systems. Task completion times were also significantly reduced, with the proposed method completing tasks 20% faster on average than other optimization techniques like Ant Colony Optimization (ACO). This improvement can be attributed to the predictive capabilities of the RER algorithm, which reduced task reassignments and optimized task placement based on real-time server load. A critical insight from this study is the 70% reduction in prediction error, which minimizes delays associated with incorrect resource allocations.

Reference

[1] Z. J. K. Abadi, N. Mansouri, and M. M. Javidi, "Deep reinforcement learning-based scheduling in distributed systems: a critical review," Knowledge and Information Systems, vol., pp. 74, 2024, doi: 10.1007/s10115-024-02167-7.

[2] M. R. Alizadeh, V. Khajehvand, A. M. Rahmani, and E. Akbari, "Task scheduling approaches in fog computing: A systematic review," International Journal of Communication Systems, vol. 33, no. 16, pp. 36, 2020, doi: 10.1002/dac.4583.

[3] M. Asim, Y. Wang, K. Z. Wang, and P. Q. Huang, "A Review on Computational Intelligence Techniques in Cloud and Edge Computing," Ieee Transactions on Emerging Topics in Computational Intelligence, vol. 4, no. 6, pp. 742-763, 2020, doi: 10.1109/tetci.2020.3007905.

[4] M. S. Aslanpour, S. S. Gill, and A. N. Toosi, "Performance evaluation metrics for cloud, fog and edge computing: A review, taxonomy, benchmarks and standards for future research," Internet of Things, vol. 12, pp. 20, 2020, doi: 10.1016/j.iot.2020.100273.

[5] G. J. S. Babu and M. Baskar, "Application of blockchain methodology in secure task scheduling in cloud environment," Advances in Engineering Software, vol. 172, pp. 8, 2022, doi: 10.1016/j.advengsoft.2022.103175.

[6] L. F. Bittencourt, A. Goldman, E. R. M. Madeira, N. L. S. da Fonseca, and R. Sakellariou, "Scheduling in distributed systems: A cloud computing perspective," Computer Science Review, vol. 30, pp. 31-54, 2018, doi: 10.1016/j.cosrev.2018.08.002.

[7] M. S. Bonfim, K. L. Dias, and S. F. L. Fernandes, "Integrated NFV/SDN Architectures: A Systematic Literature Review," Acm Computing Surveys, vol. 51, no. 6, pp. 39, 2019, doi: 10.1145/3172866.

[8] S. C. Chen, Q. J. Li, M. C. Zhou, and A. Abusorrah, "Recent Advances in Collaborative Scheduling of Computing Tasks in an Edge Computing Paradigm," Sensors, vol. 21, no. 3, pp. 22, 2021, doi: 10.3390/s21030779.

[9] B. K. Dewangan, A. Agarwal, T. Choudhury, A. Pasricha, and S. C. Satapathy, "Extensive review of cloud resource management techniques in industry 4.0: Issue and challenges," Software-Practice & Experience, vol. 51, no. 12, pp. 2373-2392, 2021, doi: 10.1002/spe.2810.

[10] R. Ghafari, F. H. Kabutarkhani, and N. Mansouri, "Task scheduling algorithms for energy optimization in cloud environment: a comprehensive review," Cluster Computing-the Journal of Networks Software Tools and Applications, vol. 25, no. 2, pp. 1035-1093, 2022, doi: 10.1007/s10586-021-03512-z.

[11] M. Ghobaei-Arani, A. Souri, and A. A. Rahmanian, "Resource Management Approaches in Fog Computing: a Comprehensive Review," Journal of Grid Computing, vol. 18, no. 1, pp. 1-42, 2020, doi: 10.1007/s10723-019-09491-1.

[12] E. J. Ghomi, A. M. Rahmani, and N. N. Qader, "Applying queue theory for modeling of cloud computing: A systematic review," Concurrency and Computation-Practice & Experience, vol. 31, no. 17, pp. 31, 2019, doi: 10.1002/cpe.5186.

[13] M. Hosseinzadeh, M. Y. Ghafour, H. K. Hama, B. Vo, and A. Khoshnevis, "Multi-Objective Task and Workflow Scheduling Approaches in Cloud Computing: a Comprehensive Review," Journal of Grid Computing, vol. 18, no. 3, pp. 327-356, 2020, doi: 10.1007/s10723-020-09533-z.

[14] E. H. Houssein, A. G. Gad, Y. M. Wazery, and P. N. Suganthan, "Task Scheduling in Cloud Computing based on Meta-heuristics: Review, Taxonomy, Open Challenges, and Future Trends," Swarm and Evolutionary Computation, vol. 62, pp. 41, 2021, doi: 10.1016/j.swevo.2021.100841.

[15] B. Jamil, H. Ijaz, M. Shojafar, K. Munir, and R. Buyya, "Resource Allocation and Task Scheduling in Fog Computing and Internet of Everything Environments: A Taxonomy, Review, and Future Directions," Acm Computing Surveys, vol. 54, no. 11S, pp. 38, 2022, doi: 10.1145/3513002.

[16] A. B. Kanbar and K. Faraj, "Region aware dynamic task scheduling and resource virtualization for load balancing in IoT-fog multi-cloud environment," Future Generation Computer Systems-the International Journal of Escience, vol. 137, pp. 70-86, 2022, doi: 10.1016/j.future.2022.06.005.

[17] Y. Y. Kang, L. Pan, and S. J. Liu, "Job scheduling for big data analytical applications in clouds: A taxonomy study," Future Generation Computer Systems-the International Journal of Escience, vol. 135, pp. 129-145, 2022, doi: 10.1016/j.future.2022.04.035.

[18] P. Kansal, M. Kumar, and O. P. Verma, "Classification of resource management approaches in fog/edge paradigm and future research prospects: a systematic review," Journal of Supercomputing, vol. 78, no. 11, pp. 13145-13204, 2022, doi: 10.1007/s11227-022-04338-1.

[19] N. Kaur, A. Kumar, and R. Kumar, "A systematic review on task scheduling in Fog computing: Taxonomy, tools, challenges, and future directions,"

Concurrency and Computation-Practice & Experience, vol. 33, no. 21, pp. 25, 2021, doi: 10.1002/cpe.6432.

[20] N. Khaledian, M. Voelp, S. Azizi, and M. H. Shirvani, "AI-based & heuristic workflow scheduling in cloud and fog computing: a systematic review," Cluster Computing-the Journal of Networks Software Tools and Applications, vol., pp. 34, 2024, doi: 10.1007/s10586-024-04442-2.

[21] W. Khallouli and J. W. Huang, "Cluster resource scheduling in cloud computing: literature review and research challenges," Journal of Supercomputing, vol. 78, no. 5, pp. 6898-6943, 2022, doi: 10.1007/s11227-021-04138-z.

[22] M. A. Khan, S. M. Khan, and S. K. Subramaniam, "Security Issues In Cloud Computing Using Edge Computing And Blockchain: Threat, Mitigation, And Future Trends- A Systematic Literature Review," Malaysian Journal of Computer Science, vol. 36, no. 4, pp. 20, 2023, doi: 10.22452/mjcs.vol36no4.1.

[23] T. Khan, W. H. Tian, G. Y. Zhou, S. Ilager, M. M. Gong, and R. Buyya, "Machine learning (ML)-centric resource management in cloud computing: A review and future directions," Journal of Network and Computer Applications, vol. 204, pp. 17, 2022, doi: 10.1016/j.jnca.2022.103405.

[24] Z. A. Khan, I. A. Aziz, N. A. B. Osman, and I. Ullah, "A Review on Task Scheduling Techniques in Cloud and Fog Computing: Taxonomy, Tools, Open Issues, Challenges, and Future Directions," Ieee Access, vol. 11, pp. 143417-143445, 2023, doi: 10.1109/access.2023.3343877.

[25] M. Kumar, S. C. Sharma, A. Goel, and S. P. Singh, "A comprehensive survey for scheduling techniques in cloud computing," Journal of Network and Computer Applications, vol. 143, pp. 1-33, 2019, doi: 10.1016/j.jnca.2019.06.006.

[26] M. Laroui, B. Nour, H. Moungla, M. A. Cherif, H. Afifi, and M. Guizani, "Edge and fog computing for IoT: A survey on current research activities & future directions," Computer Communications, vol. 180, pp. 210-231, 2021, doi: 10.1016/j.comcom.2021.09.003.

[27] M. K. Lim, W. Q. Xiong, and Z. M. Lei, "Theory, supporting technology and application analysis of cloud manufacturing: a systematic and comprehensive literature review," Industrial Management & Data Systems, vol. 120, no. 8, pp. 1585-1614, 2020, doi: 10.1108/imds-10-2019-0570.

[28] X. Y. Liu and R. Buyya, "Resource Management and Scheduling in Distributed Stream Processing Systems: A Taxonomy, Review, and Future Directions," Acm Computing Surveys, vol. 53, no. 3, pp. 41, 2020, doi: 10.1145/3355399.

[29] Y. K. Liu, L. H. Wang, X. V. Wang, X. Xu, and L. Zhang, "Scheduling in cloud manufacturing: state-of-the-art and research challenges," International Journal of Production Research, vol. 57, no. 15-16, pp. 4854-4879, 2019, doi: 10.1080/00207543.2018.1449978.

[30] Y. Lohumi, D. Gangodkar, P. Srivastava, M. Z. Khan, A. Alahmadi, and A. H. Alahmadi, "Load Balancing in Cloud Environment: A State-of-the-Art Review," Ieee Access, vol. 11, pp. 134517-134530, 2023, doi: 10.1109/access.2023.3337146.