Publication Date: 11 March 2024
Archs Sci. (2024) Volume 74, Issue 1 Pages 26-34, Paper ID 2024103.
https://doi.org/10.62227/as/74103

Resource Allocation Scheduling Optimization Based on Internet of Things under Cloud Platforms

Ying Xie[1,*]
[1]Minxi Vocational & Technical College, Longyan, Fujian, 364021, China.
Corresponding authors: Ying Xie (e-mail: sisi123xx@sina.com).

Abstract In this paper, we explore how to optimize IoT-based resource allocation and scheduling in a cloud platform environment, focusing on improving computing resource utilization and quality of service, while reducing latency and packet loss. a model is adopted, which contains a number of edge servers and randomly generated computational tasks, taking into account the network conditions between the servers and the tasks. an objective function is established, aiming to maximize the computational resource utilization and QoS, and the corresponding constraints are proposed. Simulations are conducted using CloudSim, and the experimental results show that the total number of VoCS increases from 243.63 to 1397.71 when the scheduling demand is increased from 8 to 64, demonstrating the adaptability and efficiency of the algorithm under different demands. In addition, the algorithm is effective in dealing with both small-scale (200 tasks) and large-scale (6000 tasks) tasks. In addition, the algorithm demonstrates low load imbalance and short task completion time when dealing with both small-scale (200 tasks) and large-scale (6000 tasks) task sets, which proves its effectiveness. Ultimately, the scheduling method proposed in this study not only improves resource utilization and quality of service, but also reduces task completion time and cost.

Index Terms Cloud platforms, IoT, edge servers, network conditions

I. Introduction

In recent years, based on the development of intelligent sensing technology and information and communication technology, the Internet of Things (IoT) technology is rapidly entering and profoundly influencing people’s production and consumption activities [1]–[3], from the RFID technology that realizes intelligent sensing, to the smart terminals that can be seen anytime, anywhere, to the emergence of the theory of "cloud service" and its practical application [4], [5].

As an emerging strategy in various countries, IoT has certain research significance from both academic and practical perspectives. From the point of view of current applications, IoT technology has been widely used in various industries in many fields, such as manufacturing industry, transportation and distribution, pharmaceuticals and so on, which can be regarded as a promising prospect [6], [7]. The rapid development of many platforms in the Internet is a sign of the wide application of IoT platforms that have taken shape. In the environment of Internet of Things, the supply chain enterprises are in the process of "collaborating and sharing" concept. Under the Internet of Things environment, supply chain enterprises can share the available capacity information of facilities and resources through the cloud platform, and in the process of transformation from insufficient information sharing to sufficient information sharing, the resources of supply chain enterprises will then undergo a new allocation strategy [8]–[10].

With the maturity of the Internet of Things (IoT) technology, this technology is gradually emerging in the fragmented resource integration services, and cloud manufacturing.

Literature [11] investigates the joint radio and computational resource allocation problem, aiming to optimize the system performance and customer satisfaction, and finally applies the UOC strategy, which effectively improves the performance of the system. Literature [12] proposes a computational resource allocation strategy based on drone-assisted edge computing for IoT 5G communication, which improves the allocation efficiency compared with the traditional method. A new distributed block-based q-learning algorithm is conceptualized in [13] for slot scheduling of smart devices and machine-based communication devices (mtcd) in clustered IoT networks, and the convergence of the optimized allocation mechanism is improved as confirmed by simulation experiments. A fuzzy logic offloading strategy is envisioned in [14] to be incorporated into IoT applications with uncertain parameters to improve the performance of the system. IoT applications to optimize the protocol exponentiation and robustness, and the effectiveness of the proposed method is confirmed by the study of benchmark problems and simulation experiments.

Literature [15] conceived an evolutionary algorithm using the function Reallocate to quickly converge to the global extremes to solve the new Real-RCPSP, and the effectiveness of the algorithm is verified by the test on the dataset while the efficiency of TNG's production line can be improved. Improve the production line efficiency of TNG company. Literature [16] introduced the characteristics, current status and future development trend of cloud manufacturing and elaborated the five development directions of CMfg. By collecting and organizing the related articles, it aims to provide references and directions for the future related researches. Literature [17] disassembled the UAV optimization problem into three independent sub-problems, UAV localization,UAV altitude optimization and The effectiveness and practicality of the SOTA algorithm is verified based on simulation and comparative experiments.

Literature [18] envisioned a centralized network controller (CNC), docking UAVs for resource data dissemination to cope with the challenges of regional network failures. A model-free deep reinforcement learning (DRL) cooperative computational offloading and resource allocation (cora -DRL) Literature [19] proposes a many-to-many matching-based model to rationally allocate fog node (FN) resources based on the VNF resource demand of adss, and simulation experiments are used to prove that the proposed method can realize the higher education allocation of resources.

Literature [20] proposes a matching game-based network resource allocation strategy for solving the problem of joint user association and resource allocation in the downlink of the fog network, which is based on the matching game. Network resource allocation strategy based on matching game is proposed, and the method is subjected to simulation experiments, and it is found that the algorithm has high resource allocation efficiency, while the user association is stable and has a high effect gain.

According to the domestic and foreign literature, there are still some problems in the scheduling of the Internet of things, such as the ability to expand the wireless network resources, the traditional network resource scheduling scheme will not be able to meet the quality requirements of the diversified services generated by the lateral expansion of the industry. Therefore, the optimization of resource scheduling scheme is an important means to further improve the systematic ability of the Internet of things.

In this study, real-time monitoring and time-delay optimization of emergency data and the problem of low utilization of network resources, we adopt a resource allocation scheduling model for IoT based on cloud platform, by setting the edge server and computing task parameters, and considering the network conditions between tasks and servers, such as latency and packet loss. We establish an objective function focusing on computational resource utilization and quality of service (QoS), and formulate the corresponding constraints.

The algorithms are validated using the CloudSim simulation environment, and the performance of the algorithms under different sizes of task sets (small and large scale) is analyzed through simulation experiments, focusing on key metrics such as load balancing degree, power consumption, and cost. The algorithms are validated using the CloudSim simulation environment, and the performance of the algorithms is analyzed by simulation experiments under different sizes of task sets (small and large), focusing on key metrics such as the degree of load balancing, task completion time, power consumption, and cost, etc. This methodology aims to find a balance point for achieving optimal performance in resource allocation and scheduling.

II. Method

A. Resource Allocation Scheduling Model for Internet of Things Based on Cloud Platforms

In order to facilitate the calculation and understanding, let the whole IoT system have *m* edge servers with *n* randomly generated computational tasks, where the set of tasks is denoted as $T = \{T_1, T_2, T_3, \ldots\ldots T_i, \ldots\ldots T_n\}$, and the resources required for the *i*th task to be executed are $\{R_{ic}, R_{ib}\}$, where R_{ic} denotes the computational resources required for the *i*th task to be executed, and R_{ib} denotes the bandwidth required for the *i*th task to be executed, and the set of edge servers is denoted as $H = \{H_1, H_2, H_3, \ldots\ldots H_j, \ldots\ldots H_m\}$, where the computational resources for the jth edge server is denoted as C_j, and the bandwidth is denoted as B_j.

There is a possibility of delay and packet loss between the task and the edge server, set the network-like sink matrix between the *i*th task and the *j*th edge server as $D\,(ms)$ delay(delay), L packet loss(lossrate), where, $1 \le i \le n$, $1 \le j \le m$.

Establish the task resource matching $N \times M$-order "0-1" matrix *P* to store the mapping relationship between tasks and edge servers, and set $P_{ij} = 0$ to indicate that the T_i is not processed on the H_j, and $P_{ij} = 1$ to indicate that the T_i is processed on the H_j. It is the *P* matrix that needs to make the decision, i.e., the scheduling of the tasks with the edge nodes.

1) Objective Function

The objective of this resource allocation scheduling method is to calculate the resource utilization and quality of service Qos.
(1) Computational resource utilization

For the *j*th edge server, if $\sum\limits_{i=1}^{n} P_{ij} = 0$, it means that there is no task to be processed in the *j*th edge server, and if $\sum\limits_{i=1}^{n} P_{ij} > 0$, it means that there is at least one task to be processed in the *j*th edge server. Then, the utilization of the computational resources in the *j*th server which has been turned on is as in Eq. (1),

$$U_j = \frac{\sum\limits_{i=1}^{n} (P_{ij} * R_{ic})}{C_j}. \tag{1}$$

In order to save resources, all the edge servers are not in the state of opening all the time, so the step function sgn is introduced, and the number of servers opened is set as Eq. (2),

$$\sum_{j=1}^{m} sgn\left(\sum_{i=1}^{n} P_{ij}\right). \tag{2}$$

It can be obtained that the total average computational resource utilization of the edge server is as in Eq. (3),

$$U = \frac{\sum_{j=1}^{m}\left\{\sum_{i=1}^{n}\left(P_{ij} * R_{ic}\right)/C_j\right\}}{\sum_{j=1}^{m} sgn\left(\sum_{i=1}^{n} P_{ij}\right)}. \tag{3}$$

(2) Quality of Service Qos

Wishing to maximize the quality of service Qos, the problem can be transformed into minimizing the average value of delay and packet loss between the tasks to the edge servers, i.e., minimizing the $(D + \alpha L)$, where α is the proportionate weight of the delay and packet loss. It can be concluded that the individual quality of service of the *i*th task when it is delivered to the *j*th edge server is as in Eq. (4),

$$Q_{osi} = \frac{1}{\sum_{j=1}^{m} P_{ij}\left(D_{ij} + \alpha L_{ij}\right)}. \tag{4}$$

The total average quality of service Qos can be obtained by summing the individual quality of service and dividing it by the number of tasks, as in Eq. (5),

$$Qos = \sum_{i=1}^{n}\left\{1/\sum_{j=1}^{m} P_{ij}\left(D_{ij} + \alpha L_{ij}\right)\right\}\Big/ n. \tag{5}$$

In summary, it can be obtained that the total optimization objective is to maximize the quality of service and the utilization of computing resources, i.e., the objective function is to maximize Eq. (6),

$$F = \lambda * Qos + (1 - \lambda) * U, \tag{6}$$

where λ is the proportional weighting of quality of service and computing resource utilization.

2) Constraints

According to this resource allocation scheduling method and the above assumptions, combined with the actual scheduling may occur, the following constraints are made.

(1) Each task cannot be split, and can only be processed on one edge server, which can be converted into a mathematical expression, such as Eq. (7),

$$\sum_{j=1}^{m} P_{ij} = 1. \tag{7}$$

(2) For the *i*th task and the *j*th edge server, the computational resources required to complete the task cannot exceed the computational resources owned by the edge server itself, which can be converted into a mathematical expression, such as Eq. (8),

$$\sum_{i=1}^{n} P_{ij} * R_{ic} \leq C_j. \tag{8}$$

(3) For the *i*th task and the *j*th edge server, the bandwidth required to complete the task cannot exceed the bandwidth of the edge server itself, which can be converted into a mathematical expression, such as Eq. (9),

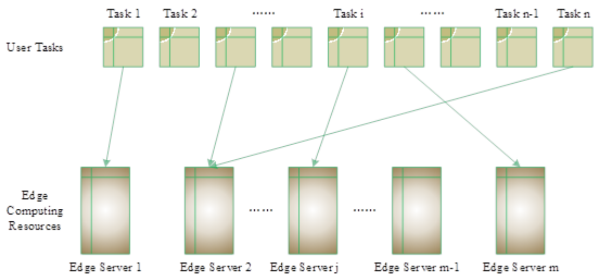


Figure 1: Resource configuration scheduling

$$\sum_{i=1}^{n} P_{ij} * R_{ib} \leq B_j. \tag{9}$$

(4) The values of *i* and *j* must be between *n* and *m*, as in Eqs. (10) and (11),

$$1 \leq i \leq n, \tag{10}$$

$$1 \leq j \leq m. \tag{11}$$

In summary, the constraints of this resource allocation scheduling method can be organized as Eq. (12),

$$\begin{cases} \sum_{j=1}^{m} P_{ij} = 1 \\ \sum_{i=1}^{n} P_{ij} * R_{ic} \leq C_j \\ \sum_{i=1}^{n} P_{ij} * R_{ib} \leq B_j \\ 1 \leq i \leq n \\ 1 \leq j \leq m \end{cases} \tag{12}$$

This paper mainly focuses on the scheduling of tasks based on the computational resources between the task requests and the edge servers, so that the whole IoT system can obtain the optimal performance. In order to simplify and clarify the scheduling problem, we set up the IoT system to have a number of edge servers and a number of randomly generated independent tasks, and the tasks are assigned to the edge servers to be processed. The specific scheduling schematic is shown in Figure 1. The attributes of the edge servers and the tasks are set up, and the resources that need to be consumed by each task are the computational resources and bandwidths, and the edge servers have different computational resources and bandwidths.

B. Resource Allocation Scheduling Optimization for Internet of Things Based on Cloud Platforms

$C^{i}_{monopolize}$ is the shortest completion time that job i can achieve assuming that it can monopolize all the available resources in the service area, then $C^{i}_{monopolize}$ can be regarded as the constant associated with job i given the configuration information of the service nodes in the computing service area of the cloud platform as well as the computational demand information of the submitted job, and, in order to find out the exact value of this constant, the computation is shown in Eq. (13) as follows,

$$\begin{aligned}
&C^i_{monopolize}\left(=\min_{\left\{x^i_{j,p,l}\right\}}\max_j F^i_j\right)\\
&\geq \min_{\left\{x^i_{j,p,l}\right\}}\max_j\Big[\textstyle\sum_{v'=1}^{r}\sum_{l'=1}^{N_r} x^i_{j,v',l'}t^i_{j,v'}+\\
&\max_k\left(F^i_k+\textstyle\sum_{v=1}^{r}\sum_{v'=1}^{r}\sum_{l=1}^{N_r}\sum_{l'=1}^{N_r} e^i_{k,j}d_{v,v'}x^i_{k,v,l}x^i_{j,v',l'}\right)\Big]\\
&x^i_{j,p,l}\in\{0,1\},\\
&\forall j=1,\ldots,N^i_{tasks}\ ,p=1,\ldots,r,k\in VP(j).
\end{aligned}\tag{13}$$

It should be noted that the equality in Eq. (13) holds only if there is no additional queue waiting time between any tasks with execution dependencies in job i, in addition to the necessary communication time overhead.

Let

$$C^i_{execution}=\min_{\left\{x^i_{j,p,l}\right\}}\max_j\left(\max_k F^i_k+\sum_{v'=1}^{r}\sum_{l'=1}^{N_r}x^i_{j,v',l'}t^i_{j,v'}\right),\tag{14}$$

where $C^i_{execution}$ denotes the pure task execution time overhead of job i in the computing service region of the cloud platform. Then from Eq. (14), $C^i_{monopolize}\geq C^i_{execution}$ is always established.

Therefore, the optimization objective of Eq. (14) can be relaxed by removing the communication time overhead between any two tasks with execution-dependent constraints and rewriting the corresponding constraints as in Eq. (15).

$$\begin{aligned}
C^{i(\,relaxed\,)}_{monopolize} &= \min_{\left\{x^i_{j,p,l}\right\}}\max_j\left(\max_k F^i_k+\sum_{v'=1}^{r}\sum_{l'=1}^{N_r}x^i_{j,v',l'}t^i_{j,v'}\right)\\
x^i_{j,p,l}\in\{0,1\},\forall p &= 1,\ldots,r.
\end{aligned}\tag{15}$$

It should be noted that the relaxed problem can be regarded as a classical problem in the field of resource allocation scheduling theory, where *n* tasks are scheduled on *m* machines with different processing speeds, and if the completion time of task *j* is denoted as C_j, the scheduling objective is the problem of "minimizing the completion time of the longest time-consuming task", i.e., the minimization of $C_{\max}=\max_j C_j$, which is also known as the problem of minimizing the job's span, where the term "span" refers to the time cost from the start of the execution of the job's first task to the end of the execution of the last task.

In the same cloud platform computing service area, the $C^i_{monopolize}$ corresponding to each job i is not correlated. Therefore, an approximate solution can be obtained by replacing the parameter $C^i_{monopolize}$ with $C^{i(relaxed)}_{monopolize}$. Not only this, but considering that the original problem is a nonconvex problem, here, we can further consider the problem after its relaxation by binarization. Here, by using the c^i to replace the $C^{i(relaxed)}_{monopolize}$ of the original problem, the problem can be transformed into the form shown in Eq. (16),

$$\min_{\left\{x^i_{j,p,l}\right\}}\sum_{i=1}^{N^r_{jobs}}\left(\max_j\left\{\max_k F^i_k+\sum_{v'=1}^{r}\sum_{l'=1}^{N_r}x^i_{j',v',l'}t^i_{j,v'}\right.\right.\tag{16}$$

Consider M computing jobs $J_1,J_2,\ldots,J_M$ arrive at the cloud computing service region at the same time at time 0. Here, it is assumed that all the relevant information about the jobs (including the computation amount of each task within the job, the placement dependency constraints of the job on the computing service region, and the execution dependency constraints among tasks within the job, etc.) are known. As mentioned earlier, although the offline version of Hoare is not feasible in practice, and its optimization results are very idealized, the results of the offline version can be used as a benchmark for the online version to measure the performance of the algorithm. specifically, first, Horae obtains the shortest completion time of each job under the placement constraints by solving for each job J_i under the current computing service area of the cloud platform, and then solves for the set of relaxed real solutions. $\left\{x^i_{j,p,l}\right\}^{relaxed}$, behind such a solution set, the corresponding practical meaning, as mentioned before, can be regarded as the mathematical expectation of the task assigned to each processor, so the next step is to obtain the processor with the highest expectation by aggregating the real numbers of task decisions on different processors for each task. finally, for each processor, Horae will decide the order of execution of the task on the processor by the original attempted time of completion, and the order of execution of the task on the processor will be determined by the original attempted time. Finally, for each processor, Horae will determine the order of execution of the task on the processor by its original attempted completion time, i.e., the ordering in the processor's virtual queue.

In practice, in jobs with inter-task execution dependency constraints, the size of the inter-task relationship $|\mathcal{E}|$ is usually equal to the size of the tasks contained in it $|\mathcal{V}|$. Therefore, the time complexity of the offline scheduling version of Horae's algorithm is $\mathcal{O}\left(M\left|_r\right|\left|\mathcal{V}\right|\log\left|\mathcal{V}\right|\right)$, excluding the process of calculating the real number of solutions for each job, and the time for calculating the $\left\{x^i_{j,p,l}\right\}^{relaxed}$ is determined by the choice of the convex programming solver, which has an iterative number of iterations of $\mathcal{O}\left(\sqrt{\left|\mathcal{V}_r\right|}\log\left(\frac{\left|\mathcal{V}_r\right|\mu}{\in}\right)\right)$, where the parameters $\in$ and μ are the error ranges that need to be attained in order to converge to the optimal solution, and the corresponding barrier parameter of the original barrier algorithm.

C. Optimization Process of Resource Allocation Scheduling for Internet of Things Based on Cloud Platforms

In this paper, the specific flow of the resource allocation scheduling optimization algorithm based on the Internet of Things is shown in Figure 2. In the process of the algorithm, the termination conditions need to be set to determine whether the search is finished. The termination conditions include the maximum iteration number when the iteration number reaches the maximum iteration number, or when the fitness length is unchanged, and the process is terminated, and the global opti-

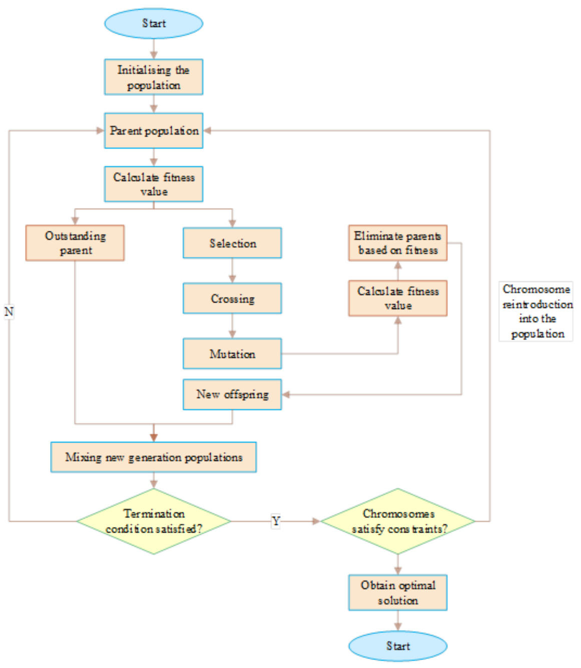

Start
Initialising the population
Parent population
Calculate fitness value
Outstanding parent
Selection
Eliminate parents based on fitness
Crossing
Calculate fitness value
Mutation
New offspring
Chromosome reintroduction into the population
N
Mixing new generation populations
Termination condition satisfied?
Y
Chromosomes satisfy constraints?
Obtain optimal solution
Start

Figure 2: Resource allocation scheduling optimization algorithm

mal solution is obtained. But in the actual resource scheduling process, when the resource allocation scheduling meets the termination conditions, the scheduling method does not satisfy the constraints in the IoT system, such as all the computational resources of the edge servers or the bandwidth is less than the computational resources and bandwidth required to process the task. If there is such a phenomenon, an operation is added to the algorithm without terminating the algorithmic process and the operation is re-run to continue searching until a truly optimal solution is obtained.

III. Results and Discussion

A. Simulation of Resource Allocation Scheduling Performance

1) Simulation of Resource Allocation Scheduling Gains

In this paper, the algorithm is validated using the simulation environment of CloudSim simulation cloud computing platform. Considering the dynamic nature of cloud computing environment, mass data, isomerization, and large task scale, the resource scheduling of cloud computing distributed system is NP-hard problem. The experimental parameters of the algorithm are set as shown in Table 1.The link capacity is set to [0.2,0.4] Mbits, the computational capacity is [4,6] GHz, and the embedding cost is [200,400]. Meanwhile, the exploring factor, the learning rate, and the attenuation factor are set to 0.2, 0.2, and 0.8.

By controlling the number of resource allocation scheduling demands raised by each terminal device, the performance of the algorithm is evaluated in four cases: 8 demands, 16 demands, 32 demands, and 64 demands. The relationship between the total number of VoCS in IoT and the number of iterations of the resource allocation scheduling algorithm in this paper is shown in Figure 3. It can be found that the total number of VoCS increases with the increase of the number of algorithmic iterations, in the case of 8 resource allocation scheduling demands, the total gain of the algorithm is 243.63 when the number of iterations reaches 60 and when the number of resource allocation scheduling demands increases to 64, the total gain of the algorithm increases with it, and the total VoCS number of the algorithm increases to 1397.71 when the number of iterations reaches 60 it shows that with the increase in the number of resource allocation scheduling demands submitted by the terminal devices, the total VoCS number of the end-device and the This indicates that as the number of resource allocation scheduling demands submitted by end devices increases, the possible supply relationship between end devices and edge computing nodes also increases, and the dimension of the solution space to be searched for is also larger. The improved resource allocation algorithm in this paper expands the search scope through the co-proposal of all brokers, and searches for the optimal solution as far as possible.

Parameter name	Parameter symbol	Numerical value
Link capacity	$L_{n,j}(t)$	[0.2,0.4]Mbits
Calculated capacity	$C_{n,\max}$	[4,6]GHz
Computational complexity	$\varphi^{f,k}$	900,1000,1100CPU Periodic number/bits
Arrival data	${A_n}^f(t)$	[0.5,0.7]Mbits
Embedded cost	$e^k(t)$	[200,400]
Minimum throughput	${r_f}^{\min}$	0.2Mbits
Maximum throughput	${r_f}^{\max}$	0.4Mbits
Weighting	$V\beta\lambda\omega$	$10^8 10, 10^4 10^6$
Exploration factor	ε	0.2
Learning rate	ψ	0.2
Attenuation factor	γ	0.8

Table 1: Algorithm parameter setting

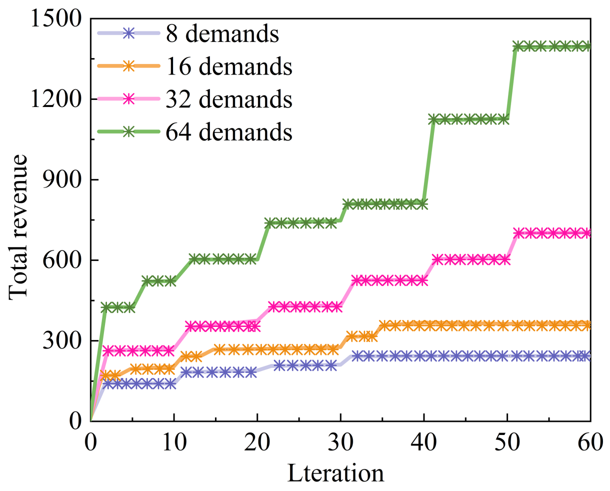


Figure 3: Iteration times and total income

2) Temperature Warning Simulation

In resource allocation scheduling, the working state of CPU is directly related to the normal operation of the whole system, and the temperature can reflect the working condition of the equipment. In resource scheduling under the cloud platform,

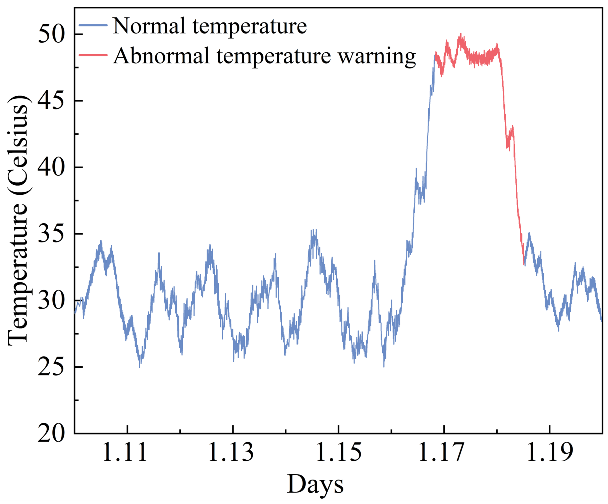


Figure 4: Example of emergency temperature data

the real-time change of the ambient temperature is taken into account in the fault diagnosis of the equipment through the temperature monitoring. Because the ambient temperature has a close relationship with the time, if the temperature threshold of the faulty equipment is set up manually in advance, the threshold value will be adjusted to ensure that the system can accurately warn the equipment failure, and the temperature threshold set up manually also has a greater uncertainty. Adjust the threshold to ensure that the system can accurately warn of equipment failure, and the temperature threshold set manually also has a large uncertainty. The resource allocation scheduling algorithm in this paper introduces an emergency data scheduling scheme, which does not need to set the data priority threshold in advance, and based on the regular data rate of change model real-time judgment of the data priority, gives priority to the allocation of resources to the emergency data and sends the data in time to inform the technicians to carry out inspection and maintenance. The temperature change of the equipment from January 10 to January 20, 2023 is shown in Figure 4, and it can be seen that the temperature data rate of change is abnormal on the 8th day and the warning is given, and the equipment is back to normal on the 9th, 5th day after timely maintenance.

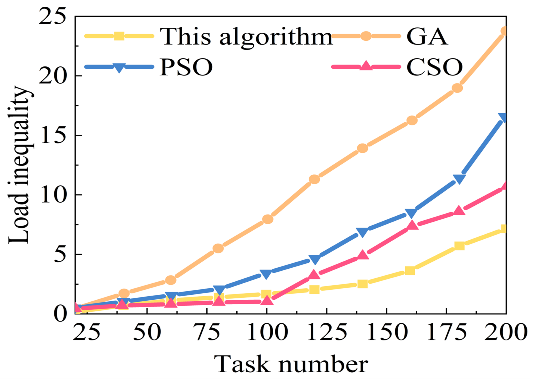
B. Comparison of Resource Allocation Scheduling Effectiveness

In order to verify the effectiveness of the resource allocation scheduling optimization method established in this paper, the difference between the optimization algorithm and the comparison algorithm in various dimensions is compared by changing the number of iterations and the number of tasks. The simulation experiments take into account the processing speed and the length of the tasks to be processed in resource allocation scheduling, and two task sets T1 and T2 are set up to set the number of tasks from 20 to 200 and from 1,000 to 6,000, respectively, representing two task scale cases: small-scale and large-scale. The number of computing nodes is 10, randomly set the performance of virtual nodes, set the capacity of virtual nodes as [1500,2500] MIP, memory as [1024,4096] MB, bandwidth as [6000,12000] b/s, randomly set the length of the task between [600,1000], the maximum number of iterations is 300 times. In order to exclude chance, each method is repeated 10 times independently, and the average value is taken as the final experimental result.

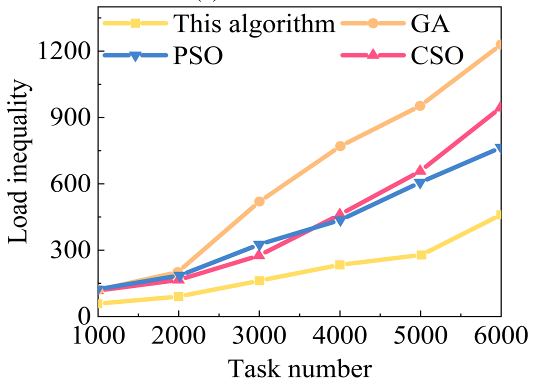
1) Compare Load Imbalance Compare Load Imbalance

This section compares the load balancing degree of the algorithms in two task sets, large-scale (T2) and small-scale (T1), respectively, under the condition of the same number of iterations, and the load imbalance degree of the four algorithms under the two task sets is shown in Figure 5 (a) and (b) represent the results of the processing of the task set T1 and the task set T2, respectively, and the load imbalance degree of the resource allocation scheduling strategy of the four algorithms is rising with the increase of the number of tasks. With the increase of the number of tasks, the load imbalance degree of the four algorithms' resource allocation scheduling policy is increasing, but the load imbalance degree of this paper's algorithm and CSO algorithm's resource allocation scheduling policy increases at a rate significantly smaller than that of the PSO scheduling policy and GA scheduling policy, and the load imbalance degree of this paper's resource allocation optimization algorithm is consistently lower than that of the three algorithms in the whole process.

From the perspective of small-scale task set processing, when the task volume reaches 200, the load imbalance degree of this paper's algorithm is only 7.15, while in the same case, the load imbalance degree of the GA algorithm has reached 23.77, and the load imbalance degree of the PSO algorithm and the CSO algorithm, although a little smaller, also reaches 16.58 and 11.12. In large-scale resource allocation scheduling, the load imbalance degree of this paper's algorithm reaches 6,000 when the task volume is 6,000, while the load imbalance degree of this paper's algorithm is only 7.15. In the large-scale resource allocation scheduling, the load imbalance degree of this algorithm can still be maintained at about 460 when the task volume is 6000, which is mainly because this algorithm improves the diversity of the population, avoids the local optimization, and can arrive at the global optimal solution in a more stable way, while the load imbalance degree of the other three algorithms has already reached more than 700, but we can also note that, unlike the results of the small-sized task setups, the load imbalance degree of PSO algorithm can reach 16.58 and 11.12 when the task volume is 6000 in large-scale resource allocation scheduling. Task processing the load imbalance degree is slightly lower than CSO algorithm. In this paper, the reduction of load balance is increased, which is due to the ability and cost of the virtual machine to balance the probability of a single service to choose the probability of the virtual machine, avoiding the fact that it is too much to be selective in the pursuit of shorter completion time or lower cost, and to be selective and low cost, which leads to the performance of the virtual machine and the low cost of the



(a) Task set T1



(b) Task set T2

Figure 5: Load inequality analysis results

Task set T1	Completion time			
	This algorithm	CSO	PSO	GA
20	0.28	2.36	2.11	11.07
40	1.52	2.38	2.31	12.24
60	3.28	10.52	14.47	22.93
80	4.35	12.91	23.87	27.86
100	13.5	23.59	23.97	33.01
120	13.62	24.02	33.24	36.39
140	24.39	35.04	42.6	40.21
160	28.41	38.04	43.02	46.62
180	37.65	40.5	43.55	51.05
200	39.28	46.2	45.97	56.46

Table 2: Task completion time analysis results(T1)(ms)

Task set T2	Completion time			
	This algorithm	CSO	PSO	GA
1000	702.15	334.17	1160.47	829.2
1500	726.31	518.02	1871.17	1108.07
2000	757.16	570.71	1925.62	1334.04
2500	906.02	644.21	2173.17	1662.63
3000	994.84	659.26	2188.69	1696.15
3500	1220.3	889.39	2853.58	2675.24
4000	1341.66	1507.95	2885.79	3055.25
4500	1606.3	1527.27	2979.4	3173.14
5000	1613.22	1528.82	3506.41	4016.64
5500	1777.16	2018.71	3693.93	4604.88
6000	1966.23	2362.32	3836.64	4682.49

Table 3: Task completion time analysis results(T2)(ms)

virtual machine.

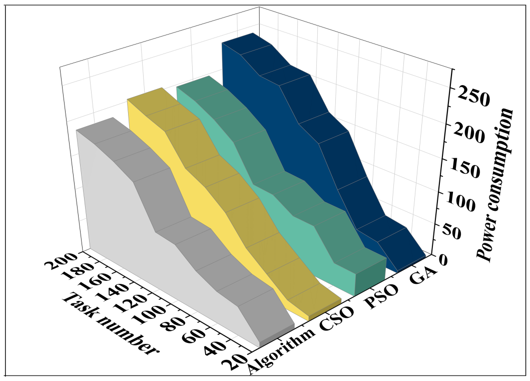
2) Comparison of Task Processing Power

(1) Analysis of task completion time.

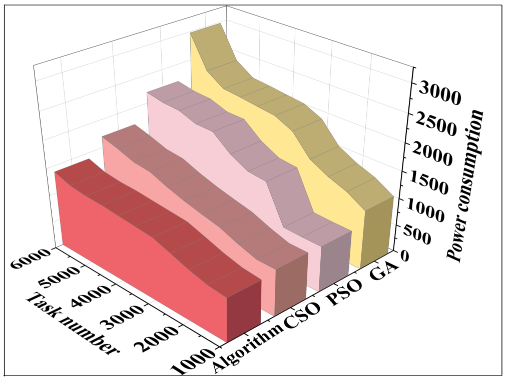
The results of this algorithm, CSO algorithm, GA algorithm and PSO algorithm in the number of small-scale tasks are shown in Table 2, and in the number of large-scale tasks are shown in Table 3. When the number of tasks is small, the difference between the three algorithms' task completion time is small, and in the number of 200 tasks, the processing time of this algorithm is 39.28ms, which is 6.92ms, 6.69ms and 17.18ms shorter than the processing time of CSO, PSO and GA algorithms, respectively. However, as the number of tasks increases, the algorithm in this paper has an obvious advantage, and the task completion time of resource allocation scheduling is significantly better than the remaining three algorithms, when the number of tasks is 6000, the task completion times of the algorithm in this paper, the CSO algorithm, the PSO algorithm and the GA algorithm are 1966.23ms, 2362.32ms, 2362.32ms, 2362.32ms, 2362.32ms, 2362.32ms, 2362.32ms, 2362.32ms, 2362.32ms, 2362.32ms, 2362.32ms and 2362.32ms, respectively. When the number of tasks is 6000, the task completion times of this algorithm, CSO algorithm, PSO algorithm and GA algorithm are 1966.23ms, 2362.32ms, 3836.64ms and 4682.49ms respectively, which indicates that the improved resource allocation scheduling algorithm of this paper improves in the reduction of task completion time.

(2) Analysis of power consumption for task completion

The comparison results of task completion power consumption of four resource allocation scheduling algorithms under two task sets are shown in Figure 6 (a) and (b) are the results of processing energy consumption for task set T1 and task set T2, respectively. For example, when the number of tasks is 140, the task completion power consumption of the three algorithms are 148.33, 138.70 and 149.21 respectively, which are significantly higher than that of the GA algorithm (207.43). Similar to the results of the task processing time analysis, as the number of tasks increases, the superiority of the algorithm of this paper is manifested in large-scale tasks, and when the number of tasks is 5,500, the power consumption of the CSO algorithm (1,611.69), PSO algorithm (2,131.72) and PSO algorithm (2,131.72) are significantly higher than that of the GA algorithm. When the number of tasks is 5500, CSO algorithm (1611.69), PSO algorithm (2131.72) and GA algorithm (2414.29) consume significantly more power than this paper's algorithm (1290.29). through this paper's IoT-based resource allocation and scheduling optimization model under the cloud platform can reduce the power consumption of the task completion.

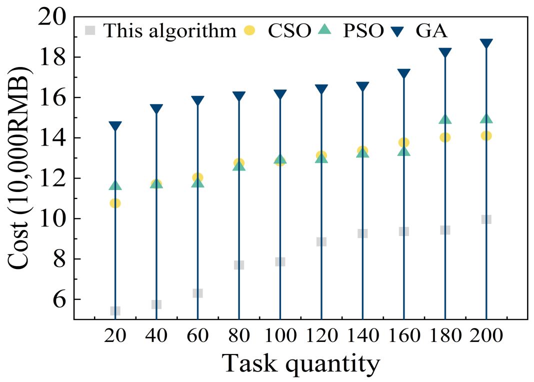


(a) T1 processing power consumption

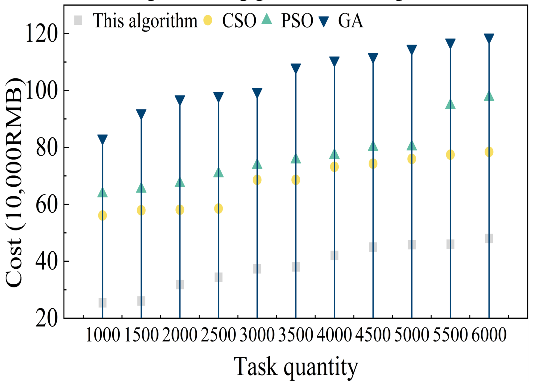


(b) T2 processing power consumption

Figure 6: Task processing energy consumption comparison analysis



(a) T1 processing power consumption



(b) T2 processing power consumption

Figure 7: Different algorithm cost analysis

3) Comparison of Resource Allocation Dispatch Costs

In the actual cloud platform, the user leases the cloud resources according to the demand, one of the leasing methods is "pay by the hour", each virtual machine has the attribute of the cost per unit of time, therefore, according to the execution time of each user's task in the virtual machine and the cost per unit of time can be calculated the total cost. The purpose of this experiment is to verify whether the algorithm for IoT configuration scheduling under the cloud platform spends less cost. Algorithm for IoT configuration scheduling under cloud platform is reduced or not. The experimental results of the cost required by different algorithms are shown in Figure 7 (a) and (b) are the results of the cost analysis under the task set T1 and T2, respectively. As the number of tasks increases, the cost consumed also increases.When the number of tasks is the same, the algorithm of this paper spends the least amount of cost, wherein,when the number of tasks is 6000, the cost is less than that of the GA algorithm. Compared with the GA algorithm, the cost of this algorithm is reduced by 59.54%, and compared with the PSO algorithm, the cost is reduced by 50.82%. The superiority of the IACO algorithm in resource scheduling is mainly due to the inclusion of task constraints in the algorithm, which includes the price of the virtual machine in the scheduling factors. The higher the cost, the less pheromone is left on the VM, and the lower the probability of choosing the VM, and vice versa.

IV. Conclusion

The proposed IoT-based resource allocation scheduling optimization method on cloud platform proves its effectiveness and efficiency in experiments. Through the tests in CloudSim simulation environment, we find that the total number of VoCS increases significantly as the number of task demands increases, which indicates that the algorithm can effectively cope with different scales of scheduling demands. Especially when dealing with large-scale tasks, the algorithm exhibits lower load imbalance and shorter task completion time. For example, when dealing with 6000 tasks, the load imbalance is maintained at around 460, and the task completion time is 1966.23ms. For example, when processing 6000 tasks, the load imbalance is maintained at around 460, and the task completion time is 1966.23ms. In addition, the algorithm also performs well in task processing, and outperforms the comparative CSO, PSO and GA algorithms in terms of task completion time and power consumption, especially in the case of large-scale task processing. Especially in large-scale task processing, the completion time and power consumption of this paper's method are significantly lower than other algorithms.

In terms of resource allocation scheduling cost, this method also shows obvious advantages, for example, when processing

6000 tasks, the cost is reduced by 59.54% compared with GA algorithm, and 50.82% compared with PSO algorithm, which is attributed to the introduction of task constraint function and time cost function in the algorithm, which can effectively balance the cost-effectiveness of resource allocation.

The scheduling method proposed in this study not only improves the resource utilization and service quality, but also achieves significant advantages in task processing time and cost. This result has important theoretical and practical significance for resource allocation and scheduling in the field of cloud computing and IoT, and provides valuable references and inspirations for the future related researches.

In this paper, there are some shortcomings in the improved cloud resource scheduling algorithm, which can not be tested in the real cloud, but only through the current mature, more comprehensive cloud simulation platform. Real cloud data centers are very large, and cloudsim can only simulate smaller cloud data centers, and the real cloud of cloud data is very large, so it is not possible to ensure that the proposed algorithm is suitable for the real cloud. In future research, we can continue to improve the algorithm of this article and extend it to the real cloud, so as to make a more complete assessment of it. At the same time, this paper defines the task of independent, independent, and non-dependent, and will further explore the task of implementing the order of the order, and should take the scheduling plan to achieve the high efficiency of the cloud resource allocation.

Funding

This research was supported by the 2021 Fujian Provincial Department of Education Young and Middle-aged Teachers' Educational Research (Science and Technology) Project: Design of Intelligent Agricultural Monitoring System Based on Cloud Platform (JAT210904).

References

[1] Song, Q. (2022). Design and Application of Land Resource Management System Based on Internet of Things. Wireless Communications and Mobile Computing, 2022, Article ID 2726673, 14 pages.

[2] Hu, C. C. (2020). Profit-based algorithm of joint real-time task scheduling and resource allocation in C-RANs. IEEE Internet of Things Journal, 8(2), 941-950.

[3] Yadav, N., Alashi, M., & Choi, K. K. (2020). Design and development of BTI model and 3d InGaAs HEMT-based SRAM for reliable and secure internet of things application. Electronics, 9(3), 469.

[4] Jamil, B., Ijaz, H., Shojafar, M., Munir, K., & Buyya, R. (2022). Resource allocation and task scheduling in fog computing and internet of everything environments: A taxonomy, review, and future directions. ACM Computing Surveys (CSUR), 54(11s), 1-38.

[5] Kalita, A., & Khatua, M. (2021). Autonomous allocation and scheduling of minimal cell in 6TiSCH network. IEEE Internet of Things Journal, 8(15), 12242-12250.

[6] Alqahtani, A. S. (2020). Lossless data transmission for Internet of things application over wireless multimedia sensor networks using enhanced and optimal path scheduling approach to maximizing the quality of service. Computational Intelligence, 36(4), 1672-1685.

[7] Lv, X., & Li, M. (2021). Application and research of the intelligent management system based on internet of things technology in the era of big data. Mobile Information Systems, 2021, 1-6.

[8] Lee, H. S., & Lee, J. W. (2018). Resource and task scheduling for SWIPT IoT systems with renewable energy sources. IEEE Internet of Things Journal, 6(2), 2729-2748.

[9] Junaid, M., Sohail, A., Turjman, F. A., & Ali, R. (2021). Agile Support Vector Machine for Energy-efficient Resource Allocation in IoT-oriented Cloud using PSO. ACM Transactions on Internet Technology (TOIT), 22(1), 1-35.

[10] Alameddine, H. A., Sharafeddine, S., Sebbah, S., Ayoubi, S., & Assi, C. (2019). Dynamic task offloading and scheduling for low-latency IoT services in multi-access edge computing. IEEE Journal on Selected Areas in Communications, 37(3), 668-682.

[11] Gu, Y., Chang, Z., Pan, M., Song, L., & Han, Z. (2018). Joint radio and computational resource allocation in IoT fog computing. IEEE Transactions on Vehicular Technology, 67(8), 7475-7484.

[12] Liu, H. (2022). An UAV-assisted edge computing resource allocation strategy for 5G communication in IoT environment. Journal of Robotics, 2022, Article ID 9397783, 9 pages.

[13] Hussain, F., Hussain, R., Anpalagan, A., & Benslimane, A. (2020). A new block-based reinforcement learning approach for distributed resource allocation in clustered IoT networks. IEEE Transactions on Vehicular Technology, 69(3), 2891-2904.

[14] Wu, C. G., Li, W., Wang, L., & Zomaya, A. Y. (2021). An evolutionary fuzzy scheduler for multi-objective resource allocation in fog computing. Future Generation Computer Systems, 117, 498-509.

[15] Dang Quoc, H., Nguyen The, L., Nguyen Doan, C., & Xiong, N. (2020). Effective evolutionary algorithm for solving the real-resource-constrained scheduling problem. Journal of Advanced Transportation, 2020, 1-11.

[16] Ghomi, E. J., Rahmani, A. M., & Qader, N. N. (2019). Cloud manufacturing: challenges, recent advances, open research issues, and future trends. The International Journal of Advanced Manufacturing Technology, 102, 3613-3639.

[17] Nouri, N., Abouei, J., Sepasian, A. R., Jaseemuddin, M., Anpalagan, A., & Plataniotis, K. N. (2021). Three-dimensional multi-UAV placement and resource allocation for energy-efficient IoT communication. IEEE Internet of Things Journal, 9(3), 2134-2152.

[18] Seid, A. M., Boateng, G. O., Anokye, S., Kwantwi, T., Sun, G., & Liu, G. (2021). Collaborative computation offloading and resource allocation in multi-UAV-assisted IoT networks: A deep reinforcement learning approach. IEEE Internet of Things Journal, 8(15), 12203-12218.

[19] Raveendran, N., Zhang, H., Song, L., Wang, L. C., Hong, C. S., & Han, Z. (2020). Pricing and resource allocation optimization for IoT fog computing and NFV: An EPEC and matching based perspective. IEEE Transactions on Mobile Computing, 21(4), 1349-1361.

[20] Abedin, S. F., Alam, M. G. R., Kazmi, S. A., Tran, N. H., Niyato, D., & Hong, C. S. (2018). Resource allocation for ultra-reliable and enhanced mobile broadband IoT applications in fog network. IEEE Transactions on Communications, 67(1), 489-502.