Decision Scheduling for Cloud Computing Tasks Relying on Solving Large Linear Systems of Equations

Jing He
College of Artificial Intelligence and Big Data, Chongqing Industry Polytechnic College, Chongqing, China

Correspondence should be addressed to Jing He; hejing@cqipc.edu.cn

Received 20 January 2022; Accepted 21 February 2022; Published 19 March 2022

Copyright © 2022 Jing He. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

With the continuous reform and innovation of Internet technology and the continuous development and progress of social economy, Big Data cloud computing technology is more and more widely used in people’s work and life. Many parallel algorithms play a very important role in solving large linear equations in various applications. To this end, this article aims to propose and summarize a cloud computing task scheduling model that relies on the solution of large linear equations. The method of this paper is to study the technology of solving large-scale linear equations and propose an M-QoS-OCCSM scheduling model. The function of the experimental method is to solve the problem of efficiently executing N mutually dependent parallel tasks within limited resources, while fully satisfying users’ expectations of task completion time, bandwidth rate, reliability, and cost. In this paper, the application experiment of large-scale linear equations in task scheduling is used to study task scheduling algorithms. The results show that when the task load is 10 and 20, the convergence speed of the MPQGA algorithm is 32 seconds and 95 seconds faster than that of the BGA algorithm, respectively.

1. Introduction

The rapid development of the Internet has also promoted the development of cloud computing, and people’s research on cloud computing has become more and more in-depth and extensive. Large-scale data mining and distributed processing technology on the Internet have also been paid more and more attention. This paper combines the characteristics of the two and proposes a model combining text mining and task scheduling under cloud computing. Cloud computing is a new business computing model that uses network connections to obtain various applications, data, and IT services. The core of cloud computing is the integrated scheduling and management of cloud environment and task resources submitted by users according to user needs, and users only need to pay on demand. Therefore, for cloud services, how to meet the different needs of different users for quality of service (QoS) is an important issue that needs to be considered when scheduling cloud computing.

The study of resource deployment and task scheduling problems and their solutions in the cloud computing environment is of great significance and value for the theoretical research and application practice of cloud computing. As a business service, cloud computing must not only consider optimizing task scheduling strategies to improve the service capabilities of the system but also consider the service revenue of cloud service providers. There are some controversies in this area, but there is no efficient solution yet. Therefore, studying cloud computing task scheduling strategies has important theoretical value and practical significance for improving the service capabilities of cloud computing systems.

This paper proposes a balanced clone scheduling algorithm that can effectively improve resource utilization and task scheduling efficiency. This paper proposes a model that combines data mining and task scheduling in a cloud environment. Combining the advantages of data mining and distributed computing, this model can calmly deal with data processing of massive amounts of information and provide efficient services for users. This paper comprehensively considers the transmission cost, processing time, processing cost, and transmission time of data in the cloud computing
environment, and proposes a mathematical model for task scheduling optimization. And this paper proposes a mathematical model and particle swarm optimization algorithm design based on the increase of the variable neighborhood. The simulation results show that the proposed optimization model and optimization algorithm can not only optimize the time, but also optimize the cost.

2. Related Work

The road to solving large-scale linear equations has never stopped. In recent years, people have gradually combined it with task scheduling in order to achieve unexpected results. Wang et al. propose an SSLE method for general constrained optimization based on the mixed exact penalty function. He added automatic adjustment rules to the algorithm for the selection of penalty parameters to ensure that the number of updates of penalty parameters is limited. He also extended the Facchinei activity set recognition technology to general constrained optimization and gave the corresponding recognition function. In each iteration, the algorithm only solves two or three simplified linear equations with the same coefficients to obtain the search direction. Without assuming strict complementarity and weaker than strong second-order sufficient conditions, the convergence rate of the algorithm is proved to be super linear [1]. Phuc and Verbitsky obtained the global solution of quasi-linear equations with metric coefficients and solved a large number of model problems [2]. Lv and Wang studied the existence, uniqueness, and asymptotic stability of traveling wave fronts for discrete quasilinear equations with time delays. He first established the existence of traveling wave fronts using hyperon solutions and monotonic iteration techniques. Then he proved that the traveling wave front is unique before the translation. Finally, he used the comparison principle and compression technique to prove that the traveling wave front has a phase shift and is globally asymptotically stable [3]. Mingtong introduced the basic process of automatic modeling of large buildings. By extending the L-system theory, using elevation maps and automatic river recognition, he researched and proposed an automatic road generation method for complex terrain. Experiments show that the results given by his method are consistent with the explanation, he researched and proposed an automatic road generation method for complex terrain. Experiments show that the results given by his method are consistent with the

3. Linear Equations Solution Method and Task Scheduling Method

3.1. Common Large-Scale Linear Equation Solving Technology. There are two methods for solving linear equations in direct method and iterative method [10]. Small and medium-sized equations \( (n < 10000) \) are often solved by the direct method, and the direct method needs to decompose the coefficient matrix \( A \). Under normal circumstances, matrix sparsity cannot be guaranteed. As the degree \( n \) increases, the amount of memory required becomes \( O (n^2) \), and the amount of floating-point operations required becomes \( O (n^3) \). The sparse direct method based on sorting has some control over memory requirements and time consumption, allowing modern single-CPU personal computers to solve 100,000-order matrices. However, the success of the sparse direct method is related to the nature of the matrix itself, and higher-order linear systems are powerless [11]. Therefore, iterative methods are often used for large or very-large equations. In simple terms, the iterative method is to create a modified \( S_k \) for the initial approximate solution vector \( X_0 \) of each step. This makes the next approximate solution vector \( X_{k+1} \) equal to the current approximate solution plus the correction vector \( S_k \).

\[
X_{k+1} = S_k + X_k. \tag{1}
\]

This produces an iterative vector sequence \( \{X_k\} \). It can be seen that the ideal correction vector should be

\[
S_k = X^* - X_k = Q(b - Q \cdot r_k), \tag{2}
\]

where \( X^* \) is the solution of the simultaneous equations, and \( r_k \) represents the \( k \)th residual vector. However, in order to obtain this ideal correction vector, this article needs to solve equation \( Q \cdot r_k = X_k \), which is actually the same problem as solving the original equation. Different iterative methods represent different methods of calculating the correction vector \( S_k \) [12]. The classic iterative method is based on matrix division, assuming that the coefficient matrix \( A \) has the following divisions.

\[
A = P - Q, \tag{3}
\]
where \( P \) is the invertible matrix and the original system of equations is \((P - Q)x = b\). Therefore, the following iteration sequence is constructed.

\[
X = P^{-1}Q_{sk} + P^{-1}b. \tag{4}
\]

In order to avoid calculating \( P^{-1}Q_{sk} \) and \( P^{-1}b \), it can turn to solving the equations in the following way:

\[
P_{sk} = Q_{sk} + b. \tag{5}
\]

Different \( P \) choices have different segmentation methods and different iterative methods [13]. Assumptions:

\[
A = D - C_L - C_K. \tag{6}
\]

Among them, \( D \) is the diagonal part of \( A \), \( C_L \) is the lower triangular part of \( A \), and \( C_K \) is the upper triangular part of \( A \).

In order to achieve better results, people constructed a symmetric over-relaxation method (SSOR method), splitting \( A \) into \( A = P_{ssor} + Q_{ssor} \), where

\[
P_{SSOR} = \frac{w}{2 - w}\left(\frac{1}{w}D - C_L\right)D^{-1}\left(\frac{1}{w}D - C_K\right). \tag{7}
\]

In addition, there are methods such as alternating direction iteration, Hermitian and anti-Hermitian splitting, and multiple splitting, all of which are based on practical problems and are all numerically algebraic [14]. The coefficient matrix of the linear equation system obtained by the finite-element method or the method of moments is generally a complex symmetric matrix, which can be described in the following format.

\[
(A + iB)(z + iy) = b + ic. \tag{8}
\]

Among them, \( A, B \in \mathbb{R} \) is a real symmetric matrix. The Krylov subspace method of common asymmetric matrices (such as GMRES, BICGSTAB, and QMR) can be used to solve complex symmetric linear equations. However, using these methods to solve the problem does not take advantage of the symmetry of the matrix. Some solutions transform the symmetric system into an equivalent real symmetric system and then use the Krylov method to solve the real symmetric matrix [15]. The following are some iterative methods for directly solving (8). Using the concept of quasi inner product \((x, y) = x^\dagger y\), where the \( x, y \in C \) superscript letter \( T \) represents transpose instead of conjugate transpose, the QMR method can recursively reach the minimum in a short time. This type of quantity and special matrix are often encountered in actual engineering [16]. The key to the success of the Krylov subspace method is the choice of prerequisites. The preprocessing is to transform a linear equation system into another linear equation system with the same solution, but the transformed equation system has the characteristic of facilitating iterative solution. The prerequisite is to generate a matrix for this transformation. If \( P \) is an invertible matrix somewhat similar to \( A \), then the preprocessed system looks like the following equation:

\[
P^{-1}A_x = P^{-1}b. \tag{9}
\]

If there is the same solution as equation (1), and there is an equation that is easy to solve, then \( P \) is called a prerequisite. Equation (9) is preprocessed from the left side and can also be preprocessed from the right side.

\[
AP^{-1}y = b, x = P^{-1}y. \tag{10}
\]

Or, preprocess both sides at the same time:

\[
P_{1}^{-1}AP_{2}^{-1}y = P_{1}^{-1}b, x = P_{2}^{-1}y. \tag{11}
\]

Then, the precondition in formula (11) is \( M = M_1M_2 \). A good precondition should have the following two properties: the system after the precondition should be easy to solve, the precondition should be easier to construct and the cost of applying the precondition should not be too high [17].

In modern computer architecture systems, the biggest problem with incomplete decomposition of prerequisites is that it is not easy to parallelize the process of construction and use. Approximate prerequisites for adapting to modern computer structure systems have appeared, and their implementation has natural parallelism [18]. In recent years, the algebraic multigrid (AMG) method that combines the area division (DD) method, multigrid (MG), and physical background to solve partial differential equations (PDEs) has become a hot spot in numerical computing [19].

### 3.2. M-QoS-OCCSM Scheduling Model Based on Cloud Computing

The architecture of the M-QoS-OCCSM collaborative scheduling model proposed in this chapter is shown in Figure 1.

As shown in Figure 1, the execution process of the entire M-QoS-OCCSM scheduling model is as follows. First, the user submits the application task with the deadline of the bottom line and the scheduling budget target constraint to the task scheduler. Then the task scheduler sends the QoS target constraint conditions and other related parameter information of the application task to the extension module of the M-QoS-OCCSM model [20]. Next, the extension module of the M-QoS-OCCSM model first applies the method described in the previous section to model the application task with multiobjective constraints. Then comprehensively consider the application task’s QoS target constraints and user expectations and based on a membership function, apply the method described to convert the application task’s multiple QoS target constraints into a single-objective constraint optimization problem. Finally, the reconstructed genetic algorithm is applied to approximate the optimal solution of the mentioned single-objective optimization problem [21]. The calculated result is the final scheduling decision plan, which is returned to the scheduler. The scheduler selects appropriate computing resources for the application tasks for scheduling according to the optimization results, thereby completing the scheduling decision.

The extension module serves as an integrated unit in the entire M-QoS-OCCSM scheduling model architecture. For some applications, it can be designed as a separate middleware module for real task-scheduling scenarios [22].
From the user’s point of view, the user always hopes to complete all application task scheduling requests within a desired time interval. This means that the actual execution of application tasks should be reduced as much as possible on the assigned computing nodes. At the same time, for a specific resource $R$ available on a computing node, the total computing power consumed by all application tasks on the resource $R$ cannot exceed the computing power that the resource can provide [23].

It can be seen from the analysis that the QoS target constraint of the aforementioned time limit bottom line can be expressed as follows:

$$\text{min} \left( O_{\text{deadline}} \right) = \sum_{t=1}^{T} \sum_{n=1}^{N} M_{tn} V_{tn},$$
$$s.t. \sum_{t=1}^{T} \sum_{n=1}^{N} \sum_{k=1}^{K} M_{tn} R_k \leq \sum_{n=1}^{N} \sum_{k=1}^{K} S(CN_{tn}).$$

On the contrary, users always want to save scheduling costs as much as possible while obtaining satisfactory services. Therefore, the cloud system should complete the scheduling and execution of all tasks as much as possible under the constraints of the established scheduling budget [24]. In other words, in an established cloud computing environment, the system should reduce the total scheduling cost of all tasks as much as possible. In addition, for a specific application task $t$, the sum of the scheduling costs consumed by all available resources on the target computing node cannot exceed the task scheduling budget.

According to the analysis, the QoS target constraint of the application task scheduling budget can be expressed as follows:

$$\text{min} \left( O_{\text{budget}} \right) = \sum_{t=1}^{T} \sum_{n=1}^{N} M_{tn} C_{tn}, s.t. \sum_{n=1}^{N} M_{tn} C_{tn} \leq B_n, \forall n \in T.$$  \hfill (13)

This paper compares the algorithm based on QoS constraints with the basic genetic algorithm. The relationship between tasks with interdependence constraints is shown in Figure 2, and the specific parameters and weight vector settings are shown in Table 1.

The user wants to pay as little scheduling cost as possible for the scheduling request of the application task. In real-world cloud computing application scenarios, the cost of scheduling application tasks is often directly proportional to the user’s satisfaction with completing the scheduling goals. Therefore, users want to spend less on their application task scheduling budget, assuming they have a satisfactory scheduling service [25]. As shown in formula (14), assuming that a satisfactory scheduling service is obtained, the membership function of the task scheduling expenditure can be used to express the membership of the task scheduling budget and the scheduling target.

$$Y_{\text{budget}} = 1 - \frac{C_{\text{min}}}{C_{\text{max}}}. \hfill (14)$$

In the formula, $C_{\text{min}}$ represents the minimum expected value of the scheduling overhead of the application task $t$, and $C_{\text{max}}$ represents the scheduling budget when the application task $t$ is scheduled on the target computing node $C_{\text{max}}$. It can be seen from equation (14) that as the scheduling budget expenditure $C_{\text{max}}$ of the application task $t$ decreases, the membership function value $Y_{\text{budget}}$ of the scheduling budget decreases. Since users want to pay as little task scheduling budget as possible, the goal of the final task scheduling solution is to minimize the value of the mentioned membership function as much as possible [26].

Through the analysis of the two QoS target constraints, deadline and scheduling budget. This article finds that with less scheduling budget and the shortest possible deadline, everything needed to complete an application task scheduling request has increased. It tries to minimize the aforementioned period. The membership function value of time and the membership function value of scheduling budget expenditure. Based on this conclusion, two membership

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure1.png}
\caption{M-QoS-OCCSM system architecture.}
\end{figure}
Functions can be used to transform the task scheduling multi-QoS objective constraint optimization problem into a single-objective constraint optimization problem. The transformed single-objective constrained optimization problem is shown as follows:

$$\text{min} \ (O) = w_1 \text{min} (O_{\text{deadline}}) + w_2 \text{min} (O_{\text{budget}}).$$  \hspace{1cm} (15)$$

In the formula, the deadline and scheduling budget weight parameters $w_1$ and $w_2$ satisfy $\sum_{i=1}^{2} w_i = 1, 0 \leq w_i \leq 1$. When scheduling tasks, pay attention to the deadlines and the scheduling budget constraints of each user. Therefore, when constructing the objectives of the mentioned task scheduling scheme, adjusting the two weight parameters for different users can meet the various QoS target constraints of users.

By default, Hadoop usually adopts a first-come first-serve strategy. The advantages of this strategy are simplicity and low overhead, while reducing the burden of the job tracker [27]. The basic idea of solving the problem is to queue up all tasks submitted by the end customer according to the time when the job was submitted. The execution order of the job queue is usually determined by priority and transmission order. That is, the job submitted first by the system has a higher priority by default and will be processed earlier. But the disadvantage of the FIFO strategy is that it is unfair. According to the solution idea of this strategy, for those jobs with low priority, the chance of being processed will be greatly reduced, and the waiting time for idle machines will be particularly long. That is to say, it is difficult to guarantee the QoS for jobs with low priority. This strategy lacks consideration of differences in job requirements.

### 3.3 Cloud Computing Task Scheduling Strategy

Cloud computing resources include storage resources, computing resources, network resources, and so on. In fact, these resources are abstracted into services through virtualization technology and provided to the outside world. On the contrary, there is a correlation between service quality and resource occupancy rate, service quality, and energy consumption. Therefore, when optimizing resources, this article must consider comprehensively, rather than biasing one aspect. Therefore, when optimizing resource deployment and scheduling, all resources must be fully considered and jointly managed and optimized. It is necessary to optimize the QoS, while optimizing energy consumption and cost [28].

The state of task scheduling in the cloud computing environment can be explained as follows. The total number of resources is $P$, and the corresponding set is $R = \{r_1, r_2 \cdots r_n\}$. The total number of jobs submitted by
users is $M$, corresponding to set $J = \{J_1, J_2, \cdots, J_M\}$. Assuming that the $M$ jobs corresponding to the set $T = \{T_1, T_2, \cdots, T_N\}$ are divided into $N$ tasks, the $J_m$th job is divided into $T_{\text{Num}}(J_m)$ tasks, and then the total number of tasks corresponding to the total number of tasks is as follows:

$$N = T_{\text{total}}(M) = \sum_{m=1}^{M} T_{\text{Num}}(J_m).$$

(16)

Network bandwidth is an important indicator to measure network usage [29]. The size of the bandwidth determines the size of the network transmission capacity, which in turn affects the communication efficiency in the cloud environment. The more frequent the communication, the greater the amount of information, and the higher the bandwidth requirements. Let $B_{\text{run}}$ be the resource bandwidth of the cloud computing environment, $B_{\text{user}}$ represents the expected bandwidth of the job $J_m$ specified by the user, and $B_i$ represents the expected bandwidth of the task $T_i$ divided by the job $J_m$, then

$$B_{\text{user}} = \sum_{m=1}^{T_{\text{Num}}(J_m)} B_i.$$  

(17)

The function of user satisfaction $w$ obtained bandwidth is as follows:

$$W = \left[ \frac{\theta}{T_{\text{Num}}(J_m)} \right] \sum_{m=1}^{T_{\text{Num}}(M)} \frac{B_{\text{run}}}{B_i}.$$  

(18)

Assuming that the resource failure rate in the cloud computing environment is $P$ (which can be obtained through the resource monitoring system) and the user’s expected task completion rate is $P_{\text{suc}}$, the user satisfaction function of the job completion rate is as follows:

$$W_{\text{suc}}(J_m) = \theta \ln \left( \frac{1 - P}{P_{\text{suc}}} \right).$$  

(19)

Cost constraint is one of the most popular QoS constraints, and cost is one of the important components of user QoS. Assuming that resources are billed per unit, $P_i$ is the number of resources, and $C_{\text{cpu}}, C_{\text{mem}}, C_{\text{stor}}$, and $C_{\text{bw}}$ represent the prices of CPU, memory, storage, and bandwidth resources. Then, the total cost of task $T_i$ can be expressed as follows:

$$C_i = P_1 C_{\text{cpu}} + P_2 C_{\text{mem}} + P_3 C_{\text{stor}} + P_4 C_{\text{bw}}.$$  

(20)

Through the previous related research on the existing results of the equations solution, it can be found that each equation solution must have a suitable data center network structure, such as a centralized structure, a distributed structure, and a hierarchical structure [30]. In view of this, before introducing the DRMS equation solution introduced by the task scheduling system architecture in the cloud computing environment, first introduce the data center network structure that implements the equation solution.

As shown in Figure 3, the data center that implements the DRMS equations solution is a hierarchical structure. This structure is mainly composed of three types of nodes: regional host node, regional head node, and ordinary host node (also called site, or Site) [31].

4. Task Scheduling Experiment and Analysis Based on Cloud Computing

4.1. Application of Large-Scale Linear Equations in Task Scheduling

In mathematics, the Gaussian elimination method is also known as the Gauss Jordan elimination method, which is an algorithm for solving linear algebra system problems using linear equations [32]. The algorithm first determines the rank of the matrix, and then calculates the inverse of the invertible square matrix. When designing the Gauss method that can cancel the inverse allocation process, the elimination method also reduces the unknown coefficients on the main diagonal to zero. The elimination method does not need to replace the diagonal coefficient matrix equation, this method is called Jordan elimination method. However, the number of multiplication and division operations in Jordan elimination is twice that of pure Gaussian elimination. This is not desirable when solving linear equations, but it is useful when solving inverse matrices.

The operating speeds of Gauss elimination method, and Jordan elimination method are shown in Tables 2 and 3, respectively.

The experimental results are shown in Table 4 by running the Gauss elimination method on a 4-core
multiprocessor to process the Poisson equation. From the data in the table, the speedup for solving the Poisson equation under four cores can be calculated as shown in Figure 5.

The following conclusions can be drawn from analyzing the graphs in this article:

① When the scale of meshing reaches a certain level, compared with the serial solution of single-core, the parallel solution of four-core will highlight its advantage in solution time. Because, when the scale of grid division is small, the communication time between each process takes a larger proportion than the overall running time. Therefore, the parallelism of multiple cores does not show its advantages in the process of solving a smaller meshing scale.

② Analyzing the speedup ratio of various improved parallel algorithms in Figure 5, it can be seen that the nonblocking method can indeed improve the efficiency of parallel solving. And on this basis, overlapping a part of the calculation with nonblocking message passing can further improve the efficiency of parallel solving.

③ From the analysis of the speedup ratio of the parallel algorithm in Figure 5, it can be seen that if the number of processes increases, the speedup ratio tends to be greater than 1, indicating an increase in efficiency.

| Number of parallel nodes | Dense matrix scale | 1002 | 2002 | 5002 | 6002 | 9002 |
|--------------------------|-------------------|------|------|------|------|------|
| Total time               | 0.053             | 0.0817 | 0.0093 | 0.1104 | 0.0523 |
| Communication time       | 0.0576            | 0.0994 | 0.0935 | 0.099 | 0.0254 |
| 1                        |                   |      |      |      |      |      |
| Parallel computing time  | 0.0446            | 0.0064 | 0.0344 | 0.1106 | 0.0059 |
| Speedup ratio            | —                 |      |      |      |      |      |
| Parallel efficiency      | —                 |      |      |      |      |      |
| Total time               | 0.1128            | 0.0884 | 0.0534 | 0.0554 | 0.0488 |
| Communication time       | 0.0777            | 0.0115 | 0.0955 | 0.1121 | 0.0932 |
| 2                        |                   |      |      |      |      |      |
| Parallel computing time  | 0.0574            | 0.0069 | 0.0928 | 0.1062 | 0.0052 |
| Speedup ratio            | 0.67              | 0.53 | 0.31 | 0.85 | 0.85 |
| Parallel efficiency      | 0.33              | 0.92 | 0.99 | 0.39 | 0.49 |
| Total time               | 0.1069            | 0.0932 | 0.0617 | 0.0501 | 0.0181 |
| Communication time       | 0.0988            | 0.0045 | 0.0873 | 0.0056 | 0.044 |
| 4                        |                   |      |      |      |      |      |
| Parallel computing time  | 0.0101            | 0.0868 | 0.0439 | 0.0328 | 0.0838 |
| Speedup ratio            | 1.12              | 0.9 | 0.88 | 0.98 | 0.68 |
| Parallel efficiency      | 0.4               | 1.19 | 0.51 | 0.3 | 0.64 |

The comparison of parallel efficiency in various situations is shown in Figure 4.
fixed, the parallel efficiency will increase as the size of the subdomain problem increases. As the problem grows, the number of Gauss-Jordan iterations also increases. Moreover, the increase in the number of iterations of the Gauss elimination method has a linear trend with the increase of the grid division scale, which shows the rationality of solving large-scale sparse linear equations by the Gauss elimination method. This paper also proposes a task scheduling optimization strategy based on service quality optimization, taking into account system load conditions. This article also gives the approximate service cost of the system and its related proofs. Through experiments, this article analyzes the model and TSSQO strategy in detail and compares them with representative strategies. It verifies that the strategy in this article can more reasonably schedule tasks, so that the entire system can not only achieve high service quality, but also relatively low service costs.

4.2. Multimode Automata Matching Experiment. Most application services in cloud computing must be decomposed into several subtasks for scheduling, and the decomposed subtasks have different degrees of dependence. How to improve the parallelism, real-time and dynamics of coupling-dependent task scheduling, improve system utilization, and perform reasonable scheduling, and deployment of dependent task requests has become a hot spot to be solved in current distributed computing and cloud computing.

This experiment tests the time performance of the multimode automata matching method. The tested hardware environment is 200G hard disk, 16G memory, and two 4-core CPUs. The computer operating system is Ubuntu Linux. This experiment tested the matching algorithm on a 1G data set file and compared the time consumption of the traditional algorithm and the multimode automata algorithm when the comparison rules are the same.

It compares the scheduling length and communication overhead performance of TDDPS with HEFT and Min_Min algorithms in a distributed system composed of three heterogeneous processor nodes with different task sets. And its statistics use TDDPS algorithm to carry on the system load balance that depends on the task scheduling.

The task graph is randomly generated in the experiment. The number of nodes in each task graph [1~100], task dependencies, the amount of communication data between nodes [1~100], and the execution time of each task on different processors [1~100] are all randomly generated. The experiment compares the three scheduling ideas from the three aspects of scheduling length, communication energy consumption, and node resource utilization. Figure 6 shows the scheduling length and communication overhead of the three on different task sets.

According to the results of Figure 6, it can be seen that the TDDPS algorithm has good performance in terms of scheduling length and communication overhead. Owing to the distributed negotiation and scheduling mechanism, each execution node in the heterogeneous system first evaluates the load of the node, and it makes a negotiation response message to the task request according to the real-time load situation of the node. This ensures a certain degree of load balance and improves resource utilization.

In order to test the performance of the BCSOA algorithm, this paper uses the CloudSim simulation platform to simulate and build a data center. At the same time, the algorithm is compared and tested with the DFGA algorithm and the ASAP algorithm. At present, the ASAP algorithm and the DFGA algorithm are typical algorithms in the field of cloud computing task scheduling research. It has achieved relatively ideal experimental results in the specific application process.

The experimental test contains two aspects: the first type of experiment is mainly compared from the perspective of task completion time, and the second type of experiment is mainly compared from the perspective of the balance factor. Figure 7 shows the comparison of the task completion time.
of the three task scheduling algorithms when the number of tasks in the system is \(n = 1000\).

Experimental data show that the traditional matching algorithm has severe time performance degradation when the number of rules increases, and the matching mechanism using multimode automata can greatly improve the matching speed when the rule set is large. Through the comparison of the results of the four rule sets, the time performance of the matching based on the multimodal automata has been greatly improved compared with the traditional algorithm.

4.3. Task-Scheduling Algorithm. In order to verify the performance of the MPQGA task-scheduling algorithm, this paper compares the algorithm with the previously mentioned heuristic algorithms HEFT-T, HEFT-B, CPDP, and the basic random search algorithm BGA. This article carried out related tests on two test sets of actual application program graph and randomly generated DAG application table.

The MPQGA algorithm is implemented by C programming. The DAG task graph is represented by a class, and its members include a series of tasks. It uses a two-dimensional matrix to represent the computing speed of each task in a heterogeneous processor and uses another two-dimensional matrix to represent the amount of communication data between each pair of tasks. The task in the program is also a class, and its members include the predecessor and successor of the task, the input and output of the task, and the amount of data calculation of the task. To implement an FFT with an input vector of \(k\), there are \(2k - 1\) recursive calls and \(k \log_2 k\) butterfly operations (this article assumes that \(k = 2^i\), \(i\) is an integer). In the FFT task graph, each path from the start of the task to the end of the task is a critical path. As the computational cost of tasks at any level is the same, the communication costs of all paths between the two levels are the same. Figure 8(a) shows the average SLR of the task scheduling algorithm for FFT task graphs of different sizes, in which the MPQGA algorithm is better than the average SLR of other algorithms. Figure 8(b) shows the efficiency of different algorithms with 64 data nodes, and the MPQGA algorithm is also better than other algorithms in efficiency.

In the experiment of this article, the value of CCR is set to 0.2, 0.4, 0.6, 1.0, 5.0, and 8.0. Figure 9 shows the average SLR and the efficiency value of the algorithm with different CCR values and different numbers of heterogeneous processors.

Figure 9(a) shows that when the CCR value increases, the average SLR of the algorithm also increases. Figure 9(b) shows the efficiency of the algorithm in the case of different numbers of processors.

In these experiments, this paper uses randomly generated task graphs to evaluate the performance of the algorithm. In order to generate random graphs, this article implements a random graph generator that allows users to generate random graphs with different characteristics. The input parameters of the generator: the number of tasks in the graph, the number of instructions for each task (calculation), the number of subsequent tasks (parallelism), and the CCR value. This article evaluates the performance of the algorithm.
under different parameters, including different numbers of tasks, different numbers of heterogeneous processors, and different CCR values. The value in Figure 10 is the average maximum and minimum completion time of more than 100 different random DAG graphs.

Figure 10 shows the convergence process of the maximum completion time for a set of randomly generated DAG application graphs (size 10 and 20, respectively). This article can observe that when the DAG application graph is small, the convergence speed of the MPQGA algorithm and the BGA algorithm is very fast. When the DAG application becomes very large, the convergence speed of the algorithm becomes slow, but the convergence speed of the two algorithms is different. The MPQGA algorithm proposed in this paper converges faster than the BGA algorithm, and the final minimum and maximum completion time obtained is better than the BGA algorithm.

In general, with the development of network reliability, cheap computer hardware and software, virtualized hardware technology, and service-oriented architecture, more and more companies have begun to invest in the development and application of cloud computing technology. More and more companies and individuals are beginning to use cloud computing technology. It continues to promote the progress and development of cloud computing technology. Today, cloud computing, as the core of the next generation of IT technology, has great advantages such as high flexibility, scalability, ease of use, economies of scale, green energy saving, and environmental protection. It is considered to be the next-generation network after the Internet.

5. Discussion

Cloud computing has become a trend in the field of IT technology. This article has conducted in-depth research on the task scheduling strategy of cloud computing and has achieved some results. Nevertheless, the work of this article is not yet full and mature enough. At the same time, due to the rapid development of cloud computing technology, there are still many problems in the field of cloud computing.
technology that need to be further improved and resolved. These problems are mainly reflected in the following aspects: user privacy and data security. The development of cloud computing requires the participation of a large number of users. How to ensure that the user information stored in the cloud is sufficiently safe to ensure that user privacy and user data will not be collected or leaked by cloud service providers and will not be stolen or illegally used by third parties. This is not only a technical issue but also a legal issue. It needs the joint promotion of researchers and government departments. Scholars in the future should strengthen the study of compressed storage of sparse matrices. In this article, the matrix storage of the MPI experiment program is a file that stores all zero elements in the sparse matrix, which will greatly waste the system's memory resources and affect the performance of the parallel program.

6. Conclusion

This paper has conducted in-depth analysis and research on the task-dependent scheduling mechanism in cloud computing, including the characteristics and shortcomings of the existing scheduling model, and the problems to be solved by task-dependent scheduling. This paper proposes a new system model suitable for cloud computing environment and a dynamic parallel dependent task scheduling mechanism. In this paper, aiming at the resource-matching problem in the scheduling process, a parallel matching method based on multimode automata is proposed. When the task amount is 10, the calculation speeds of the BGA algorithm and the MPQGA algorithm are 43 and 11, respectively, and when the task amount is 20, the calculation speeds of the BGA algorithm and the MPQGA algorithm are 108 and 13, respectively. It can be seen that the MPQGA algorithm is more excellent, and the more tasks, the more obvious the advantage.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare no conflicts of interest.

Acknowledgments

This research study was sponsored by Chongqing Yubei District Science and Technology Planning Project. The name of the project is Big Data development of service PAAS automatic delivery and management platform. The authors thank the project for supporting this article.

References

[1] Y. Wang, L. Chen, and G. He, "Sequential systems of linear equations method for general constrained optimization without strict complementarity," Journal of Computational & Applied Mathematics, vol. 182, no. 2, pp. 447–471, 2017.
[2] N. C. Phuc and I. E. Verbitsky, "Singular quasilinear and Hessian equations and inequalities," Journal of Functional Analysis, vol. 256, no. 6, pp. 1875–1906, 2017.
[3] G. Lv and M. Wang, "Existence, uniqueness and stability of traveling wave fronts of discrete quasi-linear equations with delay," Discrete and Continuous Dynamical Systems - Series B (DCDS-B), vol. 13, no. 2, pp. 415–433, 2017.
[4] Li Mingtong, "Research on the mechanism and influence factors of urban style building based on cloud computing logistics information," Cluster Computing, vol. 22, no. 6, pp. 13873–13880, 2019.
[5] J. K. Samriya, S. Chandra Patel, M. Khurana, P. K. Tiwari, and O. Cheikhrouhou, "Intelligent SLA-aware VM allocation and energy minimization approach with EPO algorithm for cloud computing environment," Mathematical Problems in Engineering, vol. 2021, no. 6, pp. 1–13, 2021.
[6] K. Jabir, D. P. Suseelan, and P. M. Keriikkattil, "An effective meta-heuristic based multi-objective hybrid optimization method for workflow scheduling in cloud computing environment," Cluster Computing, vol. 24, no. 3, pp. 2367–2384, 2021.
[7] Z. Tang, L. Qi, Z. Cheng, K. Li, S. U. Khan, and K. Li, "An energy-efficient task scheduling algorithm in DVFS-enabled
cloud environment,” *Journal of Grid Computing*, vol. 14, no. 1, pp. 55–74, 2016.

[8] A. M. Gruzlikov, N. V. Kolesov, Y. M. Skorodumov, and M. V. Tolmacheva, “Task scheduling in distributed real-time systems,” *Journal of Computer and Systems Sciences International*, vol. 56, no. 2, pp. 236–244, 2017.

[9] G. Marković, N. Zdravković, M. Karakašić, and M. Kolarević, “Modified PROMETHEE approach for solving multi-criteria location problems with complex criteria functions,” *Technicki vjesnik-Technical Gazette*, vol. 27, no. 1, pp. 12–19, 2020.

[10] V. Arvind, J. Köbler, S. Kuhnert, and J. Torán, “Solving linear equations parameterized by hamming weight,” *Algorithmica*, vol. 75, no. 2, pp. 322–338, 2016.

[11] N. V. Dung and V. T. L. Hang, “The generalized hyperstability of general linear equations in quasi-Banach spaces,” *Journal of Mathematical Analysis and Applications*, vol. 462, no. 1, pp. 131–147, 2018.

[12] A. Bihlo and R. O. Popovych, “Group classification of linear evolution equations,” *Journal of Mathematical Analysis and Applications*, vol. 448, no. 2, pp. 982–1005, 2016.

[13] Y. Liu, C. Lagema, B. D. O. Anderson, and G. Shi, “Exponential least squares solvers for linear equations over networks,” *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 2543–2548, 2017.

[14] W. V. Petryshyn, “On two variants of a method for the solution of linear equations with unbounded operators and their applications,” *Journal of Mathematics & Physics*, vol. 44, no. 1-4, pp. 297–312, 2016.

[15] L. Jin, S. Li, B. Hu, M. Liu, and J. Yu, “Noise-suppressing neural algorithm for solving time-varying system of linear equations: a control-based approach,” *IEEE Transactions on Industrial Informatics*, vol. 15, no. 1, pp. 236–246, 2018.

[16] I. E. Shparlinski, “Linear equations with rational fractions of bounded height and stochastic matrices,” *The Quarterly Journal of Mathematics*, vol. 69, no. 2, pp. 487–499, 2018.

[17] J. Müller, “A converse result for Banach space convergence rates in Tikhonov-type convex regularization of ill-posed linear equations,” *Journal of Inverse and Ill-Posed Problems*, vol. 12, no. 5, pp. 639–646, 2017.

[18] A. Visconti, C. V. Schiavo, and R. Peralta, “Improved upper bounds for the expected circuit complexity of dense systems of linear equations over GF(2),” *Information Processing Letters*, vol. 137, pp. 1–5, 2018.

[19] A. K. S. Sabonchi and B. Akay, “Cryptanalysis of polyalphabetic cipher using differential evolution algorithm,” *Technicki vjesnik-Technical Gazette*, vol. 27, no. 4, pp. 1101–1107, 2020.

[20] K. Sreenu, M. Sreelatha, and W Scheduler, “Whale optimization for task scheduling in cloud computing,” *Cluster Computing*, vol. 22, no. 6, pp. 1–12, 2019.

[21] A. Bhownik and D. De, “mTrust: call behavioral trust predictive analytics using unsupervised learning in mobile cloud computing,” *Wireless Personal Communications*, vol. 117, no. 2, pp. 483–501, 2021.

[22] J. P. B. Mapetu, L. Kong, and Z. Chen, “A dynamic VM consolidation approach based on load balancing using Pearson correlation in cloud computing,” *The Journal of Supercomputing*, vol. 77, no. 6, pp. 5840–5881, 2021.

[23] V. Barthwal and M. M. S. Rauthan, “AntPu: a meta-heuristic approach for energy-efficient and SLA aware management of virtual machines in cloud computing,” *Memetic Computing*, vol. 13, no. 1, pp. 91–110, 2021.

[24] M. A. Almaiah and A. Al-Khasawneh, “Investigating the main determinants of mobile cloud computing adoption in university campus,” *Education and Information Technologies*, vol. 25, no. 4, pp. 3087–3107, 2020.

[25] L. S. Subhash and R. Udayakumar, “Sunflower whale optimization algorithm for resource allocation strategy in cloud computing platform,” *Wireless Personal Communications*, vol. 116, no. 4, pp. 3061–3080, 2021.

[26] Z. Jin, “Green city economic efficiency based on cloud computing and machine learning,” *Arabian Journal of Geosciences*, vol. 14, no. 11, pp. 1–16, 2021.

[27] A. Ng, A. Mba, and B. Ar, “Implement 3D video call using cloud computing infrastructure - ScienceDirect,” *Ain Shams Engineering Journal*, vol. 11, no. 2, pp. 363–375, 2020.

[28] S. Rahmani, V. Khajehvand, and M. Torabian, “Burstiness-aware virtual machine placement in cloud computing systems,” *The Journal of Supercomputing*, vol. 76, no. 1, pp. 362–387, 2020.

[29] G. Hodosi, A. Haider, and L. Rusu, “Risk factors in cloud computing relationships: a study in public organizations in Sweden,” *Procedia Computer Science*, vol. 181, no. 1, pp. 1179–1186, 2021.

[30] S. Alebrahim and I. Ahmad, “Task scheduling for heterogeneous computing systems,” *The Journal of Supercomputing*, vol. 73, no. 6, pp. 2313–2338, 2017.

[31] C.-Y. Chen, “Task scheduling for maximizing performance and reliability considering fault recovery in heterogeneous distributed systems,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 2, pp. 521–532, 2016.

[32] H. He, G. Xu, S. Pang, and Z. Zhao, “AMTS: adaptive multi-objective task scheduling strategy in cloud computing,” *China Communications*, vol. 13, no. 4, pp. 162–171, 2016.