Research Article

Stochastic Matrix Modelling and Scheduling Algorithm of Distributed Intelligent Computing System

Bo Han 1,2 and Rongli Zhang 1,2

1 College of Mathematics and Computer Application, Shangluo University, Shangluo 726000, China
2 Engineering Research Center of Qinling Health Welfare Big Data, Universities of Shaanxi Province, Shangluo 726000, China

Correspondence should be addressed to Bo Han; 232021@slxy.edu.cn

Received 7 June 2022; Revised 30 July 2022; Accepted 5 August 2022; Published 28 August 2022

Academic Editor: Ning Cao

Copyright © 2022 Bo Han and Rongli Zhang. 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.

Parallel and distributed processing has always been a hot field of scientific and technological research, development, and application. It is an important solution in the fields of scientific computing and data service processing, such as weather prediction, wind tunnel Reynolds numerical calculation, and financial services. Intelligent cloud computing has higher requirements for high-capacity and efficient computing. The ability of existing computing system has been difficult to meet its needs. It is necessary to establish an intelligent computing system with the self-organizing ability and realize efficient task scheduling. Since the coordination of computing and storage resource scheduling becomes the key to scheduling, this study designs scheduling tasks based on a large-scale multi-task distributed system, establishes the model of distributed intelligent computing system and the multi-objective optimization model of the task scheduling problem, and designs the IPSO algorithm combined with improved particle swarm optimization algorithm according to the model. First, the particle swarm optimization algorithm is used to generate the initial scheduling scheme, then the ant colony algorithm is initialized, and the final scheduling results are generated. Simulation results show that the performance of the algorithm has obvious performance advantages compared with the improved particle swarm optimization algorithm and the improved ant colony algorithm. In addition, this study presents the task migration conditions and optimization methods under the dual objectives of makespan and availability. This optimization operation increases the system availability without increasing the scheduling length. In the distributed system with heterogeneous availability, the algorithm is effective in the dual objective performance optimization of task completion time and system availability.

1. Introduction

As a new information service model, cloud computing is a hot spot of current research. It provides users with reliable and cheap computing resources on demand by means of service access [1]. With the help of virtualization technology, cloud computing transforms the large-scale and complex physical resources in the cloud environment into different kinds of virtual resource pools for unified management, and automatically deploys the tasks submitted by the cloud, so that service buyers can use computing resources without increasing the cost of purchasing and maintaining resources [2]. One of the key problems to be solved in the implementation of such a platform is how to schedule resources effectively [3]. The core of resource scheduling is to efficiently establish the mapping relationship between resources and tasks, which mainly includes two levels of scheduling: (1) realize the scheduling and allocation of virtual resources, that is, dynamically formulate the mapping relationship between tasks and virtual resources; (2) To realize the scheduling and allocation of physical resources is to formulate the corresponding relationship between virtual machine instances and physical hosts. An efficient scheduling strategy should realize the user’s expectation of the quality of service (QoS) as much as possible, and achieve the purposes of load balancing and green energy saving, as well as the requirements of fault tolerance, scalability, and security [4, 5].
For the static scheduling algorithm, they divided the static scheduling algorithm into independent tasks, duplication, UNC, and BNP, and the scheduling algorithm involves every processor in the system. When the application or task in the system, and the processor structure and the communication between processors are changing in real-time, it may lead to the overload of some processors, which may make the whole execution time longer than the optimal or near-optimal result [13]. Therefore, in the real-time process, it is necessary to reorder the tasks, so as to avoid the overload of some processors. From the perspective of load balancing, it can be found that the dynamic scheduling method can better distribute the load of processors, ensure that the load of each processor remains relatively stable, and avoid the overload of some processors, so as to give full play to the role of processor resources to the greatest extent [14]. At this stage, many dynamic scheduling strategies are widely used. For example, gradient model strategy, sender initiation strategy, prediction-based strategy, and so on [15]. The main principle of this scheduling technology is to search the solution space of the problem through algorithm-oriented random selection. Therefore, this process is not achieved by simple random search. Quasi-random scheduling technology generates new results by combining the previously obtained search result knowledge and the characteristics of random search. At present, the widely used heuristic algorithm (such as the genetic algorithm) belongs to quasi-random scheduling technology, and its scheduling time overhead is usually longer than that of other scheduling algorithms [16, 17].

The rest of this study is organized as follows. The second part discusses distributed intelligent computing architecture and stochastic matrix modeling. In the third part, the research on task scheduling algorithm based on the Improved Particle Swarm Optimization Algorithm is studied. Section 4 presents the test results. Finally, the full text is summarized in Section 5. The contributions are summarized as: (a) Establish an intelligent computing system with a self-organizing ability to achieve efficient task scheduling; (b) The distributed intelligent computing system model and the multi-objective optimization model of the task scheduling problem, and based on this model, the IPSO algorithm is designed to combine with the improved particle swarm optimization algorithm to improve efficiency; (c) Task migration conditions and optimization methods under the dual goals of maximum completion time and availability. This optimization operation improves the system availability without increasing the scheduling length.

2. Distributed Intelligent Computing Architecture and Stochastic Matrix Modelling

2.1. Distributed Intelligent Computing Architecture Design. Cloud computing is a computing-based network model. Users submit jobs (or requests) of their computing resources (such as operating system (OS), programming environment, software package, etc.) through the network. After configuring the job, users usually do not know the details of the execution environment. The term “cloud” means a sense of abstraction between users and their computing resources. The cloud computing model is based on a distributed system architecture, which allows users to use computing resources from the cloud. Dynamic allocation of computing resources is achieved based on the cloud computing model. The dynamic scheduling method is called the semi-distributed processing method. For the fully distributed dynamic scheduling method, its scheduling control involves every processor in the system. When the application or task in the system, and the processor structure and the communication between processors are changing in real-time, it may lead to the overload of some processors, which may make the whole execution time longer than the optimal or near-optimal result [13]. Therefore, in the real-time process, it is necessary to reorder the tasks, so as to avoid the overload of some processors. From the perspective of load balancing, it can be found that the dynamic scheduling method can better distribute the load of processors, ensure that the load of each processor remains relatively stable, and avoid the overload of some processors, so as to give full play to the role of processor resources to the greatest extent [14]. At this stage, many dynamic scheduling strategies are widely used. For example, gradient model strategy, sender initiation strategy, prediction-based strategy, and so on [15]. The main principle of this scheduling technology is to search the solution space of the problem through algorithm-oriented random selection. Therefore, this process is not achieved by simple random search. Quasi-random scheduling technology generates new results by combining the previously obtained search result knowledge and the characteristics of random search. At present, the widely used heuristic algorithm (such as the genetic algorithm) belongs to quasi-random scheduling technology, and its scheduling time overhead is usually longer than that of other scheduling algorithms [16, 17].
environment. Gartner defines cloud computing as a computing style in which large-scale expanded functions use Internet technology to provide services to external customers. The National Institute of Standards and Technology (NIST) provides the following definitions: cloud computing is widespread, convenient, on-demand network access, shared services, storage, models, and applications of configurable computing resource pools (e.g., servers, networks), which can be quickly configured and published with minimal e-business management, service, and provider interaction. It has the following key characteristics: virtualization, standardization, automation, flexibility fast adaptation, and on-demand billing. The cloud computing platform aggregates a large number of physical resources through network interconnection, forms various shared resource pools with the help of virtualization technology, and deploys various management tools for unified management. Its architecture is shown in Figure 1.

The distributed storage solution relies on the elastic storage, distributed object storage, operation and maintenance, and business distribution management system of the cloud computing platform, and integrates the third-party enterprise online backup software, personal network disk, personal media upload and sharing software, allowing cloud storage operators to provide low-cost/high-performance network storage services for individual consumer users and online backup and recovery services for enterprise users. In the scenario of distributed storage platform and enterprise backup/recovery application software, as well as personal network disk and personal media upload/sharing software, the portal and UI interaction interface with cloud storage application users and their application-related core business functions (such as permission management, breakpoint renewal, etc.) are supported by third-party cooperative application software. The distributed computing technology architecture figure is designed in Figure 2.

As is shown in Figure 2, the architecture includes a physical resource layer, virtual resource layer, resource service and scheduling layer, data service layer, service management middleware, and service interface layer. Generally speaking, the generally recognized cloud architecture is divided into three levels: infrastructure layer, platform layer, and software service layer. The corresponding names are IaaS, PaaS, and SaaS. IaaS, infrastructure as a service. The distributed storage platform of the operator provides the back-end support for the enterprise backup/recovery application software cooperated by the third party, as well as the personal network disk and personal media upload/sharing software in the form of IaaS. At the same time, the snapshot mechanism and remote disaster recovery mechanism based on object storage provides the storage cloud data with the opportunity to restore the stored data content at the latest snapshot time in the event of a greater disaster.

The distributed intelligent computing system described in this study is physically composed of three parts: several special or general computing nodes, high-speed bus network, and solid-state mass storage for public storage. Among them, all nodes are connected and communicate with each other through high-speed bus network. The mass memory stores the intelligent application program and the location description file of the intelligent application program. Each node can download the required application...
program from the corresponding location of the memory through the high-speed bus network according to the location description file; after the node obtains the task, it will write the human operation description file in the mass storage. The operation description file contains the information of the task execution node and the information of the task execution degree. When the node fails and cannot continue to perform the task, the other nodes can continue to perform the task according to the task execution degree, so as to improve the operation efficiency of the computing system. The mass storage also stores the increasing real-time big data, for online intelligent learning and training.

2.2. Large-Scale Task Distributed Scheduling Model. All resources in enterprises or scientific research institutions are connected together through the Internet, but they may belong to different domains. Each domain has one or more schedulers to handle incoming tasks. Due to the lack of necessary resources or inefficiency, tasks may be transferred to other domains. Taking cloud computing as an example, users do not care where or how their task requests are processed. If we want to use Google’s document editor to edit files stored in Amazon cloud devices, this request can be broken down into two tasks and processed in two different domains. Figure 3 shows a large-scale distributed system. Resources are separated by the management domain, and many schedulers are distributed in the whole system. Users submit their tasks to these schedulers, and the scheduler in the background assigns corresponding resources to execute them. This study designs a hierarchical scheduling model (as shown in Figure 3). The scheduler at a higher level has a broader vision, so this layered design can reduce communication and improve efficiency. Ultimately, the processor assigns tasks to them. A PE belongs to at least one or more autonomous domains and is managed in multiple schedulers. A scheduler can communicate with the PES it manages. In other words, the scheduler can know the status of each PE it manages. The scheduler at the same layer manages the elements of the next layer. Schedulers are connected by thin lines, which represent that they are adjacent and can communicate. When scheduling, the task can be assigned to the PE directly managed by the scheduler or the PE correspondingly managed by its adjacent scheduler. If the task still cannot be reasonably allocated, the scheduler will hand it over to a higher-level scheduler.

Resources in distributed systems can be divided into three roles: (1) PE, which is mainly used for computing; (2) Homo schedulers and other Homo schedulers in the same group work together to process the received tasks; (3) Heter-scheduler, able to accept or reject tasks independently to maximize their own interests; As shown in Figure 3, schedulers that share the same interests and fully cooperate in the execution of tasks are divided into a group. This group is called scheduler Homer. Usually, a domain forms a group. The group is an independent and autonomous entity whose behavior is driven by its own interests. Groups are linked by Heter-scheduler. The resource can be a PE, home scheduler or Heter-scheduler. PE is the responsibility of one or more schedulers. In the middle layer, quadrangles represent a domain, and Homo schedulers are divided in groups. They assign tasks to other schedulers in the same group or PE attached to it. Each group selects one or more Heter-schedulers as representatives to contact the Heter-schedulers.
of other groups. In fact, Heter-scheduler and Homo scheduler can be the same resource. The tasks accepted by Heter-scheduler are passed to Homo schedulers and finally allocated to PE.

3. Research on Task Scheduling Algorithm Based on Improved Particle Swarm Optimization Algorithm

3.1. Random Matrix Model Capable of Computing System Tasks. In this study, the non-decomposable program in the task model is recorded as task\(_0\), and task consisting of one or more task\(_0\). The task\(_0\) is independent of each other, and the task length and storage occupation of each task \( o \) are known. If the task can be decomposed into multiple task\(_0\), then in the process of executing the task, it can be decomposed and multiple task\(_0\) can be completed by different computing nodes in the computing system. You can include \( t \) in one calculation task\(_0\). The task is described by the number of tasks, task length and storage space occupied, and the \( i \)th calculation task is recorded as task\(_{ij}\), then task\(_{ij}\) = \{\( \lambda _{task\_0i}, task\_length_{ij}, task\_memory_{ij}\}\}. In the computing system, there are \( n \) independent task\(_0\) assigned to \( m \) computing nodes in the computing system (\( n > m \)) Where the task set is expressed as \( T = \{ task\_01, task\_02, \ldots , task\_0n \} \), the calculation node set is expressed as \( S = \{ s_1, s_2, \ldots , s_m \} \), task\(_{0j}\) represents the \( j \)th subtask; \( S \), represents the \( i \)th node. Each subtask is executed by only one node. The allocation relationship between Task set \( T = \{ task\_01, task\_02, \ldots , task\_0n \} \) and the calculation node set \( S = \{ s_1, s_2, \ldots , s_m \} \) can be expressed by matrix \( X \) as:

\[
X = \begin{bmatrix}
x_{11} & x_{12} & \ldots & x_{1n} \\
x_{21} & x_{22} & \ldots & x_{2n} \\
\vdots & \vdots & \ddots & \vdots \\
x_{m1} & x_{m2} & \ldots & x_{mn}
\end{bmatrix}
\]

(1)

where \( x_{ij} \) represents the allocation relationship between subtask task\(_0j\) and node \( s_i \).

For users, it is expected that the computing system can complete the computing task as soon as possible. Set \( ET_{ij} \) as the expected running time of the subtask task\(_0j\) at node \( s_i \), which is similar to the mapping relationship \( X \). The running time matrix \( ET \) can be obtained as follows:

\[
ET = \begin{bmatrix}
ET_{11} & ET_{12} & \ldots & ET_{1n} \\
ET_{21} & ET_{22} & \ldots & ET_{2n} \\
\vdots & \vdots & \ddots & \vdots \\
ET_{m1} & ET_{m2} & \ldots & ET_{mn}
\end{bmatrix}
\]

(2)

\[
ET_{ij} = \frac{\sum_{j=1}^{n} \text{Length}_{task\_0j}}{\sum_{i=1}^{m} \text{process}_{s_i}}
\]

Let \( b_i \) be the start time of the \( j \)th subtask, \( CT_{ij} \) is the subtask task\(_0j\) at node \( s_i \). The expected completion time, \( \max \{ CT_{ij} \} \) is the time for the computing system to complete the whole computing task, then

\[
CT_{ij} = b_j + ET_{ij},
\]

\[
C_{X_{\max}} = \max \{ CT_{ij} \}.
\]
The normal operation of intelligent computing system needs sufficient energy supply. It is expected that the intelligent computing system can reduce energy consumption as much as possible when it meets the requirements of service quality and users’ computing needs. CPU is the most important energy-consuming component in computing-intensive tasks. The energy consumption of the \( i \)th node in time \( t \) can be expressed as

\[
W_i = \int_0^t P(\mu_i(t)) \, dt. \tag{4}
\]

The total energy consumption of the system is expressed as

\[
W_{X_{sys}} = \sum_{i=1}^{m} W_i. \tag{5}
\]

The computing nodes in the computing system are in the same position. In the process of operation, they are expected to be in a state of load balance, that is, there are no nodes with long-term high load or low load. Define the ideal completion time \( ST_i \) of the whole task as the total instruction length of \( n \) computing tasks divided by the sum of the computing speeds of \( m \) computing nodes, then

\[
ST_i = \frac{\sum_{j=1}^{n} x_{ij} \cdot ET_{ij}}{\sum_{i=1}^{m} \text{process}_{s_i}}. \tag{6}
\]

Definition of load balancing degree \( LB_X \) of computing system under distribution scheme:

\[
LB_X = \sqrt{\frac{\sum_{i=1}^{m} (ST_i - ST)^2}{m}}. \tag{7}
\]

3.2. Improved Particle Swarm Optimization. The basic particle swarm optimization algorithm is suitable for the extreme value search in the continuous domain, and the problem of executing task node selection in the computing system is a discrete problem. Therefore, it is necessary to further deal with the scheduling scheme reasonably before the iteration of the particle algorithm can be carried out.

Let the particle swarm contain \( k \) particles. For the \( i \)th particle, define the \( n \)-dimensional position vector \( x_{ij} = [x_{i1}, x_{i2}, \cdots, x_{in}] \), \( n \) is the number of tasks, the value of the \( j \)-dimensional \( x_{ij} \) in the position vector \( t \) represents the \( i \)th particle, and the \( j \)th task assignment is performed by the \( x_{ij} \) node. In order to improve the performance of particle swarm optimization in task scheduling algorithm, inertia weight \( \omega \) Make dynamic adjustment, select a large inertia weight in the initial stage, expand the global search ability of the algorithm, and decrease linearly with the increase of the number of iterations, so as to improve the local search accuracy in the later stage of the iteration, and set the predetermined maximum number of iterations as \( t \). The velocity and position formula of an improved particle swarm is as follows:

\[
v_{ij}^{t+1} = \omega v_{ij}^t + c_1 r_1 (p_{ij}^t - x_{ij}^t) + c_2 r_2 (p_{gij} - x_{ij}^t),
\]

\[
x_{ij}^{t+1} = x_{ij}^t + v_{ij}^{t+1}, \tag{8}
\]

The implementation process of the scheduling algorithm based on improved particle swarm optimization algorithm is as follows: Step 1 defines three objective functions of the task scheduling problem Step 2 set the relevant parameters of the selection algorithm Step 3 initializes the position and speed of the particle swarm Step 4 calculate the objective function value of each particle and find out the individual local Pareto optimal solution set and global Pareto optimal solution set Step 5 determines the individual extreme value phbest and the global extreme (28) value gbest of each particle Step 6 updates the particle speed and position. Step 7 if the set number of iterations is reached, output the initial selection result of task allocation; otherwise, jump to Step 4 Step 8 is based on the initial scheduling result obtained by particle swarm optimization algorithm Step 11 ant colony algorithm reaches the maximum number of iterations, retains the value of the last generation of global ants as the global optimal solution set, ends the algorithm and outputs the scheduling results, otherwise, skip to Step 5 (Figure 4).

4. Analysis of Test Results

4.1. Experimental Environment and Simulation Settings. In order to evaluate the IPSO algorithm proposed in this study, CloudSim, a distributed computing simulation platform developed by the University of Melbourne, Australia, is used to extend the datacenter broker class and add a custom task scheduling algorithm to simulate the scheduling algorithm.

The simulation environment settings are as follows: assuming that the resource pool size is 14, it contains 10 computing resources and 10 storage resources, in which 8 resources are both computing and storage resources. The processing capacity of the processor is in MIPS. Therefore, the size of the task is calculated in millions of instructions (MI). Resources have different operating systems, Windows or Linux. The processing capacity of the processor is evenly distributed within the range of [1000–4500], and the bandwidth between resources is randomly selected between 10 mbit/s and 100 mbit/s. The access delay and response time of storage resources and the availability of computing resources are ignored. The amount of data contained in a file on the storage resource follows a normal distribution, ranging from 5 GB to 10 GB. In this experiment, the parameters of the genetic algorithm and tabu algorithm are set as follows: if the mutation rate or crossover rate is too large, the algorithm becomes a random search; if it is too small, the diversity of algorithm will be lost.
The grid simulation toolkit is GridSim, and a task is a Gridlet object. Five computing nodes with the computing power of 400 MIPs, 600 MIPs, 800 MIPs, 1000 MIPs, and 1200 MIPs are selected. Under the scheduling scale of 50 tasks with the length between 5000 mi and 10000 mi, and 500 tasks with the length between 5000 mi and 10000 mi, the discrete particle swarm optimization algorithm (DPSO) and elite ant algorithm (eaco) are compared with the IPSO algorithm in this paper Discrete particle swarm part, \(\omega_{\text{max}} = 0.9, \omega_{\text{min}} = 0.4, C_1 = C_2 = 1.5\), the population size is 20, and the number of iterations is 40 generations; Elite ant colony, with a population size of 20, \(a = 1, b = 1, \rho = 0.1, q = 10\), 160 iterations As a comparison, the parameters of DPSO algorithm are the same as those of discrete particle swarm optimization, and the parameters of eaco algorithm are the same as those of elite ant colony. The number of iterations of both algorithms is 200 generations.

4.2 Verification of the Superiority of the Algorithm. First, the performance comparison of the three algorithms with the shortest execution time as the single goal is simulated. The three algorithms are run 20 times respectively, and the results of two scheduling scales are obtained, as shown in Figures 5(a) and 5(b).

It can be seen from Figure 5 that when 50 subtasks are set, the convergence speed of the DPSO algorithm is faster. After about 20 iterations, that is, 25 converges to about 116.4 s to complete the task, and the convergence of EACO algorithm is completed in about 150 generations. However,
the optimization effect is better, about 118.9 s to complete the task, and IPSO algorithm converges in about 120 generations. The optimization effect is the best, about 109.6 s to complete the task. It can be seen that although the algorithm converges slower than DPSO algorithm, however, the optimization effect is improved by about 6.8 s; IPSO algorithm converges about 30 generations earlier than EACO algorithm, and the optimization effect is also improved by about 2.2 s. When 500 subtasks are set, about 30 generations of DPSO algorithm converge to 1178 s, about 10 generations of EACO algorithm converge to 1123 s, and about generations of IPSO algorithm converge to 1116 s.

It can be seen that with the increase of the number of tasks, IPSO algorithm still has a better optimization effect than EACO algorithm and DPSO algorithm. The simulation results under two scheduling scales show that the IPSO algorithm proposed in this paper has achieved good improvement in time performance and optimization performance under a single objective. PSO algorithm only takes makespan as the optimization index, and does not fully consider the availability of resources, so its makespan is better than the EACO algorithm.

The parallel operation of IPSO may have a certain negative impact on the performance due to communication and other reasons. Compared with DPSO, IPSO can find the scheduling scheme faster through parallelization, resulting in the same execution time. In addition, to some extent, IPSO algorithm can avoid falling into local optimization and find a better solution by searching from multiple initial solutions at the same time. As shown in Figures 6(a) and 6(b), the optimization process of IPSO and DPSO is explained. In DPSO, there is only one initial population. After several iterations, it sometimes falls into local optimization, as shown in Figure 6(a). After repeating the above experiments, this makes preliminary statistics: the probability of the DPSO algorithm falling into local optimization is close to 1/8. Figure 6(b) shows the results of IPSO in the same environment. Different colors represent different initial populations. It can be seen from the figure that one population evolves to the suboptimal solution, and other populations reach the maximum. This shows that IPSO converges to the optimal solution with a higher probability than DPSO.

4.3. Convergence and Utilization Analysis of Optimal Response Strategy. The initial strategy of each scheduler is a vector composed of elements $1/n$. Then, each scheduler improves and updates its strategy in each iteration. In the first set of experiments, we study the convergence of the optimal response strategy, that is, the optimal scheduling strategy when the other schedulers keep the strategy unchanged. The experiment simulates a heterogeneous system with 32 schedulers and sets the system utilization rate to 60%, that is, the arrival rate for the whole system is 2316. The initial scheduling strategy is generated randomly, and one scheduler runs the algorithm while the other schedulers maintain their initial strategy.

The above experiments are repeated 20 times. The initial strategy of the scheduler is randomly generated each time, and the average response time of 20 experiments is taken. Figure 7(a) shows the difference in the expected response time of the last two strategies of the scheduler when the system utilization is 60%. After about 200 iterations, the absolute difference decreases to $10^4$, which is considered to reach convergence. Figure 7(b) shows the number of iterations when converging under different system utilization. With the increasing utilization of the system, our algorithm needs more iterations to obtain the best response strategy. This is because the system will reach a stable state only after a
large number of tasks come. For high utilization of 90%, the convergence rate increases to about 460 iterations.

Pareto fairness is a kind of value judgment. It defines fairness from the perspective of social welfare and measures the result of resource allocation from the perspective of efficiency, so it is fairness in the sense of efficiency. Admittedly, it is difficult to achieve absolute Pareto equity in real economic activities, but it can be close to Pareto equity to the greatest extent through the improvement of efficiency. It is precisely based on this that Pareto equity has a very important guiding significance for re-understanding the current economic system and distribution system. It can be seen from the above experimental results that when the scheduling scale is 50 tasks, the optimization result of DPSO is $T_{\text{max}} = 125.78$; Energy consumption $W_1 = 2.99 \times 10^4$ J; Load balance $LB_1 = 11.5$. The optimization result of IPSO algorithm is $T_{\text{max}} = 115.2$; Energy consumption $W_2 = 2.85 \times 10^4$ J; Load balance $LB_2 = 3.1$. Compared with DPSO algorithm, the optimization

\[ x_1 = 0, x_2 = 0 \]
performance of IPSO algorithm has been greatly improved, which avoids the problem of premature convergence to local extremum of DPSO algorithm.

5. Conclusion

This study presents the mathematical model of an intelligent distributed computing system and the multi-objective Pareto optimal optimization model of task scheduling problem. Aiming at the problem of data-intensive applications in distributed systems, a new scheduler model is proposed. This model divides the traditional scheduler into three layers, and the scheduler plays different roles in each layer: in the first layer, it is a general scheduler, which is responsible for collecting information. The second layer and the third layer each contain several small scheduling units, so that the task scheduling can be carried out in parallel. At the same time, the scheduling problem in heterogeneous distributed systems with availability constraints is defined as an integer programming problem. On this basis, an optimization method to increase system availability. Through performance comparison, it is verified that the algorithm can effectively improve the time performance and availability of heterogeneous systems when dealing with multiple types of tasks. The dynamic weight is used in the discrete particle swarm optimization algorithm, and the particle swarm optimization algorithm is used to generate the scheduling scheme. It makes comprehensive use of the fast convergence speed of the particle swarm optimization algorithm and the strong local optimization ability of ant colony algorithm, which not only achieves better optimization performance for the single objective of the shortest task completion time; It also plays a better optimization performance in finding the Pareto optimal state with the lowest cost, the shortest task completion time and load balancing. It is our current work to verify and transplant this research into the new form of distributed computing—Cloud Computing and edge computing.

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 that they have no conflicts of interest or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by Engineering Research Center of Qinling Health Welfare Big Data (Universities of Shaanxi Province), Shaanxi Province 2016–2017 Basic Education Major Bidding Project (The level of Students’ Artistic Education in Shaanxi Province Research on Measurement System, No. ZDKT1606), Shaanxi Education Society 2016 Project of the Year (Research on Strategies for Promoting Adolescent Health Growth under the Network Environment, No. SJHZDKT20160504), and Shaanxi Province Educational Science Planning Project (Research on Flipped Classroom Teaching Mode of Software Development Courses in the Era of “Internet +,” No. SGH17H342).

References


