Task Scheduling Method of Wireless Sensor Multimedia Big Data Parallel Computing Based on Bee Colony Algorithm

Tongxi Lin and Fulong Zhong(✉)

Guangzhou Huashang Vocational College, Guangzhou 511300, China
Linxixi12021@163.com

Abstract. In the parallel computing task scheduling of wireless sensor networks, the resources of the edge server itself are limited, and the scheduling speed is slow. To solve this problem, a task scheduling method for wireless sensor multimedia big data parallel computing based on bee colony algorithm is designed. Design GSM wireless signal processing multi-core scheme and implement wireless signal processing. Message passing model is used as parallel programming model in scheduling. Improved the bee colony algorithm, designed the artificial bee colony algorithm, and realized the task scheduling of multimedia big data parallel computing based on the artificial bee colony algorithm. Test the scheduling performance of the design method. The test results show that the scheduling speed of the design method is fast, and the overall resource utilization is higher than 0.91.

Keywords: Bee Colony Algorithm · Wireless Sensor · Parallel Computing · Computing Task Scheduling

1 Introduction

In wireless sensor networks, the number of network edge devices and the amount of communication data between devices are growing year by year [1, 2]. However, with the increase of wireless device functions, various applications and services require more and more computing resources of the device itself. Due to the limitations of the resources and battery capacity of general wireless devices, these computing intensive applications can not be effectively and timely processed on the wireless devices themselves. The tasks on the wireless devices are migrated to the cloud computing center for processing, which meets the needs of the tasks for computing resources, at the same time, when information is transmitted between the remote cloud and wireless devices, there is also a large communication delay, which increases the response time of users waiting for task execution, thus affecting the user's service experience. In order to remedy the problem that transmission delay affects user service experience, some scholars have proposed a new computing model, that is, most of the tasks originally completed in the cloud computing center are transferred to the edge of the network. This new computing

B. Wang et al. (Eds.): ICMTEL 2023, LNICST 534, pp. 411–426, 2024.
https://doi.org/10.1007/978-3-031-50577-5_29

model provides computing resources for computing intensive and delay sensitive tasks on wireless devices by comprehensively utilizing idle resources widely distributed at the edge of the network, so as to meet the requirements of these tasks for computing resources and response time. This new computing model is called edge computing. Its main idea is that by setting some edge cloud nodes on the side of the network edge close to the user, the tasks generated by the wireless device itself can be directly unloaded to the edge cloud for processing. Moreover, compared with unloading the tasks to the remote Cloud Computing Center for processing, this method will generate less network transmission delay, so as to improve the timeliness of task processing. Although the communication delay between the user equipment and the remote cloud computing center is greatly reduced by setting the edge cloud node to process user tasks nearby, a new problem arises, namely, how to formulate an effective multimedia big data scheduling strategy for tasks unloaded to the edge cloud node. This is because the resources of the edge server itself are limited compared to the remote cloud, and the processing speed of each edge server is often inconsistent with the total amount of resources. At the same time, there is resource competition between tasks [3]. In a certain period of time, if more tasks are unloaded to the edge node and not properly handled, the user experience will be affected. Therefore, how to reasonably schedule the multimedia big data tasks unloaded to the edge server so that the edge server can use its own limited resources to handle more tasks, while ensuring the user's service experience is an important issue. Based on this background, the task scheduling method of wireless sensor multimedia big data parallel computing is studied. Use GSM wireless signal to process multi-core information and realize wireless signal processing. The message passing model based on MPI model is built as the basis of parallel programming. Improved the bee colony algorithm, designed the artificial bee colony algorithm, and realized the multimedia big data parallel computing task scheduling based on the artificial bee colony algorithm.

2 Wireless Sensing Multimedia Big Data Parallel Computing Task Scheduling

2.1 Wireless Signal Processing

A GSM wireless signal processing multi-core scheme is designed to implement wireless signal processing. The GSM physical layer software in the solution can support the receiving of base station and terminal information at the same time, and can support the control and continuous tracking of multiple cells and users.

Global System for Mobile Communications (GSM) is a digital mobile communication standard developed by the European Telecommunications Standards Organization. Its air interface adopts TDMA technology, and the signaling and voice channels are digital. The GSM physical layer software designed in this paper is mainly responsible for receiving baseband data transmitted by FPGA, performing physical layer processing such as despreading, despreading, demodulation and channel decoding, and then reporting the data to advanced software for further processing. The specific functional requirements are as follows: (1) According to the high-level configuration requirements, it can control multiple cells, report broadcast data and public transmission channel data

in real time, maintain cell synchronization in real time, and measure cell power periodically; (2) According to the high-level configuration requirements, multiple users can be processed at the same time, the dedicated channel data can be reported in real time, and the uplink and downlink dedicated link power can be measured periodically; (3) Continuously search the information of surrounding cells according to the requirements of high-level configuration, and report the cell frequency, cell ID and power information.

The GSM wireless signal processing multi-core solution system can process up to 4 frequency bands at the same time. After the high-level sends the task of a cell initial search, the scheduling system finds a core that can execute the task and distributes the task. The system needs to continuously perform cell initial search on a frequency band by frequency point basis. For each frequency point, no matter whether the search results are successful or failed, the search results will be reported to the senior management, who will decide the next processing flow [4]. Each task will have a structure to store the status of the task, so that the task can be decoupled from the core of executing the task. The structure for recording the initial search task of a cell is STRU_ CELL INIT-SEARCH INFO。The cell initial search task is implemented by using the FCCH. The search process is shown in Fig. 1:

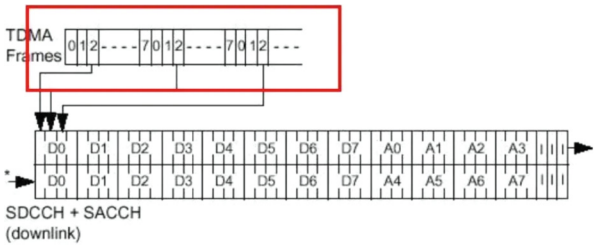

Fig. 1. Schematic representation of the search process

As shown in Fig. 1, GSM makes full use of the characteristics of FCCH frames to perform coarse frequency synchronization. The full name of FCCH is Frequency Control Channel. Its main feature is that all data in the frame is "0". Since the FCCH frame does not contain useful information, demodulation is not required. After GMSK modulation, a sine wave can be obtained. The frequency of the sine wave is 67 kHz, with obvious single peak characteristics. The FCCH is located in the 0 th time slot of the 51 superframe. Since then, it appears every 10 frames, which means that the FCCH can be found by detecting up to 11 frames. With this feature, the initial cell search task can be completed. Each time, 32 data are taken out from the input sequence, and then the eftpe coprocessor is called to perform fast Fourier transform to obtain a group of modulus values in the frequency domain. The corresponding power can be obtained by squaring these modulus values. Since the frequency corresponding to the eighth point is 270/32 $\times$ 8 = 67 kHz, we only consider the three points 7, 8 and 9 here. Remove the power values of points 7, 8 and 9, and calculate the average power values of the remaining 29 points. Find the maximum power value of 32 points. If the peak to average power ratio exceeds a certain threshold value, it means that the FCCH signal may be detected. Since

the length of a bit is 156, if the peak value appears for four consecutive times, we think that we have successfully captured the FCCH.

The cell initial search task execution process is as follows:

(1) For the frequency band to be searched issued by L3, the load balancer will select the appropriate kernel and send out the task;
(2) Call the function to obtain the data of the frame according to the FPGA channel number of the initial search request;
(3) Call the function to capture the FCCH;
(4) Judge whether the FCCH is captured successfully. If the capture is successful, and the number of frames is between 51 and 102 when the capture is successful, then you can call the function to report the success information to the higher layer and prepare for the capture of the next frequency point;
(5) If the capture fails, continue to analyze the information of the next frame [5]. If the number of captured frames is greater than or equal to 102, then you can call the function to report the failure information to the higher level and prepare for the capture of the next frequency point;
(6) When all frequency bands in this frequency band are searched, the initial search task of this cell is completed.

The structure for recording the search task of a specific cell is STRU_ CELL SPEC_SEARCH_INFO。

The purpose of cell specific search is to analyze the useful information of SCH.

The specific cell search task execution process is as follows:

(1) Task in C_ ENUM ACQ-FCCH-NO status, that is, the FCCH has not been captured. At this time, FCCH capture is required.

After successful capture, the task status changes to C_ ENUM-ACQ-FCCH YES。

(2) Task in C_ ENUM-ACQ-FCCH-YES status, that is, coarse synchronization status.

The successful capture of the FCCH means that the coarse synchronization has been completed. Next, we need to use SCH for fine synchronization [6]. The full name of SCH is Synchronization Channel, which always appears in the next frame of the FCCH. Therefore, the start position of the SCH can be calculated according to the start position of the FCCH, and then the function is called to obtain the frame data of the FPGA.

SCH contains 64 bit training sequence, and the autocorrelation of this training sequence is good, so the following functions can use this feature to perform fine synchronization and accurately calibrate the starting position of SCH.

Then call the channel estimation function for channel estimation, call the channel equalization function for channel equalization, call the Viterbi decoding function for Viterbi decoding, and finally call the CRC check function for CRC check. If CRC verification is successful, the calculated BSIC, TDMA frame number, channel power and other information will be sent to the upper layer, and the task status will change to C_ ENUM-ACQ_ _ BCCH[7]。If CRC verification fails, the task status changes to C_ ENUM-ACQ FCCH NO, return to step (1) to re analyze the data of the next frame.

(3) Task in C_ ENUM-ACQ-BCCH status, that is, tracking status.

According to the structure of 51 multiframes, when the frame numbers are 2, 3, 4 and 5, it is in the state of broadcast channel decoding, calling the channel estimation function for channel estimation, calling the channel equalization function for channel equalization, calling the Viterbi decoding function for Viterbi decoding, and finally calling the CRC check function for CRC check. If CRC verification is successful and this is a complete BCCH block, that is, when the frame number is equal to 5, data and information will be sent to the upper layer.

When the frame numbers are 1, 11, 21, 31, and 41, and the frame is in the SCH channel, the time offset adjustment function is called once to adjust the time offset.

When the frame number is 30, it is in the FCCH signal, and the frequency offset adjustment function is called again to adjust the frequency offset.

The structure for recording the cell creation task is STRU__ RRCPHY-GSM _ CELL-SETUP_ _ REQ。

The cell creation task execution process is as follows:

(1) Task in C ENUM RSRC_ SETUP COMMIT status, that is, establishment is allowed. Determine whether the current frame is the first frame of a block structure according to the channel combination type (channel combination 4 or channel combination 5) configured by the cell. If not, keep the C_ ENUM_ RSRC_ SETUP _COMMIT status until the first frame of a block structure is found. If yes, change the task status to C_ ENUM_ RSRC_ SETUP.
(2) Task in C_ ENUM_ _ RSRC-SETUP status, i.e. establishment.

If the frame is in the FCCH, obtain a frame data corresponding to the channel from the FPGA shared memory, and calculate the frequency offset for frequency offset tracking.

If the frame is in SCH, obtain a frame data corresponding to the channel from the FPGA shared memory, calculate the time offset, and calculate whether to report the cell power according to the reporting cycle and the current frame number.

If the frame is in BCCH or CCCH, obtain a frame data corresponding to the channel from the FPGA shared memory, and then call the channel estimation function to calculate the signal estimation, the channel equalization function to calculate the channel equalization, and the burst decoding function to perform decoding. If this frame is the last frame of a block, call the deinterleaving function, Viterbi decoding function and CRC check function to complete the final channel data decoding, and report the data and information to the upper layer.

L3 sends the physical layer link establishment message to notify the physical layer to establish the UE link.

STRU is the structure that records the user link creation task_ RRCPHY-GSM_ _ UE-RL SETUP_ _ IND。A structure is defined as a two-dimensional array. The reason is that the user link establishment is divided into uplink and downlink. However, the link processing of both uplink and downlink is the same, so the following link establishment tasks are executed without distinction:

(1) Task in C_ ENUM-RSRC-SETUP-COMMIT status, that is, establishment is allowed. If the current frame is not the first frame of a block structure, the task status remains C_ ENUM RSRC-SETUP_ _ COMMIT。If yes, the status of the task changes to C_ ENUM RSRC-SETUP。

(2) Task in C_ ENUM _ RSRC-SETUP status, i.e. establishment. Obtain the current user channel configuration information, judge whether the frame channel type is the configured dedicated channel, if so, obtain the current user FPGA channel number; Acquire the data of the current frame according to the FPGA channel number and frame starting position; Then call channel estimation function to calculate signal estimation, channel equalization function to calculate channel equalization, and burst decoding function to perform decoding. If this frame is the last frame of a block, call the deinterleaving function, Viterbi decoding function and CRC check function to complete the final channel data decoding, and report the data and information to the upper layer.
(3) Task in C_ ENUM-RSRC_ _ RECFG-COMMIT, namely reconfiguration. Then fill in the structure again according to the instructions.

In the multi-core scheme of GSM wireless signal processing, the multi-core scheme adopted is MSC8157 DSP, which has 6 cores. In order to maximize the utilization, it is natural to make full use of these six cores. The design scheme of the system is as follows:

Create a process in Core 1 for resource maintenance and task scheduling, which is called load balancer. It will receive the control message from the high-level L3 configuration, and then call the scheduling algorithm to allocate the tasks to the cores 1 to 6. The scheduling algorithm used is OPMMA resource scheduling algorithm. The specific steps are as follows:

Step 1: Define the task collection:

$$\overline{T} = \left\{\overline{T}_1, \overline{T}_2, \cdots, \overline{T}_n\right\} \tag{1}$$

In formula (1), $\overline{T}_n$ represents task n.

Step 2: Determine which scene the task set belongs to. If scenario 1 or scenario 2, perform the following step, otherwise perform step 10.

Step 3: In scenario 1, the largest task $\overline{T}_n$ in the task set is put on the slowest physics machine F_1, the maximum task execution time is L_1, and in scenario 2, the maximum task $\overline{T}_n$ is put on the fastest physics machine F_2, and the maximum task execution time is L_2.

Step 4: Update the task set and physical resource set, and delete the largest task and the corresponding physical machine from the set.

Step 5: In Scenario 1 and Scenario 2, start the task with the smallest average length, assign it to the current fastest physical machine for execution, and then accumulate the execution time of these tasks on the physical machine. When the cumulative execution time is greater than the maximum task execution time in Step 3, schedule the last task assigned to the physical machine to the next physical resource I_1.

Step 6: Update the task set and physical resource set again. Here, delete the two tasks with the minimum completion time from the meta task set, and delete the physical resources from the physical machine set.

Step 7: Execute steps 5 and 6 until the physical machine set is empty, and then exit the loop this time.

Step 8: If the task set is not empty, redefine the physical machine set F', the elements in the set are physical resources, and then continue to step 9, otherwise exit the full loop and have the scheduling task over.

The Machine collection F' is defined as follows:

$$F' = \left\{F'_1, F'_2, \cdots, F'_n\right\} \tag{2}$$

In formula (2), F'_n refers to the n th physical machine.

Step 9: Return to execute steps 2 to step 8, until the task set is empty, then exit the loop, and schedule the task is over.

Step 10: execute the the enhanced max-min algorithm until the task set is empty, exit the loop, and the scheduling task ends.

Nuclear 1−6 can handle any task in the community primary search task, specific community search task, community establishment task, and user link establishment task. They receive the data from the front-end FPGA, perform the corresponding tasks according to the resources assigned by kernel 1, and report the executed result information to the top level.

2.2 Parallel Programming

Message passing model is used as parallel programming model in scheduling. The programming model used in the study is the MPI model in the messaging model. MPI model is one of the most commonly used messaging models. MPI is very powerful, and it has good communication performance and good program portability. The message passing function of MPI is mainly realized through the following six basic calling functions, which constitute the minimum set required for MPI parallel programming. The functions are shown in Table 1.

Table 1. The role of each function

Serial number	Function	Specific role
1	MPI_Init	Iitialization
2	MPI_Comm_size	Determine the number of processes executing in parallel
3	MPI_Comm_rank	Determine the sequence number of a process
4	MPI_Send	Send message
5	MPI_Recv	Receive messages
6	MPI_Finalize	End MPI program

Including MPI_ Init function is an initialization function, which is called first in MPI program. The first statement in any MPI program is MPI_ Init function. This statement is called in C language as follows: int MPI_ Init(int*argc,char***argv)。 The argc and argv parameters need to be given in this call.

MPI_ Comm_ The size function can indicate how many processes are in the communication domain. When each process executes, it calls MPI_ Comm_ The size function allows you to know the number of processes executing in parallel. The calling method of this statement in C language is as follows: int MPI_ Comm_ size(MPI_Comm comm,int size)。Where comm is the given communication domain and size is an integer representing the number of processes.

MPI_ Comm_ The function of the rank function is to specify the serial number of a process. Different processes have different process serial numbers. Therefore, different processes can be distinguished by this process serial number, and the message transmission between different processes can also be easily realized. This statement is called in C language as follows: int MPI_ Comm_ Rank (MPI_Comm comm, int rank) where comm and MPI_ Comm_ Size is the same, rank is the process serial number, and the size of the process serial number is between 0 and (size-1).

MPI_ The function of Send is to send data between different processes. This statement is called in C language as follows: int MPI_ Send(void*buf,int count,MPI_Datatype_datatype,int dest,int tag, MPI_comm) MPI_ Send。It is to transfer a message from one process to another. The message is marked with tag, datatype and count. The message is sent from the sending data buffer with the starting address of buf, and the target process ID is dest.

MPI_ The Recv function is used to receive data between different processes. This statement is called in C language as follows: int MPI_ Recv(void*buf,int count,MPI_Datatype_datatype,int source,int tag,MPI_comm comm,MPI_Status status) MPI_ Recv is to receive a message from one process to another. The message flag is still tag, the message type is still datatype, and the maximum number of data that can be received is count. The message is received to the receive data buffer with the starting address of buf. Source represents the source of the data, that is, the process ID of the sending data. Status is the return status.

MPI_ The Finalize function is the last statement in the MPI program. After calling this function, the MPI program can be ended. If you want to end the MPI program but do not write this statement, the running result of the program will be confused. The calling method of this statement in C language is as follows: int MPI_ Finalize (void) After you do not need to call any MPI routine, you can call this function to eliminate the MPI environment, exit the MPI system, and execute other statements unrelated to MPI.

The calling process of MPI function is as follows: When entering MPI programming mode, the first thing to do is to run MPI_ Init function, initialize the MPI programming environment, and then determine the number of processes in the communication domain, that is, run MPI_ Comm_ Size function, and then obtain the process serial number of a process, that is, execute MPI_ Comm_ The rank function, and then all you need to do is to transfer messages between processes, and execute MPI_ Send and MPI_ Recv function. Finally, if you do not need to write MPI programs, execute MPI_ Finalize function, exit the MPI environment, and the entire MPI function call process will end.

It can be seen from the above calling functions and calling processes that parallel programming using MPI model is very simple and understandable. Calling six basic functions can achieve programming, and the compiled parallel program can be transplanted

to other places for application only with simple modification. These characteristics make MPI programming model widely used in the field of parallel computing.

MPI message consists of data and envelope. Data is the amount that needs to be transferred between processes, which should include the number of data to be transferred, the type of data and other information. As the name implies, the envelope should include the process serial number of the sending or receiving data end and the starting address of the corresponding data buffer, as well as the identification of the message and the range of the communication domain for message transmission.

The message transmission process of MPI is as follows: First, take the data information to be transmitted from the sending data buffer, and then add an envelope to the data information to clarify the relevant information of the sending end and the receiving end. The data information to be transmitted is represented by the following sets:

$$V = \{V_1, V_2, \cdots, V_m\} \tag{3}$$

In formula (3), V_m refers to the m th data information that needs to be transmitted.

Later, the assembled message is transmitted through the network. When the message reaches the receiving end, the message is disassembled, the data is removed from the message, and the data is delivered to the receiving data buffer, and the transmission of the data information is completed. The completed data information set is represented in the following formula:

$$V' = \left\{V_1', V_2', \cdots, V_m'\right\} \tag{4}$$

In formula (4), V_m' refers to the m th transmitted data information.

In the whole process, variable types and data types need to be matched to prevent some unpredictable situations.

MPI is required for message delivery_ Send function and MPI_ Recv receiving function can be completed only with the cooperation of Recv receiving function. Whether the message transmission action is completed depends on not only the sender but also the receiver.

(1) The sender and receiver start simultaneously in the process of both parties

No matter whether the sent message is larger or smaller than the default buffer provided by the MPI environment, since the receive action and the send action are started at the same time, process 0 will immediately transmit data.

(2) Receive action is later than send action

No matter whether the size of the message sent is larger or smaller than the default buffer provided by the MPI environment, since the receiving start time of process 1 is later than the sending start time of process 0, the message transmission will not start until the receiving action of process 1 is started. In this case, a long waiting time will be required in the message transmission process [8].

(3) Since the receiving start time of process 1 is earlier than the sending start time of process 0, the message transmission can be implemented immediately without waiting.

2.3 Data Scheduling of Parallel Computing Tasks

The task scheduling of multimedia big data parallel computing is realized based on the bee colony algorithm. An artificial bee colony algorithm is designed by improving the bee colony algorithm. The artificial bee colony is composed of three kinds of bees: hired bees, observation bees, observation bees, and reconnaissance bees. The honey gathering behavior of the bee colony corresponds to the optimization problem of the feasible solution, the location of the honey source corresponds to a feasible solution, the speed of the bees' foraging corresponds to the optimization speed of the feasible solution, the degree of income of the size of the honey source represents the quality of the feasible solution, and the maximum degree of income of the bee colony is the optimal solution in the optimization problem.

The basic steps to implement the artificial bee colony algorithm are as follows:

1. Initialize bee population

At the initial moment, all bees have no prior knowledge, they are reconnaissance bees, randomly search the honey source globally, and obtain the nectar amount of the honey source according to the situation of the honey source, that is, the "income degree" of the honey source.

The population parameters are as follows:

(1) Total number of bees m (generally defined as employment bees and observation bees are $M/2$ respectively);
(2) Maximum iterations maxCycle (one global search and one local search will occur during each search process);
(3) The maximum limit number of honey source stay Limit (local search Limit times, the honey source is not updated, then hire bees, observation bees are converted into reconnaissance bees).

M feasible solutions are randomly generated upon initialization with the following formula:

$$Y_j(m) = Y_{\min} + rand(0, 1)(Y_{\max} - Y_{\min}) \tag{5}$$

In formula (5), $Y_{\min}$ is the minimum solution; $Y_{\max}$ is the maximum solution; and $rand(0, 1)$ is a random number between 0 and 1.

The profitability calculation formula is:

$$d_j = \begin{cases} \dfrac{1}{1 + c_i} \\ 1 + a_{bs}(c_i), (c_i < 0) \end{cases} \tag{6}$$

In formula (6), c_i refers to the function value; $a_{bs}(c_i)$ refers to the absolute value function of c_i.

2. In step 1, according to the relative size of the "profitability" of all honeybees' honey sources, bees are transformed into hired bees and observed bees. Those with the highest profitability become hired bees, and those with the lowest profitability become

observed bees. Observing bees wait in the dance area. According to the swing dance of hired bees and other information, we know the nectar amount of honey source. The higher the nectar amount, the more observation bees will be recruited (the recruitment process will be completed with probability). The probability calculation function is as follows:

$$q_j = \frac{d_j}{\sum_{j=1}^{A} d_j} \tag{7}$$

In formula (7), A refers to the number of bees employed.

3. Each hired bee continues to collect honey near the original honey source (local search process), looks for other new honey sources, and calculates the new value of its profitability. If its profitability is high, the bee will replace the original honey source with the new honey source according to the greedy rule.
4. Each observation bee selects the honey source according to the probability that is proportional to the value of the honey source's income degree, and collects honey nearby to find other honey sources, just like the hired bees in step 3, if the following formula holds:

$$d_s > d_e \tag{8}$$

In formula (9), d_s refers to the yield of the surrounding new honey source; d_e refers to the income of the original honey source. Then observe that the bees convert to hire the bees, and the new honey source replaces the original honey source.

Current establishment:

$$\chi > L_{imit} \tag{9}$$

In formula (9), χ refers to the number of times the hired bees and observation bees search for the honey source; L_{imit} is the limit value.

At this time, if no honey source with higher profitability is found, you can choose to give up the honey source. At the same time, the role of the bee is changed from hiring bees or observing bees to detecting bees, and a new honey source is randomly generated.

6. Record the optimal honey source found by all bees, and skip to step 2 until the condition of maxCycle with the maximum number of iterations is met or the global optimal position is output when it is less than the optimization error.

The steps to apply the artificial bee colony algorithm to the task scheduling of multimedia big data parallel computing are as follows:

(1) The user submits the job to the master node. The ResourceManager in the master node places the job in the queue and its corresponding ApplicationMaster sends the resource request for the job.
(2) The NodeManager returns heartbeat information from the node to the Resource-Manager, and reports the computing and storage resources of the node to the ResourceManager.
(3) After the Resource Manager obtains the current running status and remaining resource information of each slave node, it uses an improved artificial bee colony algorithm to initialize and code the submitted job and the slave node to be used for computing tasks to generate an initial artificial bee colony.

(4) Calculate the income degree of honey source. According to the relative income degree of honey source of all bees, bees are transformed into hired bees and observed bees. Those with the highest income degree become hired bees, and those with the lowest income degree become observed bees.
(5) Each hired bee continues to collect honey near the original honey source (a local search process), looks for other new honey sources, and calculates the new value of its profitability. If its profitability is high, the bee will replace the original honey source with the new honey source according to the greedy rule; Each observation bee selects the honey source according to the probability proportional to the income value of the honey source, and collects honey near it to find other honey sources. If it finds that the income of the new honey source around it is higher, the observation bee switches to the hired bee, and the new honey source replaces the original honey source.
(6) If the number of times the hired bees and observed bees search for this honey source exceeds the limit value, and no honey source with higher profitability is found, they choose to give up this honey source, and the role of the bee changes from the hired bees or observed bees to the investigated bees, and a new honey source is randomly generated.
(7) Record the optimal honey source (i.e. the global optimal solution) found by all bees at present, and skip to step 4, until the conditions for the maximum number of iterations maxCycle are met or less than the optimization error, output the global optimal location, output the optimal sequence, i.e. the optimal scheduling scheme, according to which the cluster is allocated from the node's computing and storage resources to each job, otherwise, return to step (4).
(8) The ApplicationMaster of each job sends a periodic heartbeat to the ResourceManager to get the newly allocated resource container.
(9) After the ResourceManager receives the heartbeat information from the ApplicationMaster, the containers in the slave nodes assigned to it will be returned to the ApplicationMaster in the form of heartbeat response.

3 Experimental Test

3.1 Construction of the Experimental Platform

Test the scheduling performance of the designed task scheduling method for wireless sensor multimedia big data parallel computing based on bee colony algorithm. The experimental environment is built by 6 computers, using the virtualization software VMware Workstation to establish 11 virtual machines with Linux operating system as 11 nodes, select one node as the master node of the cluster, and the remaining 10 nodes as the slave nodes of the cluster. The master node is used to manage the allocation and management of computing resources and jobs of each slave node, and the slave node is used to store and calculate data. The Linux operating system used in the experiment is Cent OS 6, the Hadoop version is Hadoop2.5.1, and the JDK version is jdk1.7. The experiment uses three machines with different configurations to build a heterogeneous cluster environment for wireless sensor multimedia big data parallel computing. The specific machine parameter configurations are shown in Table 2, and the configuration information of each node is shown in Table 3.

Table 2. The configuration of each machine parameters

Hardware category	Desktop	Notebook	Desktop
Operating system	Windows 10	Windows 10	Windows XP
CPU	Core dual core processing i6-2400u	Intel(R)Core (TM)-i7–4680	Intel(R)Core (TM)-i9–7458
Dominant frequency	5.3 GHZ	5.2 GHZ	5.6 GHZ
Memory	12 GB	16 GB	8 GB
Number of units	2	1	3
Number of virtual machines	1	3	2

Table 3. Configuration of Cluster Node Parameters

Node type	Processor	IP address	Memory
Master	i7–7452	174.198.0.0	80G
Slave01	i7–7452	174.198.0.1	120G
Slave02	i7–7452	174.198.0.2	120G
Slave03	i7–7452	174.198.0.3	120G
Slave04	i9–9562	174.198.0.4	240G
Slave05	i9–9562	174.198.0.5	240G
Slave06	i9–9562	174.198.0.6	240G
Slave07	i9–9562	174.198.0.7	240G
Slave08	i9–9562	174.198.0.8	240G
Slave09	i9–9562	174.198.0.9	240G
Slave10	i9–9562	174.198.1.0	240G

Configure a wireless sensor multimedia big data parallel computing environment for each node. The specific steps are as follows:

(1) First, install the jdk software on each node, unzip the jdk file and use gedit to configure the environment variables.
(2) Configure the hostname of each node, and use the gedit/etc./sysconfig/network command to enter the edit mode to modify the hostname.
(3) Configure the IP address of each node. Use the setup command to pop up a tool window and select the "Network configuration" option. Fill in the new IP address. After the configuration is completed, use the/sbin/service network restart command to restart the network service.
(4) Close the firewall of each node, and use the setup command to pop up a tool window and select "Firewalls configuration" to enter the setting window for modification.

(5) Configure the hosts file. Use the gedit/etc./hosts command in each node to pop up the editing window, and then fill in the IP address and host information of each node in the file.
(6) Each node is configured to log in without password authentication, so that each node can access each other without a password.
(7) Configure the Hadoop environment on the master node, unzip the Hadoop file, and modify the Hadoop-env.sh, yarn-env.sh, core-site.xml, hdfs-site.xml, and yarn-site in the folder Xml and mapred-site.xml file contents.
(8) Copy the Hadoop configured on the master node to each slave node.
(9) Format the file system on the primary node, and use the start-all.sh command to start the Hadoop cluster.
(10) Use the jps command to view the processes running on each node. The master node runs four processes: ResourceManager, NameNode, Secondary NameNode, and Jps. The slave node runs three processes: Node Manager, DataNode, and Jps. This indicates that the processes of both the master node and the slave node are successfully started.

3.2 Experimental Results and Analysis

The two traditional methods of computational task scheduling are used as comparison method 1 and comparison method 2. The WordCount job was run using the design method, with the number of jobs being 10, 30, 50, 70, 100, and 150, and each job size was 512 MB. The average of 5 times and then 5 times is regarded as the time needed to complete the job, and the experimental results are shown in Fig. 2.

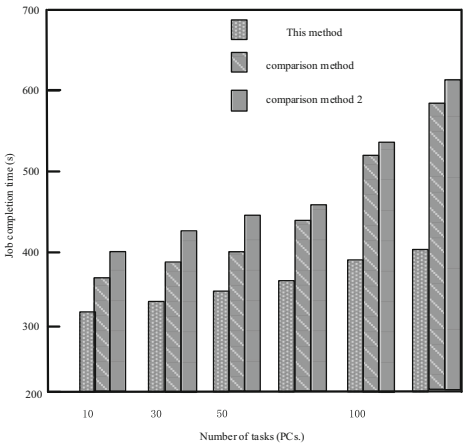

Fig. 2. Time required to complete the job

According to the results of Fig. 2, compared to the other two methods, the operation time of the design method is shorter, which proves that the design method has a good performance of parallel computing task scheduling.

The overall resource utilization rate in the task scheduling is tested. When the number of tasks increases from 1000 to 7000, the overall resource utilization test results of the design method are shown in Fig. 3.

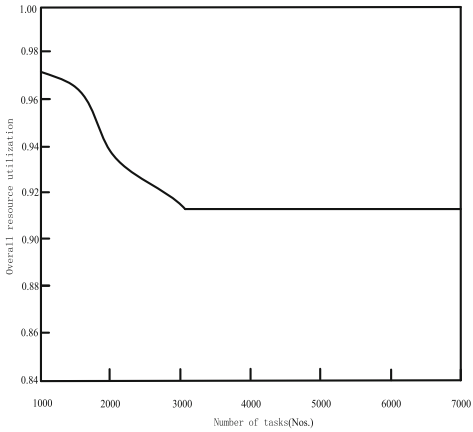

Fig. 3. Overall Resource Utilization Test Results

The test results in Fig. 3 show that the overall resource utilization of the design method reaches a stable value decreasing when the number of tasks increases from 1000 to 7000, while the overall resource utilization is above 0.91.

4 Conclusion

In the study of the parallel computing task scheduling method of wireless sensing multimedia big data, the artificial swarm algorithm is applied to realize the rapid scheduling of parallel computing tasks, which is of great significance for the development of wireless sensing network.

References

1. Zhao, N.: Cloud computing platform non-recurring task parallel scheduling simulation. Comput. Simul. **38**(1), 5 (2021)
2. Qi, Q., Zhang, L., Wang, J., et al.: Scalable parallel task scheduling for autonomous driving using multi-task deep reinforcement learning. IEEE Trans. Veh. Technol. **69**(11), 13861–13874 (2020)
3. Saleem, U., Liu, Y., Jangsher, S., et al.: Mobility-aware joint task scheduling and resource allocation for cooperative mobile edge computing. IEEE Trans. Wireless Commun. (99), 1 (2020)
4. Zhou, C., Wu, W., He, H., et al.: Deep reinforcement learning for delay-oriented IOT task scheduling in space-air-ground integrated network. IEEE Trans. Wirel. Commun. (99) 1 (2020)
5. Liu, S., He, T., Dai, J.: A survey of CRF algorithm based knowledge extraction of elementary mathematics in Chinese. Mob. Netw. Appl. (2021)

6. Nie, L., Wang, X., Sun, W., et al.: Imitation-learning-enabled vehicular edge computing: toward online task scheduling. IEEE Netw. **35**(3), 102–108 (2021)
7. Al-Habob, A.A., Dobre, O.A., Armada, A.G., Muhaidat, S., et al.: Task scheduling for mobile edge computing using genetic algorithm and conflict graphs. IEEE Trans. Veh. Technol. (99), 1 (2020)
8. Huang, X., Yu, R., Ye, D., et al.: Efficient workload allocation and user-centric utility maximization for task scheduling in collaborative vehicular edge computing. IEEE Trans. Veh. Technol. (99), 1 (2021)