

# **TASK ASSIGNMENT METHOD, APPARATUS, ELECTRONIC DEVICE, AND COMPUTER-READABLE STORAGE MEDIUM**

**[0001]** The present disclosure claims priority to the Chinese patent application with filing No. 2024103742384, titled "TASK ASSIGNMENT METHOD, APPARATUS, ELECTRONIC DEVICE, AND COMPUTER-READABLE STORAGE MEDIUM", filed on March 29, 2024, the entire contents of which are incorporated herein by reference as part of the present disclosure.

## **TECHNICAL FIELD**

**[0002]** The present disclosure relates to the technical field of data processing, and specifically, relates to a task assignment method, an apparatus, an electronic device, and a computer-readable storage medium.

## **BACKGROUND ART**

**[0003]** Based on the development of the Internet, various services also transfer from offline implementation to online implementation. For example, in some lending industries, repayment reminders for loans of loan users are usually implemented through automatic reminders by means of short messages and in-application output prompt messages.

**[0004]** However, the above implementation may have the situation of poor reminding effect. Based on this, the current processing manner is to assign various tasks to related executors, and the executors perform subsequent follow-ups on various tasks. In the current task assignment, the assignment is usually implemented based on the number of tasks, but the resource values corresponding to different tasks are different, and some tasks can achieve a balanced assignment effect based on the number. However, if the resource value difference corresponding to individual tasks is large, an assignment based on the number may lead to a large deviation in the assignment result.

## **SUMMARY**

**[0005]** The objective of the present disclosure is to provide a task assignment method, an apparatus, an electronic device, and a computer-readable storage medium, which can enable the task assignment to be more uniform.

**[0006]** In a first aspect, the present disclosure provides a task assignment method, including: partitioning a to-be-assigned task set according to resource values carried by tasks to obtain a plurality of task intervals, wherein each of the task intervals includes one or more tasks; assigning, for a target task interval, tasks in the target task interval to each of task executors according to a required execution proportion of each of the task executors, wherein the target task interval is any one of the plurality of task intervals; and aggregating assignment results of each of the task executors for each of the task intervals to obtain an assignment result of each of the task executors; and sending the assignment result corresponding to each of the task executors to a user account of the task executor.

**[0007]** In the above embodiment, by first aggregating tasks to obtain a plurality of task intervals, and then assigning tasks based on the task intervals, task assignment can not only meet

execution proportions of the task executors, but also enable resource values of tasks acquired by each of the task executors to be more balanced. Further, by dividing into a plurality of task intervals and assigning tasks within the intervals, both task quantity and task resource value acquired by the task executors can be better balanced.

**[0008]** In an optional embodiment, the step of assigning tasks in the target task interval to each of task executors according to a required execution proportion of each of the task executors includes: sorting the task executors in a descending order according to their required execution proportions to obtain an initial user queue; and sequentially acquiring task executors from the initial user queue according to an arrangement order of the task executors in the initial user queue, and assigning the tasks in the target task interval to each of them according to their required execution proportion.

**[0009]** In the above embodiment, sorting in the descending order according to the execution proportion enables tasks with higher execution proportions to be preferentially assigned, which can preferentially satisfy the task executors with higher execution proportions.

**[0010]** In an optional embodiment, the step of sequentially acquiring task executors from the initial user queue according to an arrangement order of the task executors in the initial user queue, and assigning the tasks in the target task interval to each of them according to their required execution proportion includes:

**[0011]** sequentially acquiring task executors from a real-time user queue according to the arrangement order of the task executors in the real-time user queue, and assigning the tasks in the target task interval to each of them; deleting a target task executor from the real-time user queue to obtain an updated user queue when a proportion of assigned tasks of the target task executor in the target task interval reaches the required execution proportion, wherein the target task executor is any one of the task executors in the real-time user queue; repeating the above steps until the tasks in the target task interval are completely assigned, wherein when the tasks in the target task interval are assigned for a first time, the real-time user queue is the initial user queue; when the tasks in the target task interval are assigned for an N-th time, the real-time user queue is an updated user queue obtained from a latest update; and N is a positive integer greater than or equal to two.

**[0012]** In the above embodiment, the task executors to whom sufficient tasks have been assigned can be timely deleted from the real-time user queue, thereby enabling the assignment structure to better meet the execution proportions. Further, by deleting task executors to whom sufficient tasks have been assigned from the real-time user queue, task assignment can also be completed without changing execution logic, thereby reducing execution difficulty.

**[0013]** In an optional embodiment, the step of deleting a target task executor from the real-time user queue to obtain an updated user queue when a proportion of assigned tasks of the target task executor in the target task interval reaches the required execution proportion includes:

**[0014]** deleting a target task executor from the real-time user queue to obtain an updated user queue when a proportion of resource values of assigned tasks of the target task executor in the target task interval reaches the required execution proportion; or deleting a target task executor from the real-time user queue to obtain an updated user queue when a proportion of the number of assigned tasks of the target task executor in the target task interval reaches the required execution proportion.

**[0015]** In an optional embodiment, the step of sequentially acquiring task executors from the real-time user queue and assigning the tasks in the target task interval to them includes: sorting, for an  $i$ -th round of assignment, remaining tasks in the target task interval according to a first order to obtain a first ordered task interval, sequentially acquiring first target tasks according to the sorting of the first ordered task interval, and sequentially assigning the first target tasks to each task executor in the real-time user queue according to the order of the real-time user queue; and sorting, for an  $(i+1)$ -th round of assignment, remaining tasks in the target task interval according to a second order to obtain a second ordered task interval, sequentially acquiring second target tasks according to the sorting of the first ordered task interval, and sequentially assigning the second target tasks to each task executor in the real-time user queue according to the order of the real-time user queue, wherein the second order is an order opposite to the first order, and  $i$  is a positive integer less than a number of tasks in the target task interval.

**[0016]** In the above embodiment, tasks are assigned alternately in order and reverse order, and tasks with maximum resource values and the tasks with minimum resource values in the target task interval are acquired alternately for assignment. Therefore, the assigned tasks can satisfy the execution proportion and the resource values of the tasks obtained by each executor can be more balanced.

**[0017]** In an optional embodiment, the step of partitioning a to-be-assigned task set according to resource values carried by tasks to obtain a plurality of task intervals includes: determining a number of intervals according to a total number of tasks in the to-be-assigned task set; determining an interval step length according to the number of intervals and a distribution of resource values of each of the to-be-assigned tasks in the to-be-assigned task set; and dividing the to-be-assigned task set into intervals according to the interval step length to obtain the task intervals of the number of intervals.

**[0018]** In the above embodiment, task intervals are assigned as needed based on the interval step length and the number of intervals, which can better satisfy the assignment requirement in terms of quantity and assignment of task intervals.

**[0019]** In an optional embodiment, the step of determining an interval step length according to a distribution of resource values of each of the to-be-assigned tasks in the to-be-assigned task set includes: determining a first quantile and a second quantile according to the distribution of resource values of each of the to-be-assigned tasks in the to-be-assigned task set; and determining the interval step length according to the first quantile, the second quantile, and the number of intervals.

**[0020]** In the above embodiment, the interval step length can be determined based on the two quantiles, which enables the interval step length to better match the assignment requirements of the to-be-assigned task set. The tasks, assigned to the partitions determined based on the interval step length, can better satisfy the balance for resource values.

**[0021]** In an optional embodiment, the step of dividing the to-be-assigned task set into intervals according to the interval step length to obtain the task intervals of the number of intervals includes: determining, for an  $m$ -th task interval, two endpoints of the  $m$ -th task interval based on the interval step length and a minimum resource value among the resource values of the to-be-assigned tasks in the to-be-assigned task set.

**[0022]** In an optional embodiment, the method further includes: saving the plurality of task intervals to obtain a backup to-be-assigned task set, wherein the backup to-be-assigned task set is used as backup data for assigning tasks to each of the task executors.

**[0023]** In the above embodiment, the plurality of task intervals can also be backed up, and in case of assignment exception, task assignment can continue based on the backup to-be-assigned task set, which can reduce the time consumption required for repeated processing and improve task assignment efficiency.

**[0024]** In a second aspect, the present disclosure provides a task assignment apparatus, including: a partitioning module, configured for partitioning a to-be-assigned task set according to resource values carried by tasks to obtain a plurality of task intervals, wherein each of the task intervals includes one or more tasks; an assignment module, configured for assigning, for a target task interval, tasks in the target task interval to each of task executors according to a required execution proportion of each of the task executors, wherein the target task interval is any one of the plurality of task intervals; an aggregation module, configured for aggregating assignment results of each of the task executors for each of the task intervals to obtain an assignment result of each of the task executors; and a sending module, configured for sending the assignment result corresponding to each of the task executors to a user account of the task executor.

**[0025]** In a third aspect, an electronic device is provided in the present disclosure, which includes a processor and a memory. The memory stores machine-readable instructions that are executed by the processor. When the electronic device is in operation, the machine-readable instructions, when run by the processor, execute the steps of the method as described in any of the above embodiments.

**[0026]** In a fourth aspect, the present disclosure provides a computer-readable storage medium, wherein the computer-readable storage medium stores a computer program thereon. The computer program, when run by the processor, executes the steps of the method as described in any of the above embodiments.

## **BRIEF DESCRIPTION OF DRAWINGS**

**[0027]** To more clearly illustrate the technical solutions of the embodiments of the present disclosure, the following will briefly introduce the drawings used in the embodiments. It should be understood that the following drawings only show some embodiments of the present disclosure, and therefore they should not be regarded as a limitation on the scope. Those ordinary skilled in the art can also obtain other related drawings based on these drawings without inventive effort.

**[0028]** FIG. 1 is a schematic diagram of a running environment of a task assignment method provided by an embodiment of the present disclosure;

**[0029]** FIG. 2 is a schematic block diagram of an electronic device provided by an embodiment of the present disclosure;

**[0030]** FIG. 3 is a flowchart of a task assignment method provided by an embodiment of the present disclosure; and

**[0031]** FIG. 4 is a schematic diagram of functional modules of a task assignment apparatus provided by an embodiment of the present disclosure.

## **DETAILED DESCRIPTION OF EMBODIMENTS**

**[0032]** The technical solutions in the embodiments of the present disclosure will be described below with reference to the drawings in the embodiments of the present disclosure.

**[0033]** It should be noted that similar numerals and letters denote similar terms in the following drawings so that once an item is defined in one drawing, it does not need to be further discussed in subsequent drawings. Meanwhile, in the description of the present application, the terms "first", "second" and the like are only used for distinguishing descriptions, and cannot be understood as indicating or implying relative importance.

**[0034]** In the current consumer credit industry, the collection scheme of consumer credit mainly adopts manual cooperation with the system to perform interaction for implementing task assignment, but the accuracy and efficiency of the assignment results obtained in this manner are relatively insufficient. This kind of collection task delegation scheme has many problems, for example: low collection efficiency, unbalanced task delegation, and so on. These problems lead to many loopholes and defects appearing in the current consumer credit collection chain.

**[0035]** Based on the above situation, the present disclosure provides a task assignment method, an apparatus, an electronic device, and a computer-readable storage medium, so that the task assigned based on this method can better satisfy the requirements of proportion and resource balance.

**[0036]** For the convenience of understanding the embodiment, the running environment and the electronic device for executing the task assignment method disclosed by the embodiment of the present disclosure are described in detail first.

**[0037]** As shown in FIG. 1, FIG. 1 shows a schematic diagram of an application environment of a task assignment method provided in an embodiment of the present disclosure. The application environment of the task assignment method can include a server 110 and a user terminal 120. The server 110 is in communication connection with one or more user terminals 120 through a network, to perform data communication or interaction. The server 110 can be a network server 110, a database server 110, and the like. The user terminal 120 can be a personal computer (PC), a tablet computer, a smartphone, a personal digital assistant (PDA), and the like.

**[0038]** A server end can run on the server 110, and functional modules included in the server end can be configured to execute steps in the task assignment method, so as to implement task assignment. Optionally, the server 110 can also include an operation interface, and the operation interface can be configured to receive restriction parameters in the task assignment method to be input. For example, the restriction parameters can include task executors, execution proportions required by the task executors, to-be-assigned task sets, acquisition approaches of the to-be-assigned task sets, and so on.

**[0039]** A client end can be installed in the user terminal 120, and the client end can access the server end in the server 110 to obtain the task assignment result implemented by the server end.

**[0040]** As shown in FIG. 2, it is a schematic block diagram of the electronic device. The electronic device 200 can include a memory 211 and a processor 213. It can be understood by those skilled in the art that the structure shown in FIG. 2 is merely schematic and does not impose limitations on the structure of the electronic device 200. For example, the electronic device 200 can also include more or fewer components than those shown in FIG. 2, or can have

a configuration different from that shown in FIG. 2. The server 110 and the user terminal 120 shown in FIG. 1 can include the components of the electronic device 200 shown in FIG. 2. Of course, based on different functions, the server 110 and the user terminal 120 can further include more components than those of the electronic device 200 shown in FIG. 2. For example, the server 110 can further include components such as a positioning system and a communication unit, and the user terminal 120 can further include components such as a display unit and a positioning system.

**[0041]** The above memory 211 and processor 213 are electrically connected with each other directly or indirectly, so as to implement data transmission or interaction. For example, these components can be electrically connected to each other through one or more communication buses or signal lines. The processor 213 is configured to execute the executable modules stored in the memory.

**[0042]** The memory 211 includes, but is not limited to, a random access memory (RAM), a read-only memory (ROM), a programmable read-only memory (PROM), an erasable programmable read-only memory (EPROM), an electric erasable programmable read-only memory (EEPROM), etc. The memory 211 is configured to store a program, and the processor 213 executes the program after receiving an execution instruction. The task assignment method executed by the electronic device 200 defined by the process disclosed in any embodiment of the present disclosure can be applied to the processor 213 or implemented by the processor 213.

**[0043]** The processor 213 can be an integrated circuit chip with signal processing capability. The processor 213 mentioned above can be a general-purpose processor, comprising a central processing unit (CPU), a network processor (NP), etc. It can also be a digital signal processor (DSP), application specific integrated circuit (ASIC), field programmable gate array (FPGA); or other programmable logic devices, discrete gates, transistor logic devices, discrete hardware components, etc. It can implement or perform each of the methods, steps, and logical block diagrams disclosed in the embodiment of the present disclosure. The general-purpose processor can be a microprocessor, or the processor can also be any conventional processor, and so on.

**[0044]** In the embodiment, the electronic device 200 can be configured to execute each step in each method provided in the embodiments of the present disclosure. Several embodiments are described below in detail to illustrate the implementation process of the task assignment method.

**[0045]** Referring to FIG. 3, FIG. 3 is a flowchart of a task assignment method provided by an embodiment of the present disclosure. The task assignment method in the embodiment can be applied in a server, and the steps of the task assignment method are executed by the server when executing its included functional modules. The specific flow shown in FIG. 3 is described in detail below.

**[0046]** Step 310: partitioning a to-be-assigned task set according to resource values carried by tasks to obtain a plurality of task intervals,

**[0047]** wherein each task interval includes one or more tasks.

**[0048]** Optionally, the to-be-assigned task set can be obtained from a task pool. The task pool can be a dynamically updated storage location, where a task is deleted from the task pool after being successfully assigned. When the server receives a new task, the new task can be stored in the task pool.

**[0049]** The resource values of the tasks in each task interval can fall within the value range corresponding to its task interval. As an example, the resource value can be a value that characterizes task attributes, for example, the value can be a score or an amount. In one example, the to-be-assigned task set can be collection cases, and the resource value can represent the amount corresponding to the collection case. In another example, the task can be a level-passing task in a game, and the resource value can represent a score corresponding to the level-passing task.

**[0050]** Optionally, the plurality of task intervals can be stored to obtain a backup to-be-assigned task set. As an example, the plurality of task intervals can be stored in a specified temporary storage location.

**[0051]** As an example, when an exception or error occurs in the subsequent task assignment process, the backup to-be-assigned task set can be obtained, and the task assignment can be continued based on the backup to-be-assigned task set. By backing up the plurality of task intervals, even if reassignment is required due to an error in subsequent distribution, task assignment can still be directly performed based on the backup to-be-assigned task set. This can reduce the consumption for computing resources and improve the efficiency of task assignments.

**[0052]** Step 320: assigning, for a target task interval, tasks in the target task interval to each of the task executors according to a required execution proportion of each of the task executors,

**[0053]** wherein the target task interval is any one of the plurality of task intervals.

**[0054]** The tasks in each task interval can be assigned according to the assignment method of step 320.

**[0055]** Optionally, the required execution proportion of each task executor can be the same or different. The required execution proportion of each task executor can be a preset default proportion, or an adaptively set proportion each time a task assignment is performed.

**[0056]** Exemplarily, if execution proportions required by each task executor are the same, then tasks in each task interval can be evenly assigned to each task executor.

**[0057]** Exemplarily, if execution proportions required by each task executor are different, then tasks in each task interval are assigned to each task executor according to the required execution proportion.

**[0058]** In one embodiment, the to-be-assigned task set needs to be assigned to four task executors, respectively task executor A, task executor B, task executor C, and task executor D; the execution proportion of task executor A is 10%, the execution proportion of task executor B is 20%, the execution proportion of task executor C is 30%, and the execution proportion of task executor D is 40%. Then tasks of the target task interval can be assigned to task executor A at 10% of the tasks, task executor B at 20% of the tasks, task executor C at 30% of the tasks, and task executor D at 40% of the tasks. Similarly, other task intervals are also assigned according to the assignment method, to obtain assignment results of each task interval.

**[0059]** Certainly, in actual use, in task assignment more task executors can be included.

**[0060]** Step 330: aggregating assignment results of each of the task executors for each of the task intervals to obtain an assignment result of each of the task executors.

**[0061]** By aggregating the assignment intervals of each task interval, the assignment result of the to-be-assigned task set can be obtained.

**[0062]** Taking any task executor as an example, the obtained assignment result is the sum of tasks obtained in each task interval.

**[0063]** Taking the above embodiment as an example, based on the assignment method of step 320 and step 330, tasks of the to-be-assigned task set can be assigned according to the method, in which task executor A is assigned 10% of tasks, task executor B is assigned 20% of tasks, task executor C is assigned 30% of tasks, and task executor D is assigned 40% of tasks. The assignment proportion of each task executor in each task interval is also the required execution proportion.

**[0064]** Step 340: sending the assignment result corresponding to each of the task executors to a user account of the task executor.

**[0065]** Exemplarily, the user account can be an account for logging into the server to obtain services provided by the server, each task executor can use the user account to log into a client running on the user terminal held by the task executor, and the assignment result can be obtained via the client.

**[0066]** Exemplarily, the user account can also be an instant messaging account, the assignment result corresponding to each task executor can be sent to the user account of the task executor, and the assignment result can be obtained by logging into the instant messaging tool via the user account.

**[0067]** Exemplarily, the user account can also be a phone number, and the assignment result can be sent to the user account of the task executor via methods such as short messages or phone calls.

**[0068]** In the above embodiment, it can be realized that the task assignment can better meet the requirement of the execution proportion required by each task executor. In addition, by assigning tasks to each task executor in each task interval, the resource value of tasks assigned to each task executor can be more balanced.

**[0069]** Considering that execution proportions of different task executors are different, the task executor with higher execution proportion relatively needs to be assigned with more tasks, if tasks are assigned in a polling manner, it will occur that task executors with higher required execution proportion are assigned later. In order to better guarantee the execution proportion of task executors with a high proportion, tasks can be preferentially assigned to task executors with a higher execution proportion. Based on this, the above step 320 can include step 321 and step 322.

**[0070]** Step 321: sorting the task executors in a descending order according to their required execution proportions to obtain an initial user queue.

**[0071]** Taking the above embodiment as an example, the order can be arranged as task executor D, task executor C, task executor B, and task executor A.

**[0072]** Step 322: sequentially acquiring task executors from the initial user queue according to an arrangement order of the task executors in the initial user queue, and assigning the tasks in the target task interval to each of them according to their required execution proportion.

**[0073]** Optionally, task executors from the initial user queue are sequentially acquired according to an arrangement order of the task executors in the initial user queue. Then, the proportion of tasks required to be executed by the task executor is obtained from the target task interval, and the proportion of tasks is assigned to the task executor until all task executors in the initial user queue are assigned.

**[0074]** Taking the above embodiment as an example, task executors D, task executors C, task executors B, and task executors A are sequentially obtained from the initial user queue; 40% of tasks, 30% of tasks, 20% of tasks, and 10% of tasks are obtained from the target task interval, then assigned to task executor D, task executor C, task executor B, and task executor A sequentially.

**[0075]** Optionally, task executors are sequentially acquired from a real-time user queue according to the arrangement order of the task executors in the real-time user queue, and tasks in the target task interval are assigned to the task executors until the tasks in the target task interval are completely assigned.

**[0076]** Exemplarily, one task is obtained from the target task interval each time, then sequentially assigned to each task executor according to the arrangement order of the real-time user queue.

**[0077]** The method of obtaining one task from the target task interval each time can be random, or can be obtained according to a predetermined acquisition rule. The acquisition rule can be in descending order of resource value; or in ascending order of resource value; or alternately obtaining the task with the minimum resource value and the task with the minimum resource value among the remaining tasks in the target task interval.

**[0078]** For any target task executor, if in the process of assigning tasks in the target task interval, the proportion of tasks in the target task interval, assigned to the target task executor, reaches the required execution proportion, then it is removed from the real-time user queue to obtain the updated user queue.

**[0079]** Taking the above embodiment as an example, one task is obtained from the target task interval each time, then sequentially assigned to task executor D, task executor C, task executor B, and task executor A. After 10% of tasks have been assigned to task executor A, no more tasks in the target task interval are assigned to the task executor A, and at this time, the updated user queue includes only task executor D, task executor C, and task executor B. Then one task continues to be obtained from the target task interval and is sequentially assigned to task executor D, task executor C, and task executor B, and so on, until all tasks in the target task interval are assigned.

**[0080]** In one embodiment, the method for removing the task executor from the real-time user queue is to remove the task executor from the real-time user queue when the number of tasks executed reaches the required execution proportion. That is, if a proportion of resource values of tasks assigned to the target task executor in the target task interval reaches the required execution proportion, the target task executor is deleted from the real-time user queue to obtain an updated user queue.

**[0081]** Taking the above embodiment as an example, one task is obtained from the target task interval each time, then sequentially assigned to task executor D, task executor C, task executor B, and task executor A. After the number of tasks assigned to task executor A reaches 10% of the number of tasks in the target task interval, no more tasks in the target task interval are assigned to the task executor, and at this time, the updated user queue includes only task executor D, task executor C, and task executor B. Then one task continues to be obtained from the target task interval, and is sequentially assigned to task executor D, task executor C, and task executor B, and so on, until all tasks in the target task interval are assigned.

**[0082]** In another embodiment, the method for removing the task executor from the real-time user queue is to remove the task executor from the real-time user queue when the total of the resource values of the tasks executed reaches the required execution proportion. That is, if the proportion of the number of tasks assigned to the target task executor in the target task interval reaches the required execution proportion, the target task executor is deleted from the real-time user queue to obtain an updated user queue.

**[0083]** Taking the above embodiment as an example, one task is obtained from the target task interval each time, then sequentially assigned to task executor D, task executor C, task executor B, and task executor A. After a total of resource values of tasks assigned to the task executor A reaches 10% of the total resource values of tasks in the target task interval, no more tasks in the target task interval are assigned to the task executor, and at this time, the updated user queue includes only task executor D, task executor C, and task executor B. Then one task continues to be obtained from the target task interval, and is sequentially assigned to task executor D, task executor C, and task executor B, and so on, until all tasks in the target task interval are assigned.

**[0084]** In consideration that although the resource values of tasks in each task interval are relatively concentrated in value, there still exist differences in magnitude, in order to make the assigned resource values to task executors more balanced, the above step 322 can include step 3221 and step 3222.

**[0085]** Step 3221: sorting, for an  $i$ -th round of assignment, remaining tasks in the target task interval according to a first order to obtain a first ordered task interval, sequentially acquiring first target tasks according to the sorting of the first ordered task interval, and sequentially assigning the first target tasks to each task executor in the real-time user queue according to the order of the real-time user queue.

**[0086]** Exemplarily, the first order can be in descending order.

**[0087]** Step 3222: sorting, for an  $(i+1)$ -th round of assignment, remaining tasks in the target task interval according to a second order to obtain a second ordered task interval, sequentially acquiring second target tasks according to the sorting of the second ordered task interval, and sequentially assigning the second target tasks to each task executor in the real-time user queue according to the order of the real-time user queue,

**[0088]** wherein the second order is an order opposite to the first order, and  $i$  is a positive integer less than a number of tasks in the target task interval.

**[0089]** Exemplarily, the second order can be in ascending order.

**[0090]** In consideration that although the resource values of tasks in each task interval are relatively concentrated in value, there still exist differences in magnitude, in order to make the assigned resource values to task executors more balanced, the above step 322 can include: sorting tasks in the target task interval in an ascending order according to the magnitude of resource value carried by each task, to obtain a task queue; for the  $i$ -th round of assignment, a task is obtained from the head of the task queue, when  $i$  is not greater than the number of task executors in the real-time user queue, the task is assigned to the  $i$ -th task executor in the real-time user queue; when  $i$  is greater than the number of task executors in the real-time user queue, the task is assigned to the  $x$ -th task executor in the real-time user queue;

**[0091]** for the  $(i+1)$ -th round of assignment, a task is obtained from the tail of the task queue, when  $i+1$  is not greater than the number of task executors in the real-time user queue, the task

is assigned to the (i+1)-th task executor in the real-time user queue; and when i+1 is greater than the number of task executors in the real-time user queue, the task is assigned to the y-th task executor in the real-time user queue, wherein y is the remainder of (i+1) divided by the number of task executors in the real-time user queue.

**[0092]** Through the above task assignment method, by alternately obtaining the task with the maximum resource value and the task with the minimum resource value in the target task interval for assignment, the task assignment result can be more balanced, and the required execution proportion of the task executors can be better satisfied.

**[0093]** The basis of balanced task assignment in the method provided by the embodiment is to partition the to-be-assigned task set according to the resource value. The above step 310 can include step 311 to step 313.

**[0094]** Step 311: determining a number of intervals according to a total number of tasks in the to-be-assigned task set.

**[0095]** Optionally, the number of intervals of the to-be-assigned task set can be determined using the Sturges formula. For example, the number of intervals =  $1 + \log_2(n)$ , where n indicates the total number of tasks in the to-be-assigned task set.

**[0096]** Step 312: determining an interval step length according to the number of intervals and a distribution of resource values of each of the to-be-assigned tasks in the to-be-assigned task set.

**[0097]** Optionally, the first quantile and the second quantile can be determined according to the distribution of the resource values of the to-be-assigned task in the to-be-assigned task set; then, the interval step length is determined according to the first quantile, the second quantile, and the number of intervals.

**[0098]** The interval step length can be calculated using the following formula:

**[0099]**  $St = (P1 - P2) / N$ ,

**[00100]** where St indicates the interval step length; P1 indicates the first quantile; P2 indicates the second quantile; and N indicates the number of intervals.

**[00101]** Exemplarily, the first quantile can be a relatively large quantile such as the 75% quantile, the 80% quantile, the 95% quantile, etc. The 75% quantile indicates a value greater than the resource values of 75% of tasks in the to-be-assigned task set.

**[00102]** Exemplarily, the second quantile can be a relatively small quantile such as the 1% quantile, the 5% quantile, the 10% quantile, etc.

**[00103]** Step 313: dividing the to-be-assigned task set into intervals according to the interval step length to obtain the task intervals of the number of intervals.

**[00104]** Exemplarily, for an m-th task interval, two endpoints of the m-th task interval can be determined based on the interval step length and a minimum resource value among the resource values of the to-be-assigned tasks in the to-be-assigned task set.

**[00105]** Optionally, the boundaries of each value interval can be determined according to the determined interval step length and the minimum resource value among the resource values of the to-be-assigned task, for example, determined using the following formula:

**[00106]**  $Bli = Vmin + m * St$ , and

**[00107]**  $Bui = Vmin + (m+1) * St-1$ ,

**[00108]** where  $B_{li}$  indicates the lower boundary of the  $i$ -th value interval;  $B_{ui}$  indicates the upper boundary of the  $i$ -th value interval;  $S_t$  indicates the interval step length; and  $V_{min}$  indicates the minimum resource value among the resource values of the to-be-assigned task.

**[00109]** Optionally, the boundaries of the value interval can be determined as integers, and the boundaries obtained through the above calculation method can be rounded to obtain integer values.

**[00110]** By dividing the to-be-assigned task set using the above interval division method, the resource values of the tasks in each interval can be relatively more concentrated, so that the subsequent task assignment can also be more balanced.

**[00111]** The task assignment provided in the embodiment of the present disclosure can realize that under the condition where the resource values of tasks are highly scattered, task assignment according to a predetermined proportion can still be ensured, and the assigned resource values of tasks to each task executor can also be more balanced. Further, partitioning is performed by resource value intervals, and even when the number of tasks in the value interval is small, the reliability of task assignment proportion can still be ensured. Through the above assignment method, while assigning according to the proportion of resource values of tasks, the number of assigned tasks can also better conform to the proportional configuration.

**[00112]** Based on the same inventive concept of the present disclosure, the embodiment of the present disclosure further provides a task assignment apparatus corresponding to the task assignment method. Since the principle of solving the problem by the apparatus in the embodiment of the present disclosure is similar to that of the foregoing embodiment of the task assignment method, the implementation of the apparatus in the present embodiment can refer to the description in the above-mentioned embodiment of the method, and repeated parts will not be described again.

**[00113]** Referring to FIG. 4, FIG. 4 is a schematic diagram of functional modules of a task assignment apparatus provided by an embodiment of the present disclosure. Each module in the task assignment apparatus of the present embodiment is configured for executing each step in the above-mentioned embodiment of the method. The task assignment apparatus includes: a partitioning module 410, an assignment module 420, an aggregation module 430, and a sending module 440, wherein the contents of each module are as follows.

**[00114]** The partitioning module 410 is configured for partitioning a to-be-assigned task set according to resource values carried by tasks to obtain a plurality of task intervals, wherein each of the task intervals includes one or more tasks.

**[00115]** The assignment module 420 is configured for assigning, for a target task interval, tasks in the target task interval to each of the task executors according to a required execution proportion of each of the task executors, wherein the target task interval is any one of the plurality of task intervals.

**[00116]** The aggregation module 430 is configured for aggregating assignment results of each of the task executors for each of the task intervals to obtain an assignment result of each of the task executors.

**[00117]** The sending module 440 is configured for sending the assignment result corresponding to each of the task executors to a user account of the task executor.

**[00118]** In one possible embodiment, the above-mentioned assignment module 420 includes: a personnel sorting unit and a task assigning unit.

**[00119]** The personnel sorting unit is configured for sorting the task executors in a descending order according to their required execution proportions to obtain an initial user queue. The task assigning unit is configured for sequentially acquiring task executors from the initial user queue according to an arrangement order of the task executors in the initial user queue, and assigning the tasks in the target task interval to each of them according to their required execution proportion.

**[00120]** In one possible embodiment, the task assigning unit is configured for sequentially acquiring task executors from a real-time user queue according to the arrangement order of the task executors in the real-time user queue, and assigning the tasks in the target task interval to each of them.

**[00121]** The task assignment apparatus can further include a queue updating unit, configured for deleting a target task executor from the real-time user queue to obtain an updated user queue when a proportion of assigned tasks of the target task executor in the target task interval reaches the required execution proportion, wherein the target task executor is any one of the task executors in the real-time user queue.

**[00122]** The above task assigning unit and queue updating unit are executed repeatedly until the tasks in the target task interval are completely assigned, wherein when the tasks in the target task interval are assigned for the first time, the real-time user queue is the initial user queue; when the tasks in the target task interval are assigned for an N-th time, the real-time user queue is the updated user queue obtained from the latest update; and N is a positive integer greater than or equal to two.

**[00123]** In one possible embodiment, the queue updating unit is configured for deleting a target task executor from the real-time user queue to obtain an updated user queue when a proportion of resource values of assigned tasks of the target task executor in the target task interval reaches the required execution proportion.

**[00124]** The queue updating unit above is further configured for deleting a target task executor from the real-time user queue to obtain an updated user queue when a proportion of a number of assigned tasks of the target task executor in the target task interval reaches the required execution proportion.

**[00125]** In one possible embodiment, the task assigning unit is configured for sorting, for an i-th round of assignment, remaining tasks in the target task interval according to a first order to obtain a first ordered task interval, sequentially acquiring first target tasks according to the sorting of the first ordered task interval, and sequentially assigning the first target tasks to each task executor in the real-time user queue according to the order of the real-time user queue; and sorting, for an (i+1)-th round of assignment, remaining tasks in the target task interval according to a second order to obtain a second ordered task interval, sequentially acquiring second target tasks according to the sorting of the first ordered task interval, and sequentially assigning the second target tasks to each task executor in the real-time user queue according to the order of the real-time user queue, wherein the second order is an order opposite to the first order, and i is a positive integer less than a number of tasks in the target task interval.

**[00126]** In one possible embodiment, the above-mentioned partitioning module 410 includes: a quantity determining unit, a step length determining unit, and a region partitioning unit.

**[00127]** The quantity determining unit is configured for determining a number of intervals according to a total number of tasks in the to-be-assigned task set.

**[00128]** The step length determining unit is configured for determining an interval step length according to the number of intervals and a distribution of resource values of each of the to-be-assigned tasks in the to-be-assigned task set.

**[00129]** The region partitioning unit is configured for dividing the to-be-assigned task set into intervals according to the interval step length to obtain the task intervals of the number of intervals.

**[00130]** In one possible embodiment, the step length determining unit is configured for determining a first quantile and a second quantile according to the distribution of resource values of each of the to-be-assigned tasks in the to-be-assigned task set; and determining the interval step length according to the first quantile, the second quantile, and the number of intervals.

**[00131]** In one possible embodiment, the region partitioning unit is configured for determining, for an m-th task interval, two endpoints of the m-th task interval based on the interval step length and a minimum resource value among the resource values of the to-be-assigned tasks in the to-be-assigned task set.

**[00132]** In one possible embodiment, the task assignment apparatus of the present embodiment can further include a backup module, configured for saving the plurality of task intervals to obtain a backup to-be-assigned task set, wherein the backup to-be-assigned task set is used as backup data for assigning tasks to each of the task executors.

**[00133]** In addition, the embodiments of the present disclosure also provide a computer-readable storage medium, wherein the computer-readable storage medium stores computer programs. When the computer program is executed by a processor, the steps of the task assignment method described in the above method embodiment are executed.

**[00134]** The computer program product of the task assignment method provided in the embodiment of the present disclosure includes a computer-readable storage medium storing program code, wherein the instructions included in the program code are used for executing the steps of the task assignment method described in the above method embodiments. For details, reference may be made to the above method embodiments, and repeated description is omitted herein.

**[00135]** In the several embodiments provided in the present disclosure, it should be understood that the devices and methods disclosed can be implemented in other ways. The device embodiments described above are merely illustrative. For example, the flowcharts and diagrams in the drawings illustrate possible implementations, architectures, functions, and operations of devices, methods, and computer program products according to multiple embodiments of the present disclosure. At this point, each box in the flowchart or diagram can represent a module, program segment, or part of the code. Each part of the module, program segment, or code includes one or more executable instructions for implementing the specified logical functions. It should also be noted that in some alternative implementations, the functions indicated in the boxes may occur in a different order than those indicated in the drawings. For example, two consecutive boxes can actually be executed in parallel, and sometimes they can also be executed in reverse order, depending on the functionality involved. It should also be noted that each box in the block diagram and/or flowchart, and the combination of boxes in the block diagram and/or flowchart, can be implemented by

specialized hardware systems designed to perform the specified functions or actions, or by a combination of specialized hardware and computer instructions.

**[00136]** Further, each functional module in each embodiment of the present disclosure can be integrated together to form a separate part, or each module may exist separately, or two or more modules may be integrated to form a separate part.

**[00137]** The functionality, when implemented as a software functional module and sold or used as a stand-alone product, can be stored in a computer-readable storage medium. Based on this understanding, the technical solution of the present disclosure can essentially be embodied in the form of a software product, which contributes to or includes parts in the prior art. The software product is stored in a storage medium and includes multiple instructions for causing a computer device (which can be a personal computer, server, network device, etc.) to execute all or some of the steps of the methods described in various embodiments of the present disclosure. The aforementioned storage media include various media that can store program code, such as USB drives, external hard drives, read-only memory (ROM), random access memory (RAM), disks, or optical discs. It is important to note that, in the context, relationship terms such as first and second are used only to distinguish one entity or operation from another entity or operation, without necessarily requiring or implying any such actual relationship or order between those entities or operations. Furthermore, the terms “comprise”, “include”, or any other variations are intended to encompass non-exclusive inclusion. This allows a process, method, item, or device that includes a series of elements to not only include those elements but also include other elements that are not explicitly listed, or elements that are inherent to the process, method, item, or device. In the absence of further limitations, the inclusion of an element specified by the phrase "comprising..." does not exclude the presence of additional identical elements in the process, method, article, or apparatus that includes the specified element.

**[00138]** The above is only a preferred embodiment of the present disclosure, which is not intended to limit, and the present disclosure may have various changes and variations for those skilled in the art. Any modification, equivalent substitution, improvement, etc. made within the spirit and principles of the present disclosure shall be included in the scope of protection of the present disclosure. It should be noted that similar numerals and letters denote similar terms in the following drawings so that once an item is defined in one drawing, it does not need to be further defined or discussed in subsequent drawings.

**[00139]** The above is only a specific embodiment of the present disclosure, but the scope of protection of the present disclosure is not limited thereto. Any person skilled in the art can easily envisage changes or substitutions within the technical scope disclosed in the present disclosure, which should be encompassed within the scope of protection of the present disclosure. Therefore, the scope of protection of the present disclosure shall be stated to be subject to the scope of protection of the claims.

## CLAIMS

1. A task assignment method, comprising:

partitioning a to-be-assigned task set according to resource values carried by tasks to obtain a plurality of task intervals, wherein each of the task intervals comprises one or more tasks;

assigning, for a target task interval, tasks in the target task interval to each of task executors according to a required execution proportion of each of the task executors, wherein the target task interval is any one of the plurality of task intervals;

aggregating assignment results of each of the task executors for each of the task intervals to obtain an assignment result of each of the task executors; and

sending the assignment result corresponding to each of the task executors to a user account of the task executor.
2. The method according to claim 1, wherein the step of assigning tasks in the target task interval to each of task executors according to a required execution proportion of each of the task executors comprises:

sorting the task executors in a descending order according to their required execution proportions to obtain an initial user queue; and

sequentially acquiring the task executors from the initial user queue according to an arrangement order of the task executors in the initial user queue, and assigning the tasks in the target task interval to each of them according to their required execution proportions.
3. The method according to claim 2, wherein the step of sequentially acquiring the task executors from the initial user queue according to an arrangement order of the task executors in the initial user queue, and assigning the tasks in the target task interval to each of them according to their required execution proportions comprises:

sequentially acquiring the task executors from a real-time user queue according to the arrangement order of the task executors in the real-time user queue, and assigning the tasks in the target task interval to each of them;

deleting a target task executor from the real-time user queue to obtain an updated user queue when a proportion of assigned tasks of the target task executor in the target task interval reaches the required execution proportion, wherein the target task executor is any one of the task executors in the real-time user queue;

repeating the above steps until the tasks in the target task interval are completely assigned, wherein when the tasks in the target task interval are assigned for a first time, the real-time user queue is the initial user queue; when the tasks in the target task interval are assigned for an N-th time, the real-time user queue is the updated user queue obtained from a latest update; and N is a positive integer greater than or equal to two.

4. The method according to claim 3, wherein the step of deleting a target task executor from the real-time user queue to obtain an updated user queue when a proportion of assigned tasks of the target task executor in the target task interval reaches the required execution proportion comprises:

deleting a target task executor from the real-time user queue to obtain the updated user queue when a proportion of resource values of the assigned tasks of the target task executor in the target task interval reaches the required execution proportion; or

deleting a target task executor from the real-time user queue to obtain the updated user queue when a proportion of the number of the assigned tasks of the target task executor in the target task interval reaches the required execution proportion.

5. The method according to claim 3, wherein the step of sequentially acquiring the task executors from the real-time user queue and assigning the tasks in the target task interval to each of them comprises:

sorting, for an i-th round of assignment, remaining tasks in the target task interval according to a first order to obtain a first ordered task interval, sequentially acquiring first target tasks according to a sorting of the first ordered task interval, and sequentially assigning the first target tasks to each task executor in the real-time user queue according to an order of the real-time user queue; and

sorting, for an (i+1)-th round of assignment, remaining tasks in the target task interval according to a second order to obtain a second ordered task interval, sequentially acquiring second target tasks according to the sorting of the first ordered task interval, and sequentially assigning the second target tasks to each task executor in the real-time user queue according to the order of the real-time user queue, wherein the second order is an order opposite to the first order, and i is a positive integer less than the number of the tasks in the target task interval.

6. The method according to claim 1, wherein the step of partitioning a to-be-assigned task set according to resource values carried by tasks to obtain a plurality of task intervals comprises:

determining the number of intervals according to a total number of tasks in the to-be-assigned task set;

determining an interval step length according to the number of intervals and a distribution of resource values of each of to-be-assigned tasks in the to-be-assigned task set; and

dividing the to-be-assigned task set into intervals according to the interval step length to obtain the task intervals of the number of intervals.

7. The method according to claim 6, wherein the step of determining an interval step length according to a distribution of resource values of each of to-be-assigned tasks in the to-be-assigned task set comprises:

determining a first quantile and a second quantile according to the distribution of the resource values of each of the to-be-assigned tasks in the to-be-assigned task set; and

determining the interval step length according to the first quantile, the second quantile, and the number of intervals.

8. The method according to claim 6, wherein the step of dividing the to-be-assigned task set into intervals according to the interval step length to obtain the task intervals of the number of intervals comprises:

determining, for an m-th task interval, two endpoints of the m-th task interval based on the interval step length and a minimum resource value among the resource values of the to-be-assigned tasks in the to-be-assigned task set.

9. The method according to any one of claims 1 to 8, wherein the method further comprises:

saving the plurality of task intervals to obtain a backup to-be-assigned task set, wherein the backup to-be-assigned task set is used as backup data for assigning tasks to each of the task executors.

10. A task assignment apparatus, comprising:

a partitioning module, configured for partitioning a to-be-assigned task set according to resource values carried by tasks to obtain a plurality of task intervals, wherein each of the task intervals comprises one or more tasks;

an assignment module, configured for assigning, for a target task interval, tasks in the target task interval to each of the task executors according to a required execution proportion of each of the task executors, wherein the target task interval is any one of the plurality of task intervals;

an aggregation module, configured for aggregating assignment results of each of the task executors for each of the task intervals to obtain an assignment result of each of the task executors; and

a sending module, configured for sending the assignment result corresponding to each of the task executors to a user account of the task executor.

11. An electronic device, comprising: a processor and a memory, wherein the memory stores machine-readable instructions that are executed by the processor; and when the electronic device is in operation, the machine-readable instructions, when run by the processor, execute the steps of the method according to any one of claims 1 to 9.
12. A computer-readable storage medium, wherein the computer-readable storage medium is provided with a computer program stored thereon, and the computer program executes the steps of the method according to any one of claims 1 to 9 when run by a processor.