

USER CODE RUNNING METHOD FOR PROGRAMMING PLATFORM, PLATFORM, NODE, DEVICE AND MEDIUM

CROSS-REFERENCE TO RELATED APPLICATION

[0001] This application is filed based upon and claims priority to Chinese Patent Application No. 202010093681.6 filed on February 14, 2020, the contents of which are hereby incorporated by reference in its entirety.

TECHNICAL FIELD

[0002] The present disclosure relates to the technical field of programming, and more particularly, to a user code running method for a programming platform, a platform, a node, a device and a medium.

BACKGROUND

[0003] With the rapid development of artificial intelligence, various systems that run by relying on codes have emerged. Not only the codes are becoming increasingly diverse, but also increasingly higher requirements are imposed on the functions that the systems can provide.

[0004] Faced with increasingly complex codes, in a present programming platform, the execution environment of the codes such as Kernel (execution environment) usually needs not only to maintain the connection with a user terminal but also to compute codes sent from the user terminal, so that the capability of the system cannot satisfy code running requirements.

[0005] In view of the above, how to improve the concurrency ability for user code

running of the programming platform becomes an urgent problem to be solved.

SUMMARY

[0006] The present disclosure provides a user code running method for a programming platform, a platform, a node, a device and a medium.

[0007] The programming platform of the present disclosure includes a service node, a task queue and a computing node. A first aspect of the present disclosure provides a user code running method for a programming platform, the method includes: acquiring, by the service node, a user code for representing a user terminal, and placing the user code as a running task in the task queue; monitoring, by the computing node, the task queue to acquire the running task; computing, by the computing node, the running task to obtain a computation result; and acquiring, by the service node, the computation result from the computing node, and feeding back the computation result to the user terminal.

[0008] Therefore, the service node is configured to implement the connection with the user terminal, place the user code in the queue, and the computing node is configured to compute the user code, thereby realizing the running of the user code. In the running process of the user code, the service node and the computing node are set separately, so that the connection with the user terminal and the computation of the user code are completed by the service node and the computing node respectively, asynchronous communication for the user code running is realized, and the concurrency capability of the system is improved.

[0009] According to some embodiments, the programming platform includes at least two computing nodes; and the operation of monitoring, by the computing node, the task queue to acquire the running task includes: monitoring, by the at least two computing nodes, the task queue, and requesting acquisition of the running task from the task queue; and assigning, by the task queue, one running task to one of the computing nodes.

[0010] Therefore, when a running task in the task queue is monitored by multiple computing nodes, the multiple computing nodes acquire the running task by requesting from the task queue, and the task queue assigns the running task to one computing node, so that the reasonable assignment of the running task can be completed by the task queue.

[0011] According to some embodiments, the task queue includes a first task queue and a second task queue, and at least two computing nodes include a first computing node and a second computing node; the operation of placing the user code as the running task in the task queue includes: placing, according to a computation characteristic of the user code, the user code as a running task in the first task queue or the second task queue; and the operation of monitoring, by the computing node, the task queue includes: monitoring, by the first computing node, the first task queue, and monitoring, by the second computing node, the second task queue.

[0012] Therefore, the service node may distinguish the user codes based on the computation characteristics of the user codes, and place the user code in the first task queue or the second task queue, respectively, so that different computing nodes monitor different task queues, thereby distinguishing the user codes according to different running requirements of the user code.

[0013] According to some embodiments, the programming platform includes at least two service nodes; and the operation of acquiring, by the service node, the user code for representing the user terminal includes: the user code being assigned to the at least two service nodes through a reverse proxy.

[0014] Therefore, the present disclosure can provide at least two service nodes, and the service nodes are assigned user code through the reverse proxy, so that multiple service nodes receive the user code sent from the user terminal, thereby improving the concurrent processing capability.

[0015] According to some embodiments, after the computation of the running task is

completed, the computing node releases the computing resources corresponding to the computation, and continues to monitor the task queue.

[0016] Therefore, after the computation is completed, the computing node releases resources, and continues to monitor the task queue to acquire a new running task. After the running task is performed, the computing node releases the resources, so that the computing and memory resources are reused, and hardware resources are fully and reasonably utilized.

[0017] According to some embodiments, the programming platform further includes a message queue; and the operation of acquiring, by the service node, the computation result from the computation node and feeding back the computation result to the user terminal includes: placing, by the computing node, the computation result in the message queue; and monitoring, by the service node, the message queue to acquire the computation result, and feeding back the computation result to the user terminal.

[0018] Therefore, after the running task corresponding to any user code is performed, the computing node uniformly places the computation result in the message queue, and the service node monitors the message queue to obtain the computation result from the computing node, so that the computation result is also uniformly managed.

[0019] According to some embodiments, the service node maintains a session connection with the user terminal before the computation result is acquired.

[0020] Therefore, the service node maintains the session connection with the user terminal until the computation result is acquired, that is, the connection between the service node and the user terminal is maintained, so that the computation result is conveniently feedback to the user terminal.

[0021] A second aspect of the present disclosure provides a user code running method for a programming platform, the method includes: acquiring a user code for representing a user terminal; placing the user code as a running task in a task queue to enable a

computing node to acquire the running task from the task queue and compute the running task; and acquiring a computation result from the computing node, and feeding back the computation result to the user terminal.

[0022] Therefore, the service node is configured to connect with the user terminal and place the user code sent from the user terminal as a running task in the task queue, so that after the computing node independent of the service node acquires the running task from the task queue and computes the running task to obtain the computation result, the service node acquires the computing result from the computing node, and feeds back the computation result to the user terminal to realize the running of the user code.

[0023] According to some embodiments, the operation of placing the user code as the running task in the task queue to enable to the computing node acquires and computes the running task in the task queue includes: placing, according to a computation characteristic of the user code, the user code as a running task in the first task queue or the second task queue to enable a first computing node to acquire and compute the running task from the first task queue, or to enable a second computing node to acquire and compute the running task from the second task queue.

[0024] Therefore, the service node may distinguish the user code based on the computation characteristic of the user code, and place the user code in the first task queue or the second task queue, respectively, so that different computing nodes monitor different task queues, thereby distinguishing the user codes according to different running requirements of the user code.

[0025] According to some embodiments, the operation of acquiring the user code for representing the user terminal includes: the user code being assigned through a reverse proxy.

[0026] Therefore, the user code is assigned to the service node through the reverse proxy, so that the received user code issued from the user terminal is reasonably assigned to the service node.

[0027] According to some embodiments, the method includes: maintaining a session connection with the user terminal before the computation result is acquired.

[0028] Therefore, the service node maintains the session connection with the user terminal until the computation result is acquired, that is, the connection is maintained between the service node and the user terminal, to conveniently feedback the computation result to the user terminal.

[0029] A third aspect of the present disclosure provides a user code running method for a programming platform, the method includes: monitoring a task queue to acquire a running task, where the task queue includes the running task corresponding to a user code; and computing the running task to obtain a computation result.

[0030] Therefore, the task queue includes the running task corresponding to the user code. Therefore, the computing node monitors the task queue including the running task, so as to acquire the running task corresponding to the user code, and then computes the running task to obtain the computation result, thereby realizing the acquisition and computation of the running task.

[0031] According to some embodiments, the user code running method for a programming platform further includes: after computation of the running task is completed, releasing computing resources corresponding to the computation, and continuing to monitor the task queue.

[0032] Therefore, after the computation is completed, the computing node releases the resources, and continues to monitor the task queue to acquire a new running task, so that the user codes can be executed in different computing nodes, and the computing node releases the resources after the running task is performed, so that the computing and memory resources are reused, and hardware resources are fully and reasonably utilized.

[0033] A fourth aspect of the present disclosure provides a programming platform

including a service node, a task queue and a computing node. The service node is configured to acquire a user code for representing a user terminal, and place the user code as a running task in the task queue; the computing node is configured to monitor the task queue to acquire the running task; the computing node is further configured to compute the running task to obtain a computation result; and the service node is further configured to acquire the computation result from the computing node, and feedback the computation result to the user terminal.

[0034] According to some embodiments, the programming platform includes at least two computing nodes; and when the computing node is configured to monitor the task queue to acquire the running task, at least two computing nodes are configured to monitor the task queue and request acquisition of the running task from the task queue, where the task queue is configured to assign one running task to one of the computing nodes.

[0035] According to some embodiments, the task queue includes a first task queue and a second task queue, and the at least two computing nodes include a first computing node and a second computing node; when the service node is configured to place the user code as the running task in the task queue, the service node is specifically configured to place, according to a computation characteristic of the user code, the user code as a running task in the first task queue or the second task queue; and when the computing node is configured to monitor the task queue, the first computing node is specifically configured to monitor the first task queue, and the second computing node is configured to monitor the second task queue.

[0036] According to some embodiments, the programming platform includes at least two service nodes; and when the service node is configured to acquire the user code for representing the user terminal, the user code is assigned to the at least two service nodes through a reverse proxy.

[0037] According to some embodiments, the computing node is configured to, after

completing computation of the running task, release computing resources corresponding to the computation and continue to monitor the task queue.

[0038] According to some embodiments, the programming platform further includes a message queue; and when the service node is configured to acquire the computation result from the computation node and feedback the computation result to the user terminal, the computing node is configured to place the computation result in the message queue; and the service node is configured to monitor the message queue to acquire the computation result, and feedback the computation result to the user terminal.

[0039] According to some embodiments, the service node is further configured to maintain a session connection with the user terminal before the computation result is acquired.

[0040] A fifth aspect of the present disclosure provides a service node including a receiving unit, a placing unit and an acquiring unit. The receiving unit is configured to acquire a user code for representing a user terminal; the placing unit is configured to place the user code as a running task in a task queue to enable a computing node to acquire and compute the running task from the task queue; and the acquiring unit is configured to acquire a computation result from the computing node and feedback the computation result to the user terminal.

[0041] According to some embodiments, when the placing unit is configured to place the user code as a running task in a task queue to enable a computing node to acquire and compute the running task from the task queue, the placing unit is specifically configured to place, according to a computation characteristic of the user code, the user code as a running task in the first task queue or the second task queue to enable a first computing node to acquire and compute the running task from the first task queue, or to enable a second computing node to acquire and compute the running task from the second task queue.

[0042] According to some embodiments, when the receiving unit is configured to

acquire the user code for representing the user terminal, the receiving unit is specifically configured to be assigned with the user code through a reverse proxy.

[0043] According to some embodiments, the acquiring unit is configured to maintain a session connection with the user terminal before acquiring the computation result.

[0044] A sixth aspect of the present disclosure provides a computing node including a monitoring unit and a computing unit. The monitoring unit is configured to monitor a task queue to acquire a running task, where the task queue includes the running task corresponding to a user code; and the computing unit is configured to computing the running task to obtain a computation result.

[0045] According to some embodiments, the computing unit is further configured to, after completing computation of the running task, release computing resources corresponding to the computation and continue to monitor the task queue.

[0046] A seventh aspect of the present disclosure provides an electronic device including a memory and a processor coupled to each other, the processor is configured to execute the program instructions stored in the memory to implement at least one of a user code running method for a programming platform of the first aspect, a user code running method for a programming platform of the second aspect or a user code running method for a programming platform of the third aspect.

[0047] Therefore, the service node is configured to realize the connection with the user terminal, place the user code in the queue, and the computing node is configured to compute the user code, thereby realizing the running of the user code. In the running process of the user code, the service node and the computing node are set separately, so that the connection of the user terminal and the computation of the user code are completed by the service node and the computing node respectively, asynchronous communication of the user code running is realized, and the concurrency capability of the system is improved.

[0048] An eighth aspect of the present disclosure provides a computer readable storage medium having stored thereon program instructions capable of being executed by a processor, the program instructions being used for implementing at least one of a user code running method for a programming platform of the first aspect, and a user code running method for a programming platform of the second aspect, or a user code running method for a programming platform of the third aspect.

[0049] Therefore, in the running process of the user code, the service node is configured to realize the connection with the user terminal, and the computing node is configured to compute the user code, so that asynchronous communication of the user code running is realized, and the concurrency capability of the system is improved.

BRIEF DESCRIPTION OF THE DRAWINGS

[0050] The accompanying drawings, which are incorporated in and constitute a part of this specification, illustrate embodiments consistent with the disclosure and, together with the description, serve to explain the technical solutions in the embodiments of the disclosure.

[0051] FIG. 1 is a schematic flowchart of a user code running method for a programming platform according to an embodiment of the present disclosure;

[0052] FIG. 2 is another schematic flowchart of a user code running method for a programming platform according to an embodiment of the present disclosure;

[0053] FIG. 3 is yet another schematic flowchart of a user code running method for a programming platform according to an embodiment of the present disclosure;

[0054] FIG. 4 is a schematic structural diagram of a programming platform according to an embodiment of the present disclosure;

[0055] FIG. 5 is a schematic structural diagram of a service node 41 of a

programming platform according to an embodiment of the present disclosure;

[0056] FIG. 6 is a schematic structural diagram of a computing node 43 of a programming platform according to an embodiment of the present disclosure;

[0057] FIG. 7 is another schematic structural diagram of a programming platform according to an embodiment of the present disclosure;

[0058] FIG. 8 is a schematic block diagram of an electronic device according to an embodiment of the present disclosure;

[0059] FIG. 9 is a schematic block diagram of a storage device according to an embodiment of the present disclosure.

DETAILED DESCRIPTION

[0060] Various exemplary embodiments, features and aspects of the disclosure will be described below in detail with reference to the accompanying drawings. The same reference signs in the drawings represent components with the same or similar functions. Although each aspect of the embodiments is shown in the drawings, the drawings are not required to be drawn to scale, unless otherwise specified.

[0061] The embodiments of the present disclosure will now be described in detail with reference to the accompanying drawings.

[0062] In the following, the technical solutions in the embodiments of the present disclosure will be clearly and completely described in connection with the accompanying drawings of the embodiments of the present disclosure, and it is apparent that the described embodiments are only a part of the embodiments of the present disclosure and not all of the embodiments. Based on the embodiments in the present disclosure, all other embodiments obtained by a person of ordinary skill in the art without involving any inventive effort fall within the scope of the present disclosure. The terms "first" and

"second" are used herein for descriptive purposes only and are not to be construed as indicating or implying relative importance or implying the number of indicated technical features.

[0063] In some embodiments, with the popularity of artificial intelligence, machine learning and artificial intelligence programming education are also becoming an important component of programming education. Unlike the conventional program introduction education, the artificial intelligence code executed by the students in the artificial intelligence programming education often needs to perform a large amount of computation, and the corresponding resource cost is large, such as occupancy rate of Central Processing Unit (CPU)/Graphics Processing Unit (GPU), memory space. In the programming education platform of the related art, a separate Kernel is created for each user and stored in the memory of the server, so that each user performs programming experiments in the created separate Kernel. As a result, each Kernel occupies a large amount of CPU/GPU resources and memory space due to the large resource overhead required by the artificial intelligence code. If the Kernel of each user resides in the memory, the overall overhead is large, and resource utilization is low. Since each user and the Kernel are in a strict one-to-one correspondence, the number of users capable of on-line programming learning at the same time is limited to the number of Kernel supported by the server. Since the running environment of the Kernel process of each user is unchanged, it is impossible to flexibly schedule, according to the resource occupation of the user code, the Kernel process to different machines for execution. In the case of multi-machine deployment, the resource allocation is easily unreasonable, which is unfavorable to improving the system concurrency capability. In this way, in the online programming education of artificial intelligence, the costs and overheads of server are greatly increased, and the programming education is difficult to popularized.

[0064] Based on this, an embodiment of the present disclosure provides a user code running method for a programming platform, in which a set of equal service nodes and a set of equal computing nodes are constructed, and after a user submits a user code, the user code is transmitted to any one of the service nodes through a websocket (one of

protocols for full-duplex communication) technology. The service node creates a running task, places the running task into the task queue and maintains the current session connection; after the running task is acquired from the task queue, any computing node executes the user code, feeds back the execution result to the corresponding service node in real time through the task queue, and then the computing node releases the resources. The service node reads the result from the task queue, pushes the result back to the user terminal in real time through the protocol for full-duplex communication, and ends the current session after the push is completed. The service node supports multi-task concurrent connection, and the computing node supports single-task execution.

[0065] The flowchart of the method is shown in FIG. 1 that is a flowchart of a user code running method for a programming platform according to an embodiment of the present disclosure. In particular, the method according to embodiments of the present disclosure includes operations S101 to S104.

[0066] In operation S101, the service node acquires a user code for representing a user terminal, and places the user code as a running task in the task queue.

[0067] The programming platform in the embodiments of the present disclosure includes a service node, a task queue and a computing node.

[0068] The service node may acquire the user code of any one of multiple user terminals, and may acquire the user codes of multiple user terminals at the same time, so that the concurrent connections of multiple user terminals are supported. The service node may be implemented by any electronic device having a processing capability, such as a server, a computer, and the like. The electronic device may also be configured to perform other functions, for example, the electronic device may also be configured to perform the computation of the running task acquired by the computing node, i.e., the service node and the computing node may be implemented by the same electronic device or by different electronic devices. The manner in which the service node receives the user code includes, but is not limited to, websocket communication technology, Hyper

Text Transfer Protocol (HTTP) communication technology, and is not specifically limited herein. The sources of the user code received by the service node may be varied, for example, the user code formed by the user-edited code acquired from the code editing window carried by the user terminal, or the user code received by the user terminal and transmitted by the code editing device connected to the user terminal.

[0069] In the embodiments of the present disclosure, the programming platform includes at least two service nodes, when the service node is configured to acquire the user code for representing the user terminal, the user code is assigned to the at least two service nodes through a reverse proxy. These service nodes open a service to a front-end application, and the front-end application communicates with the service node by using a full-duplex communication protocol. The service node receives a user code of a user terminal assigned by a reverse proxy server. The user code of the user terminal represents identification information of the user terminal, for example, indicates a serial number of the user terminal or a user account of the user terminal. User code assignment between the user terminal and at least two service nodes is achieved by means of a reverse proxy server, and when the reverse proxy server acquires the user code of the user terminal, the user code can be assigned to the at least two service nodes, so that each service node is assigned the user code through the reverse proxy. Similarly, the service node may receive user codes of multiple user terminals at the same time, so that the concurrent connections of the multi-user terminals are supported.

[0070] After the user code is received, the service node creates a running task for each user code, and places the running task in a task queue, so that the task queue includes running tasks that are placed by the service node and have one-to-one correspondence with the user codes. The queue may be a storage system, such as a Remote Dictionary Server (Redis) list, a message oriented middleware, such as a Message Queue (RabbitMQ), or the like.

[0071] In the embodiments of the present disclosure, the running task corresponding to the user code is placed separately according to the resource dependence requirement

of the user code. Specifically, the task queue includes a first task queue and a second task queue, and the running tasks in the first task queue and the second task queue are computed according to different code resources, for example, the running tasks in the first task queue depend on CPU code running environment resources, and the running tasks in the second task queue depend on GPU code running environment resources. The assignment of the running tasks in the first task queue and the second task queue depends on the computation characteristic of the user code, i.e. the computation characteristic is configured to indicate that the user code belongs to the first task queue or the second task queue. The representation of the computation characteristic includes, but is not limited to, a character string identifier of a preset bit length in a user code, a binary code, and the like. Therefore, when the service node places the user code as the running task in the task queue, the service node places the user code as the running task in the first task queue or the second task queue according to the computation characteristic of the user code.

[0072] In the embodiments of the present disclosure, a task queue may include at least two sub-task queues, and association relationships between service nodes and multiple sub-task queues may be established in advance. The service node acquires user codes for representing user terminals, and places the user codes as running tasks in different sub-task queues; or establishes an association relationship between the multiple service nodes and the multiple sub-task queues, in this way, a part of the service nodes assigns the received user codes to a part of the sub-task queues with which the association relationships are established, and another part of the service nodes assigns the received user codes to another part of the sub-task queues with which the association relationships are established, so as to implement association relationships between the service nodes and the task queues.

[0073] In operation S102, the computing node monitors the task queue to acquire the running task.

[0074] The task queue includes a running task, so that the computing node monitors

the task queue and obtains the running task when detecting that the task queue includes the running task that can be executed by the computing node. The manner in which the computing node acquires the running task from the task queue includes, but is not limited to, the task queue actively assigns the running task to the computing node, or the computing node requests acquisition of the running task from the task queue, and the task queue assigns the running task to the computing node according to the request.

[0075] In embodiments of the present disclosure, the programming platform comprises at least two computing nodes. When the computing node monitors a task queue to acquire a running task, at least two computing nodes monitor the task queue to request acquisition of the running task from the task queue. The task queue assigns one running task to one of the computing nodes, so that the task queue completes a reasonable assignment of running task. A rule for assigning a running task to a computing node by a task queue may be customized, for example, priority levels of all computing nodes are stored in advance, and when the task queue receives a request for acquiring a same running task from multiple computing nodes at the same time, the running task is assigned to a computing node with a high priority level. Alternatively, the running task is assigned to a computing node having the least number of running tasks in the preset time, which is not specifically limited herein.

[0076] In the embodiments of the present disclosure, when the task queue includes the first task queue and the second task queue, and the computing node includes the first computing node and the second computing node, the computing node monitors the task queue, specifically, the first computing node monitors the first task queue, and the second computing node monitors the second task queue, so that different computing nodes respectively monitor different task queues, which can facilitate management, and improve efficiency of acquiring running tasks. The first computing node and the second computing node may be respectively located on servers with different configurations, so that different computing and storage resources are respectively owned, and computation of different user codes can be realized. With respect to the computation of the user code, the computation and storage resources on which the user code is computed can be

learned from the computation characteristic, that is, the computation characteristic of the user code is configured to indicate that the user code belongs to the first task queue or the second task queue, and further to indicate that the user code is computed by the first computing node or the second computing node located on the servers with different configuration.

[0077] Furthermore, a first computing node is provided on a CPU server, and a second computing node is provided on a GPU server. The first task queue includes a running task dependent on the CPU code running environment resource, and the second task queue includes a running task dependent on the GPU code running environment resource. Thus, the first computing node on the CPU server monitors the first task queue to acquire the running task, and the second computing node on the GPU server monitors the second task queue to acquire the running task.

[0078] In operation S103, the computing node computes the running task to obtain a computation result.

[0079] The computing node computes the acquired running task by using the available computing and memory resources on the server, so that the computation result can be obtained. After completing the computation of the running task, the computing node releases the computing resource corresponding to the computation and continues to monitor the task queue. That is, after completing the computation of the current running task, the computing node releases the resource and continues to monitor the task queue to acquire a new running task, so that the computing node maximally reuses the computing and memory resources. In the present disclosure, regardless of whether the computing node can acquire any running task in the task queue, or the first computing node and the second computing node respectively acquire the running tasks in the first task queue and the second task queue, the same computing node can release the resources when the running task is executed, and continuously acquire a new running task, so long as the computing and memory resources of the computing node conform to the running environment resources of the user code. Therefore, the user codes may be

executed in different computing nodes, so that the computing and memory resources of the computing nodes are reused, and the hardware resources of the server are fully and reasonably utilized.

[0080] As discussed above, the service node places, according to a computation characteristic of the user code, the user code as a running task in the first task queue or the second task queue. The first computing node and the second computing node acquire running tasks from the first task queue and the second task queue, respectively. The first computing node and the second computing node on the servers with different configurations respectively compute running tasks by using respective computing and memory resources to obtain the computing results. Therefore, the task queue distributes the running tasks to the computing nodes for running according to the actual requirements of the user codes, and the computing nodes belong to servers with different configurations, so as to realize reasonable utilization of the resources. For example, the first task queue is a CPU task queue, the second task queue is a GPU task queue, the first computing node and the second computing node are respectively located on the CPU server and the GPU server, the first computing node computing the running task from the first task queue and computes the running task, and the second computing node acquires the running task from the second task queue and computes the running task. Specifically, a computation characteristic of a user code indicates that the corresponding user code belongs to the first task queue, and the first computing node acquires a running task from the first task queue, and computes the running task by using the computing and memory resources on the CPU server to obtain a computation result. The computing node on the CPU server may execute conventional code that does not rely on the GPU server resources, and the computing node on the GPU server may execute code that uses the GPU server resources including, but not limited to, neural network model training, picture processing, and the like.

[0081] In operation S104, the service node acquires the computation result from the computing node, and feeds back the computation result to the user terminal.

[0082] After the computing node computes the running task to obtain the computation result, the service node acquires the computation result from the computing node and feeds back the computation result to the user terminal. The manner in which the service node acquires the computation result from the computing node includes but is not limited to an acquisition request, a subscription, and the like, and is not specifically limited herein. When the service node acquires the computation result from the computing node in a manner of the acquisition request, the service node monitors the computing node, sends the acquisition request for acquiring the computation result to the computing node, and the corresponding computing node feeds back the computation result to the service node according to the acquisition request. When the service node acquires the computation result from the computing node in a subscription manner, the computing node directly feeds back the computation result to the service node after the computing the running task and obtaining the computation result.

[0083] In the embodiments of the present disclosure, the programming platform further includes a message queue, and the message queue is configured to temporarily store the computation result obtained by computing the running task by the computing node . The message queue may be a storage system such as a Redis list or a message oriented middleware such as a RabbitMQ. When a message-oriented service node acquires a computation result from the computing node to feedback the computation result to a user terminal, the computing node places the computation result in a message queue; and the service node monitors the message queue to obtain a computation result and feeds back the computation result to the user terminal. When the service node obtains a computation result from the computing node to feed back the computation result to the user terminal, the service node subscribes to the computation result of the computing node from the message queue, and feeds back the computation result to the user terminal by using a communication technology such as a websocket.

[0084] In the embodiments of the present disclosure, after acquiring the user code for representing the user terminal, the service node establishes a session connection with the user terminal; the service node maintains the session connection with the user terminal

before acquiring the computation result; and after acquiring the computation result from the computing node, the service node disconnects the session connection with the corresponding user terminal. Therefore, the service node maintains connection between the service node and the user terminal before acquiring the computation result, thereby facilitating feedback of the computation result to the user terminal. In the running of the user code, the service node is set separately from the computing node, and the connection with the user terminal and the computation of the user code are completed by the service node and the computing node, respectively. The service node maintains the long connection of the session with the user terminal while the computing node releases, after completing the computation of the running task, the corresponding computation resources and continues to monitor the task queue, so that the computing node continuously reuses the computation resources, and the computing node not be occupied all the time by the running process of the user code, so that asynchronous communication of the user code running is realized, and the concurrency capability of the system is improved. After acquiring the running task and obtaining the computing result by computing the running task, the computing node releases the corresponding computing resources, and continuously acquires a new running task , so that the computing resources of the computing node are not occupied all the time, the computing resources of the same computing node are reused, and resource utilization rate is improved.

[0085] In the above manner, the programming platform includes a service node, a task queue, and a computing node. The service node and the computing node are set separately, and the service node acquires a user code for representing a user terminal, and places the user code as a running task in the task queue. The computing node monitors the task queue to acquire the running task. The computing node computes the running task to acquire a computation result. The service node acquires the computation result from the computing node to feed back the computation result to the user terminal, so that the service node implements the connection with the user terminal, and the computing node implements the computation of the user code, thereby realizing running

of the user code. In addition, during the running of the user code, the service node and the computing node are set separately, so that the connection with the user terminal and the computation of the user code are completed by the service node and the computing node, respectively, thereby realizing asynchronous communication for the running of the user code and improving the concurrency capability of the system.

[0086] The present disclosure also provides a service node which is also capable of executing a user code running method for a programming platform. Referring to FIG. 2,, another schematic flowchart of a user code running method for a programming platform according to an embodiment of the present disclosure is shown. In particular, the method of embodiments of the present disclosure includes operations S201 to S203.

[0087] In operation S201, a user code for representing a user terminal is acquired.

[0088] The user code running method for a programming platform in the embodiments of the disclosure is performed by the service node in any embodiment of the user code running method for a programming platform. The programming platform includes a service node, a task queue and a computing node.

[0089] The service node may receive the user code of any one of user terminals of multiple user terminals, and may simultaneously receive the user codes of the multiple user terminals, so that the concurrent connections of the multi-user terminals are supported. The manner in which the service node receives the user code includes, but is not limited to, the websocket communication technology and the HTTP communication technology, and is not specifically limited herein. The sources of the user code received by the service node may be varied, for example, the user code formed by the user-edited code acquired from the code editing window carried by the user terminal, or the user code received by the user terminal and transmitted by the code editing device connected to the user terminal.

[0090] In the embodiments of the present disclosure, the user code is assigned to the service node through the reverse proxy, so that the user terminal does not need to

establish a one-to-one corresponding association relationship with the service node, and the reverse proxy server assigns the user code of the user terminal.

[0091] In operation S202, the user code is placed as a running task in a task queue to enable a computing node to acquire the running task from the task queue and compute the running task.

[0092] The service node places the user codes as running tasks in the task queue, so that the running tasks in the task queue are uniformly assigned, and the concurrent response capability of the system can be improved through the queuing mechanism. The service node acquires the user code of the user terminal, maintains the session connection with the user terminal, and realizes the connection with the user terminal. For the user code, the user code is placed as a running task in the task queue to enable the computing node to acquire the running task from the task queue and compute the running task, so that the connection with the user terminal and the computation of the user code are respectively completed by the service node and the computing node, thereby realizing asynchronous communication of the user code running, and improving the concurrency capability of the system.

[0093] In the embodiments of the present disclosure, the task queue includes a first task queue and a second task queue, and the computing node includes a first computing node and a second computing node. When a service node places the user code as the running task in the task queue to enable the computing node to acquire the running task from the task queue and compute the running task, the user code is placed, according to a computation characteristic of the user code, as a running task in the first task queue or the second task queue to enable a first computing node to acquire the running task from the first task queue and compute the running task, or to enable a second computing node to acquire the running task from the second task queue and compute the running task.

[0094] For descriptions of the remaining operations S201 and S202, reference may be made to the above operations S101 to S103, and details are not described herein.

[0095] In operation S203, a computation result is acquired from the computing node, and the computation result is fed back to the user terminal.

[0096] After the computing node obtains the computation result, the service node acquires the computation result from the computing node and feeds back the computation result to the user terminal. In the embodiments of the present disclosure, the programming platform further includes a message queue, and the message queue is configured to temporarily store the computation result obtained by computing the running task by the computing node. When the service node obtains the computation result from the computing node to feed back the computation result to the user terminal, the service node monitors the message queue to acquire the computation result and feeds back the computation result to the user terminal. The computation result acquired by the service node the computation result placed in the message queue by the computing node.

[0097] In the embodiments of the present disclosure, after acquiring the user code for representing the user terminal, the service node establishes a session connection with the user terminal; the service node maintains a session connection with the user terminal before acquiring the computation result; after acquiring the computation result from the computing node, the service node disconnects the session connection with the corresponding user terminal. Therefore, the service node maintains connection between the service node and the user terminal before acquiring the computation result, thereby facilitating feedback of the computation result to the user terminal.

[0098] In the above manner, the service node acquires a user code for representing the user terminal; places the user code as a running task in a task queue to enable a computing node to acquire the running task from the task queue and compute the running task; and acquires a computation result from the computing node, and feeds back the computation result to the user terminal, thereby implementing the running of the user code. The service node is configured to connect with the user terminal and place the user code sent from the user terminal as a running task in the task queue, so that the computing node independent of the service node acquires the computation result from

the task queue and compute the running task, and then the service node acquires the computation result from the computing node, and feeds back the computation result to the user terminal, thereby implementing the running of the user code.

[0099] The present disclosure also provides a computing node that is also capable of executing a user code running method for a programming platform. Referring to FIG. 3, yet another schematic flowchart of a user code running method for a programming platform according to an embodiment of the present disclosure is shown. In particular, the method of embodiments of the present disclosure includes operations S301 to S302.

[00100] In operation S301, a task queue is monitored to acquire a running task.

[00101] The user code running method for a programming platform in the embodiments of the disclosure is performed by the computing node in any embodiment of the user code running method for a programming platform. The task queue includes the running task corresponding to the user code.

[00102] The computing node monitors the task queue including the running task, and acquires the running task corresponding to the user code one-to-one from the task queue.

[00103] The computing node includes a first computing node and a second computing node, and the first computing node and the second computing node are located on the CPU server and the GPU server, respectively, so that the first computing node monitors the tasks that depend on computing and memory resources in the CPU server in the task queue, and the second computing node monitors the tasks that depend on computing and memory resources in the GPU server in the task queue. Specifically, the task queue may include a first task queue and a second task queue, the first computing node and the second computing node monitor the first task queue and the second task queue, respectively, to obtain the running tasks.

[00104] That is, the user code corresponds to the task queue, and when the computing node in the embodiments of the present disclosure is the first computing node, the

running task that depend on computing and memory resources in the CPU server is acquired from the first task queue; and when the computing node is the second computing node, the running task that depend on computing and memory resources in the GPU server is acquired from the second task queue.

[00105] In operation S302, the running task is computed to obtain a computation result.

[00106] The computing node computes the running task by using the computing and memory resources on the server to which it belongs to obtain the computing result. After the computation of the running task is completed, the computing node releases the corresponding computing resources and continues to monitor the task queue. That is, after the computation of the current running task is completed, the computing node releases the resources and continues to monitor the task queue to obtain a new running task, so that the computing node maximally reuses the computing and memory resources.

[00107] In the manner described above, the task queue is monitored to obtain the running task; the task queue includes a running task and a user code corresponding to the running task; and the running task is computed to obtain the computation result, thereby realizing the acquisition of running task by the computing node in an orderly manner. The task queue includes the running task, and the running task corresponds to the user code. Therefore, the computing node monitors the task queue including the running task, so as to acquire the running task corresponding to the user code, and then computes the running task to obtain the computation result, thereby realizing the acquisition and computation of the running task.

[00108] The present disclosure also provides a programming platform. Referring to FIG. 4, schematic structural diagram of a programming platform according to an embodiment of the present disclosure is shown. Specifically, the programming platform 40 in the embodiments of the present disclosure includes a service node 41, a task queue

42 and a computing node 43. The service node 41 is configured to acquire a user code for representing a user terminal, and place the user code as a running task in the task queue 42; the computing node 43 the computing node is configured to monitor the task queue 42 to acquire the running task; the computing node 43 is further configured to compute the running task to obtain a computation result; and the service node 41 is further configured to acquire the computation result from the computing node 43, and feedback the computation result to the user terminal.

[00109] In the embodiments of the present disclosure, the programming platform comprises at least two computing nodes; when the computing node 43 is configured to monitor the task queue 42 to obtain the running task, the at least two computing nodes 43 are configured to monitor the task queue 42 and request the task queue 42 to acquire the running task; and the task queue 42 is configured to assign one running task to one of the computing nodes 43.

[00110] In the embodiments of present disclosure, the task queue 42 includes a first task queue and a second task queue, and at least two computing nodes 43 include a first computing node and a second computing node. When the service node 41 is configured to place the user code as a running task in the task queue 42, the service node 41 is specifically configured to place the user code as a running task in the first task queue or the second task queue according to the computation characteristic of the user code. The computing node 43 is configured to monitor the task queue 42, in particular, the first computing node monitors the first task queue and the second computing node monitors the second task queue.

[00111] In the embodiments of present disclosure, the programming platform comprises at least two service nodes. The service node 41 is configured to acquire a user code for representing the user terminal, and the user code is assigned to the at least two of the service nodes 41 through the reverse proxy.

[00112] In the embodiments of the present disclosure, the computing node 43 is further

configured to release the corresponding computing resources and continue to monitor the task queue 42 after the computation of the running task is completed.

[00113] In the embodiments of the present disclosure, the programming platform 40 further includes a message queue 44. The service node 41 is configured to acquire a computation result from the computing node 43 to feed back the computation result to the user terminal, and the computing node 43 is further configured to place the computation result in the message queue 44. The service node 41 is also configured to monitor the message queue 42 to acquire a computation result and to feed back the computation result to the user terminal.

[00114] In the embodiments of the present disclosure, the service node 41 is further configured to maintain a session connection with the user terminal before the computation result is acquired.

[00115] The service node 41 in the programming platform 40 shown in Fig. 4 is also configured to perform the user code running method for the programming platform. Specifically, referring to FIG. 5, a schematic structural diagram of a service node 41 of a programming platform according to an embodiment of the present disclosure is shown. The service node 41 includes a receiving unit 411, a placing unit 412, and an acquiring unit 413. The receiving unit 411 is configured to acquire a user code for representing a user terminal; the placing unit 412 is configured to place the user code as a running task in a task queue to enable the computing node to acquire the running task from the task queue and compute the running task; and the acquiring unit 413 is configured to acquire a computation result from the computing node and feedback the computation result to the user terminal.

[00116] In the embodiments of the present disclosure, the placing unit 412 is configured to place the user code as a running task in the task queue, so as to enable the computing node to acquire the running task from the task queue and compute the running task, and further configured to place, according to the computation characteristic

of the user code, the user code as a running task in the first task queue or the second task queue to enable the first computing node to acquire the running task from the first task queue and compute the running task, or to enable the second computing node to acquire the running task from the first task queue and compute the running task.

[00117] In the embodiments of the present disclosure, the receiving unit 411 is configured to be assigned the user code through the reverse proxy when the user code for representing the user terminal is acquired.

[00118] In the embodiments of the present disclosure, the service node 41 maintains a session connection with the user terminal before the computation result is acquired by the acquiring unit 413.

[00119] The computing node 43 of the programming platform 40 shown in Fig. 4 is also configured to perform the user code running method for the programming platform. Specifically, referring to FIG. 6, a schematic structural diagram of a computing node 43 of a programming platform according to the present disclosure is shown. The computing node 43 includes a monitoring unit 431 and a computing unit 432. The monitoring unit 431 is configured to monitor a task queue to acquire a running task, where the task queue includes a running task, and the running task corresponds to a user code. The computing unit 432 is configured to compute a running task to obtain a computation result.

[00120] In the embodiments of the present disclosure, after the computation of the running task is completed by the computing unit 432, the computing node 43 releases the computing resources corresponding to the computation and continues to monitor the task queue.

[00121] To further illustrate the programming platform of the present disclosure, reference is made to FIG. 7, which is another schematic structural diagram of the programming platform according to an embodiment of the present disclosure.

[00122] The programming platform 70 in the embodiment of the present disclosure

includes a service node 71, a task queue 72 including a first task queue 721 and a second task queue 722, a computing node 73 including a first computing node 731 located on the CPU server 701 and a second computing node 732 located on the GPU server 702, and a message queue 74.

[00123] In a specific application, the service node 71 may receive a user code of any one of multiple user terminals, and may simultaneously receive user codes of multiple user terminals, so as to implement the acquisition of the user code for representing the user terminal; the service node 71 performs a websocket connection with the user terminal, that is, the service node 71 obtains the user code of the user terminal by means of websocket communication. When the user code is placed as a running task in the task queue 72, if it is determined that the user code belongs to the first task queue 721 according to the computation characteristic of the user code and the code running resource of the CPU server 701 will be used, then the user code is placed in the first task queue 721 as a running task, and the first computing node 731 monitors the first task queue 721 to acquire the running task and computes the running task to obtain a computation result. Alternatively, if it is determined that the user code belongs to the second task queue 722 according to the computation characteristic of the user code and the code running resource of the GPU server 702 will be used, then the user code is placed in the second task queue 722 as a running task, and the second computing node 732 monitors the second task queue 722 to acquire the running task and computes the running task to obtain a computation result. The first computing node 731 and the second computing node 732 both place the computation result in the message queue 74, and the service node 71 monitors the message queue 74 to acquire the computation result and feeds back the computation result to the corresponding user terminal. For descriptions of the service node 71, the task queue 72, the computing node 73 and the message queue 74 in the embodiments of the present disclosure, reference may be made to the embodiment of the programming platform described above, and details are not described herein.

[00124] In the above manner, the service node places the received user codes as running tasks in different task queues respectively; then, the computing nodes on

different servers respectively monitor different task queues to acquire corresponding running tasks, so as to compute the running tasks by using different computing and storage resources to obtain the computing results, and all the computing nodes place the computing results in the message queue; finally, the service node acquires the computation result of the computing node from the message queue, and feeds back the computation results to the user terminal. In the whole running process of the user code, the service node is set separately from the computing node, so that the connection with the user terminal and the computation of the user code are respectively completed by the service node and the computing node, the computation resources of the computing node can be continuously reused, and different computing nodes correspond to different message queues, so that the user codes dependent on different computing resources and storage resources are acquired for computation, asynchronous communication for the user code running is realized, and the concurrency capability of the system is improved.

[00125] In the embodiments of the present disclosure, the programming platform may be an online programming education platform or an artificial intelligence platform. For example, the programming platform is an online programming education platform, which is oriented to a school or a training institution, etc., and the user terminal is a student terminal that accesses the online programming education platform, so that the user code is running. The online programming education platform includes a service node, a task queue and a computing node, where the task queue is a Redis list. The student inputs a user code through a programming interface of the student terminal, and the service node can receive the user code from the student terminal, and can simultaneously receive the user codes from multiple student terminals, so as to implement the reception of the user code sent from the student in the online programming education platform; the service node places the user code as a running task in the Redis list that is the task queue. The computing node monitors the Redis list to acquire a running task, and computes the running task to obtain a computation result. The service node acquires a computation result from the computing node to feed back the computation result to the student terminal.

[00126] In the embodiments of the present disclosure, the programming platform is an online programming education platform, and the user terminal is a student terminal that accesses the online programming education platform. The online programming education platform includes a service node, a task queue, a computing node, and a message queue, where the task queue is a Redis list and includes a first task queue and a second task queue; the computing node includes a first computing node and a second computing node; and the message queue is RabbitMQ. If the service node of the online programming education platform determines, according to the computation characteristic of the user code, that the user code belongs to the first task queue and uses the code running resources of the CPU server, then the user code is placed as a running task in the first task queue, and the first computing node monitors the first task queue to acquire and computes the running task to acquire a computation result. Alternatively, if the service node determines, according to the computation characteristic of the user code, that the user code belongs to the second task queue and uses the code running resource of the GPU server, then the user code is placed as a running task in the second task queue, and the second computing node monitors the second task queue to acquire and computes the running task to acquire a computation result. Both the first computing node and the second computing node place the computation results in the message queue RabbitMQ, and the service node monitors the RabbitMQ to acquire the computation results and feeds back the computation results to respective student terminals.

[00127] In the above manner, for a school or a training institution, the online programming education platform can meet the requirement that a larger number of students simultaneously perform online programming learning on the online programming education platform, thereby realizing full utilization of computational resources and saving costs.

[00128] Referring to FIG. 8, a schematic block diagram of an electronic device according to an embodiment of the present disclosure is shown. Specifically, the electronic device 800 in the embodiments of the present disclosure includes a memory 810 and a processor 820 coupled to each other. The memory 810 is configured to store

program instructions and data required for processing by the processor 820.

[00129] The processor 820 controls the memory 810 and itself to implement any embodiment of the user code running method for the programming platform described above. The processor 820 may also be referred to as a Central Processing Unit (CPU). The processor 820 may be an integrated circuit chip having signal processing capabilities. The processor 820 may also be a general purpose processor, a Digital Signal Processor (DSP), an Application Specific Integrated Circuit (ASIC), a Field-Programmable Gate Array (FPGA) or other programmable logic device, a discrete gate or transistor logic element, a discrete hardware component. The general purpose processor may be a microprocessor or the processor may be any conventional processor or the like. In addition, the processor 820 may be jointly implemented by multiple circuit chips.

[00130] Referring to FIG. 9, a schematic block diagram of a storage device according to an embodiment of the present disclosure is shown. The memory device 900 of the present disclosure stores program instructions 901 capable of being executed by a processor, and the program instructions 901 are used for implementing any one of operations in the embodiment of the user code running method for the programming platform described above.

[00131] The storage device 900 may be specifically a medium (including a volatile storage medium or a non-volatile storage medium) in which a program instruction 901 may be stored, such as a USB flash drive, a removable hard disk, a read-only memory (ROM, Read-Only Memory), a random access memory (RAM, Random Access Memory), a magnetic disk, or an optical disk; the storage device 900 may be a server in which the program instructions 901 are stored, and the server may send the stored program instructions 901 to another device for running; or the storage device 900 may execute the stored program instructions 901.

[00132] In several embodiments provided by the present disclosure, it should be understood that the disclosed systems, devices and methods can be realized in other

ways. For example, the embodiment of the device described above is only schematic. For example, the division of the unit is only a logical function division, and there can be another division method in actual implementation, for example, multiple units or components can be combined or integrated into another system, or some features can be ignored or not implemented. On the other hand, the mutual coupling or direct coupling or communication connection illustrated or discussed can be indirect coupling or communication connection through some interfaces, devices or units, and can be electric, mechanical or other forms.

[00133] The unit described as a separation part may or may not be physically separated, and the unit displayed as a unit may or may not be a physical unit, that is, it may be located in one place, or it may be distributed to multiple network units. Some or all of the units can be selected according to the actual needs to achieve the purpose of the embodiment.

[00134] In addition, each functional unit in each embodiment of the present disclosure may be integrated in one processing unit, each unit may exist physically alone, or two or more units may be integrated in one unit. The integrated unit may be implemented in the form of hardware or in the form of software functional units.

[00135] If the integrated unit is implemented in the form of a software function unit and sold or used as an independent product, it can be stored in a computer readable storage medium. Based on such understanding, the technical solution of the present disclosure, in essence or in the form of a software product, which is stored in a storage medium, includes several instructions for making a computer device (which can be a personal computer, a server, a network device, etc.) or a processor to perform all or part of the steps of the method according to each embodiment of the present disclosure. The aforementioned storage media include: U disk, mobile hard disk, read-only memory (ROM), random access memory (RAM), disk or optical disk and other media that can store program code.

INDUSTRIAL APPLICABILITY

The disclosure discloses a user code running method for a programming platform, a platform, a node, a device and a medium. The programming platform includes a service node, a task queue, and a computing node; the user code running method for the programming platform includes: a service node acquires a user code for representing a user terminal, and places the user code as a running task in a task queue; a computing node monitors a task queue to acquire a running task; the computing node computes the running task to obtain a computation result; and the service node acquires the computation result from the computing node to feed back the computation result to the user terminal.

CLAIMS

1. A user code running method for a programming platform comprising a service node, a task queue and a computing node, the method comprising:

acquiring, by the service node, a user code for representing a user terminal, and placing the user code as a running task in the task queue;

monitoring, by the computing node, the task queue to acquire the running task;

computing, by the computing node, the running task to obtain a computation result; and

acquiring, by the service node, the computation result from the computing node, and feeding back the computation result to the user terminal.

2. The method of claim 1, wherein the programming platform comprises at least two computing nodes; and monitoring, by the computing node, the task queue to acquire the running task comprises:

monitoring, by the at least two computing nodes, the task queue, and requesting acquisition of the running task from the task queue; and

assigning, by the task queue, one running task to one of the computing nodes.

3. The method of claim 1, wherein the task queue comprises a first task queue and a second task queue, and at least two computing nodes comprise a first computing node and a second computing node;

placing the user code as the running task in the task queue comprises:

placing, according to a computation characteristic of the user code, the user code as a running task in the first task queue or the second task queue; and

monitoring, by the computing node, the task queue comprises:

monitoring, by the first computing node, the first task queue, and monitoring, by the second computing node, the second task queue.

4. The method of claim 1 or 2, wherein the programming platform comprises at least two service nodes; and wherein acquiring, by the service node, the user code for representing the user terminal comprises:

the user code being assigned to the at least two service nodes through a reverse proxy.

5. The method of any one of claims 1 to 4, wherein after completing computation of the running task, the computing node releases computing resources corresponding to the computation, and continues to monitor the task queue.

6. The method of any one of claims 1 to 5, wherein the programming platform further comprises a message queue; and wherein acquiring, by the service node, the computation result from the computation node and feeding back the computation result to the user terminal comprises:

placing, by the computing node, the computation result in the message queue; and

monitoring, by the service node, the message queue to acquire the computation result, and feeding back the computation result to the user terminal.

7. The method of claim 6, wherein the service node maintains a session connection with the user terminal before acquiring the computation result.

8. A user code running method for a programming platform, comprising:

acquiring a user code for representing a user terminal;

placing the user code as a running task in a task queue to enable a computing node to acquire the running task from the task queue and compute the running task; and

acquiring a computation result from the computing node, and feeding back the computation result to the user terminal.

9. The method of claim 8, wherein placing the user code as the running task in the task queue to enable to the computing node acquires and computes the running task in the task queue comprises:

placing, according to a computation characteristic of the user code, the user code as a running task in the first task queue or the second task queue to enable a first computing node to acquire the running task from the first task queue and compute the running task, or to enable a second computing node to acquire the running task from the first task queue and compute the running task.

10. The method of claim 8 or 9, wherein acquiring the user code for representing the user terminal comprises:

the user code being assigned through a reverse proxy.

11. The method of any one of claims 8 to 10, comprising: maintaining a session connection with the user terminal before acquiring the computation result.

12. A user code running method for a programming platform, comprising:

monitoring a task queue to acquire a running task, wherein the task queue comprises the running task corresponding to a user code; and

computing the running task to obtain a computation result.

13. The method of claim 12, comprising: after completing computation of the running task, releasing computing resources corresponding to the computation, and continuing to monitor the task queue.

14. A programming platform, comprising: a service node, a task queue and a computing node,

wherein the service node is configured to acquire a user code for representing a user terminal, and place the user code as a running task in the task queue;

the computing node is configured to monitor the task queue to acquire the running task;

the computing node is further configured to compute the running task to obtain a computation result; and

the service node is further configured to acquire the computation result from the computing node, and feedback the computation result to the user terminal.

15. The programming platform of claim 14 or 15, wherein the programming platform comprises at least two computing nodes; and when the computing node is configured to monitor the task queue to acquire the running task, at least two computing nodes are configured to monitor the task queue and request acquisition of the running task from the task queue, wherein the task queue is configured to assign one running task to one of the computing nodes.

16. The programming platform of claim 15, wherein the task queue comprises a first task queue and a second task queue, and the at least two computing nodes comprise a first computing node and a second computing node;

when the service node is configured to place the user code as the running task in the task queue, the service node is specifically configured to place, according to a computation characteristic of the user code, the user code as a running task in the first task queue or the second task queue; and when the computing node is configured to monitor the task queue, the first computing node is specifically configured to monitor the first task queue, and the second computing node is configured to monitor the second task queue.

17. The programming platform of claim 14, the programming platform comprises at least two service nodes; and when the service node is configured to acquire the user code for representing the user terminal, the user code is assigned to the at least two service nodes through a reverse proxy.

18. The programming platform of any one of claims 14 to 17, wherein the computing node is configured to, after completing computation of the running task, release computing resources corresponding to the computation and continue to monitor the task queue.

19. The programming platform of any one of claims 14 to 18, wherein the programming platform further comprises a message queue; and when the service node is configured to acquire the computation result from the computation node and feedback the computation result to the user terminal, the computing node is configured to place the computation result in the message queue; and the service node is configured to monitor the message queue to acquire the computation result, and feedback the computation result to the user terminal.

20. The programming platform of claim 19, wherein the service node is further configured to maintain a session connection with the user terminal before acquiring the computation result.

21. A service node, comprising a receiving unit, a placing unit and an acquiring unit, wherein the receiving unit is configured to acquire a user code for representing a user terminal; the placing unit is configured to place the user code as a running task in a task queue to enable a computing node to acquire and compute the running task from the task queue; and the acquiring unit is configured to acquire a computation result from the computing node and feedback the computation result to the user terminal.

22. The service node of claim 21, when the placing unit is configured to place the user code as a running task in a task queue to enable a computing node to acquire and compute the running task from the task queue, the placing unit is specifically configured to place, according to a computation characteristic of the user code, the user code as a running task in the first task queue or the second task queue to enable a first computing node to acquire and compute the running task from the first task queue, or to enable a second computing node to acquire and compute the running task from the second task queue.

23. The service node of claim 21 or 22, when the receiving unit is configured to acquire the user code for representing the user terminal, the receiving unit is specifically configured to be assigned with the user code through a reverse proxy.

24. The service node of any one of claims 21 to 23, wherein the acquiring unit is configured to maintain a session connection with the user terminal before acquiring the computation result.

25. A computing node, comprising a monitoring unit and a computing unit, wherein the monitoring unit is configured to monitor a task queue to acquire a running task, wherein the task queue comprises the running task corresponding to a user code; and the computing unit is configured to computing the running task to obtain a computation result.

26. The computing node of claim 25, wherein the computing unit is further configured to, after completing computation of the running task, release computing resources corresponding to the computation and continue to monitor the task queue.

27. An electronic device, comprising a memory and a processor coupled to each other,

wherein the processor is configured to execute the program instructions stored in the memory to implement the method of any one of claims 1 to 7, and/or to implement the method of any one of claims 8 to 11, and/or to implement the method of claim 12 or 13.

28. A computer readable storage medium storing a computer program that can be run to implement a user code running method for a programming platform of any one of claims 1 to 7, and/or storing a computer program that can be run to implement a user code running method for a programming platform of any one of claims 8 to 11, and/or storing a computer program that can be run to implement a user code running method for a programming platform of claim 12 or 13.