

METHOD FOR NETWORK BANDWIDTH ADJUSTMENT AND RELATED PRODUCTS

TECHNICAL FIELD

[1] This disclosure relates to the field of computers, and more particularly to methods for network bandwidth adjustment and related products.

BACKGROUND

[2] In a distributed deep learning training system, calculation results of different computing nodes are synchronized in stages through parameter aggregation. However, simultaneous data interaction between multiple computing nodes and parameter servers may lead to network congestion at service nodes, thereby affecting training efficiency of the whole deep learning model.

SUMMARY

[3] Implementations of the disclosure provide a method for network bandwidth adjustment and related products.

[4] According to a first aspect, implementations of the disclosure provide a method for network bandwidth adjustment. The method includes the following. A time length spent in completing at least one training iteration during execution of a training task by a work node is obtained. A bandwidth update request is transmitted to a first server in response to determining that the time length spent on the at least one training iteration is timed out. The bandwidth update request is used for requesting the first server to update a bandwidth of a service node, and the service node stores data for the training task.

[5] Optionally, before executing a N^{th} training iteration, the work node obtains from the service node (e.g., a server node) parameters required for executing the N^{th} training iteration. Implementations of the disclosure can be performed by a second server. The second server may be a server cluster or a server. In some implementations, the second server, the work node, and the service node are included in a same distributed training cluster. The server node, e.g., a

parameter server, mainly stores a parameter(s) for a deep learning training task, receives a gradient(s) pushed by the work node, and updates the local parameters. The work node obtains the parameters from the server node and pushes the gradients obtained through iterative calculation to the server node. The work node obtaining the parameters from the server node and pushing the gradients to the server node may cause network congestion at the server node and eventually cause loss of data in transmission. If the network of the server node is congested, when the work node obtains parameters from the server node and pushes gradients to the server node once again, timeout may occur, thereby affecting a subsequent training process. In implementations of the disclosure, the second server can monitor in real time or near real time a time length spent in completing a training iteration each time by the work node, and then can determine whether the training iteration each time is timed out. In the case of determining that a training iteration is timed out, it can be accurately determined that a present network bandwidth of the service node is insufficient, and then the network bandwidth of the service node can be automatically adjusted. In implementations of the disclosure, the second server can dynamically adjust the network bandwidth of the service node in real time, thereby avoiding a timeout of training by the work node and improving training efficiency.

[6] In implementations of the disclosure, on condition that the time length spent in completing the at least one training iteration during execution of the training task by the work node is timed out, the bandwidth update request is transmitted to the first server to adjust (update) the bandwidth of the service node, which can effectively solve the problem of the insufficient network bandwidth of the parameter server, thereby improving the training efficiency of the work node.

[7] In an alternative implementation, the at least one training iteration is embodied as N training iterations, and determine that the time length spent on the at least one training iteration is timed out as follows. Determine that the time length spent on the at least one training iteration is timed out, according to a first duration of the time length spent on the at least one training iteration and historical iteration duration information of executing the training task by the work node. The first duration is a duration spent in completing a N^{th} training iteration in the N training iterations during execution of the training task by the work node.

[8] Since operations of each training iteration during execution of the training task by the work node are similar, a time length spent in completing each training iteration during execution

of the training task by the work node is substantially the same. The historical iteration duration records include a duration spent in completing the at least one training iteration during execution of the training task by the work node. According to the first duration and the historical iteration duration records, it can be accurately determined whether the first duration is longer than a previous iteration duration, and then whether the time length spent in completing the at least one training iteration is timed out can be determined. In some implementations, the first duration of the time length spent on the at least one training iteration is a duration spent in currently executing a N^{th} iteration. Determining that the time length spent on the at least one training iteration is timed out can be achieved by determining that a time length spent in currently executing the N^{th} iteration is timed out.

[9] In implementations, according to the first duration and the historical iteration duration information, it is possible to accurately and quickly determine whether the time length spent in completing the at least one training iteration by the work node is timed out.

[10] In an alternative implementation, determine that the time length spent on the at least one training iteration is timed out, according to the first duration of the time length spent on the at least one training iteration and the historical iteration duration information of executing the training task by the work node as follows. A second duration is obtained according to a duration spent in completing at least one historical training iteration during execution of the training task by the work node. The second duration is an average duration spent in completing the at least one historical training iteration during execution of the training task by the work node. Determine that the time length spent on the at least one training iteration is timed out, on condition that a difference between the first duration and the second duration is greater than or equal to a first time threshold.

[11] In an alternative implementation, determine that the time length spent on the at least one training iteration is timed out, according to the first duration of the time length spent on the at least one training iteration and the historical iteration duration information of executing the training task by the work node as follows. A maximum duration among durations from a duration spent in completing a first training iteration in the N training iterations to a duration spent in completing a $(N-1)^{\text{th}}$ training iteration in the N training iterations by the work node is obtained according to the historical iteration duration information of executing the training task by the work node. The maximum duration is determined as a third duration. Determine that the time

length spent on the at least one training iteration is timed out, on condition that a difference between the first duration and the third duration is greater than or equal to a second time threshold.

[12] In implementations, it is possible to accurately and quickly determine whether the time length spent in completing the at least one training iteration by the work node is timed out.

[13] In an alternative implementation, the at least one training iteration is embodied as K training iterations that are consecutive, and determine that the time length spent on the at least one training iteration is timed out as follows. A fourth duration spent in consecutively completing the K training iterations by the work node is obtained. An average duration spent in consecutively completing K historical training iterations of the training task by the work node is obtained, and the average duration is determined as a fifth duration. Determine that the time length spent on the at least one training iteration is timed out, on condition that a difference between the fourth duration and the fifth duration is greater than or equal to a third time threshold.

[14] In implementations, it is possible to accurately and quickly determine whether a time length spent in consecutively completing multiple training iterations by the work node is timed out.

[15] In an alternative implementation, the work node and the service node are both physical nodes. Alternatively, the method for network bandwidth adjustment is applied to a second server. One of the work node and the service node is a virtual machine running on a third server, and the other of the work node and the service node is a physical node or a virtual machine running on a fourth server.

[16] In an alternative implementation, the method for network bandwidth adjustment is applied to a first virtual machine on a second server, and the second server further runs a second virtual machine and a third virtual machine. The second virtual machine is the work node, and the third virtual machine is the service node.

[17] Optionally, the second server may be a server, a cloud server, or a server cluster. In one example, the second server may be a computing node included in an OpenStack cloud platform system, and the first server is a control node included in the OpenStack cloud platform system.

[18] In an alternative implementation, the method further includes the following prior to

obtaining the time length spent in completing the at least one training iteration during execution of the training task by the work node. A training task startup script is obtained, where the training task startup script is used for obtaining the time length spent in completing the at least one training iteration during execution of the training task by the work node.

[19] In an alternative implementation, the training task startup script includes at least one of information used for determining that the time length spent on the at least one training iteration is timed out or a preset bandwidth adjustment range.

[20] In an alternative implementation, the method further includes the following. A present first bandwidth of the service node is obtained. Determine to adjust the bandwidth of the service node to a second bandwidth according to the first bandwidth and a preset bandwidth adjustment range. The bandwidth update request includes the second bandwidth, and the second bandwidth is greater than the first bandwidth.

[21] In a second aspect, implementations of the disclosure provide a device for network bandwidth adjustment. The network bandwidth adjustment includes an obtaining unit, a determining unit, and a transmitting unit. The obtaining unit is configured to obtain a time length spent in completing at least one training iteration during execution of a training task by a work node. The determining unit is configured to determine that the time length spent on the at least one training iteration is timed out. The transmitting unit is configured to transmit a bandwidth update request to a first server in response to determining that the time length spent on the at least one training iteration is timed out. The bandwidth update request is used for requesting the first server to update a bandwidth of a service node, and the service node stores data for the training task.

[22] In an alternative implementation, the at least one training iteration is embodied as N training iterations, and the determining unit is specifically configured to determine that the time length spent on the at least one training iteration is timed out, according to a first duration of the time length spent on the at least one training iteration and historical iteration duration information of executing the training task by the work node. The first duration is a duration spent in completing a N^{th} training iteration in the N training iterations during execution of the training task by the work node.

[23] In an alternative implementation, the determining unit is specifically configured to obtain a second duration according to a duration spent in completing at least one historical

training iteration during execution of the training task by the work node and determine that the time length spent on the at least one training iteration is timed out, on condition that a difference between the first duration and the second duration is greater than or equal to a first time threshold, where the second duration is an average duration spent in completing the at least one historical training iteration during execution of the training task by the work node.

[24] In an alternative implementation, the determining unit is configured to obtain a maximum duration among durations from a duration spent in completing a first training iteration in the N training iterations to a duration spent in completing a $(N-1)^{\text{th}}$ training iteration in the N training iterations by the work node, according to the historical iteration duration information of executing the training task by the work node; determine the maximum duration as a third duration; determine that the time length spent on the at least one training iteration is timed out, on condition that a difference between the first duration and the third duration is greater than or equal to a second time threshold.

[25] In an alternative implementation, the at least one training iteration is embodied as K training iterations that are consecutive, and the determining unit is specifically configured to: obtain a fourth duration spent in consecutively completing the K training iterations by the work node; obtain an average duration spent in consecutively completing K historical training iterations of the training task by the work node, and determine the average duration as a fifth duration; determine that the time length spent on the at least one training iteration is timed out, on condition that a difference between the fourth duration and the fifth duration is greater than or equal to a third time threshold.

[26] In an alternative implementation, the work node and the service node are both physical nodes. Alternatively, a method for network bandwidth adjustment is applied to a second server. One of the work node and the service node is a virtual machine running on a third server, and the other of the work node and the service node is a physical node or a virtual machine running on a fourth server.

[27] In an alternative implementation, a method for network bandwidth adjustment is applied to a first virtual machine on a second server, and the second server further runs a second virtual machine and a third virtual machine. The second virtual machine is the work node, and the third virtual machine is the service node.

[28] In an alternative implementation, the device further includes a running unit

configured to run a training task startup script before the obtaining unit obtains the time length spent on the at least one training iteration, where the training task startup script is used for obtaining the time length spent in completing the at least one training iteration during execution of the training task by the work node.

[29] In an alternative implementation, the device further includes the following. The training task startup script includes at least one of information used for determining that the time length spent on the at least one training iteration is timed out or a preset bandwidth adjustment range.

[30] In an alternative implementation, the obtaining unit is further configured to obtain a present first bandwidth of the service node. The determining unit is further configured to determine, according to the first bandwidth and a preset bandwidth adjustment range, to adjust the bandwidth of the service node to a second bandwidth. The bandwidth update request includes the second bandwidth, and the second bandwidth is greater than the first bandwidth.

[31] In a third aspect, implementations of the disclosure provide an electronic device. The electronic device includes a memory configured to store programs and a processor configured to execute the programs stored in the memory. When the programs are executed, the processor is configured to perform the method as mentioned in the above first aspect and any of alternative implementations.

[32] In a fourth aspect, implementations of the disclosure provide a computer-readable storage medium. The computer-readable storage medium stores computer programs. The computer programs include program instructions. The program instructions, when executed by the processor, cause the processor to perform the method as mentioned in the above first aspect and any of alternative implementations.

[33] In a fifth aspect, implementations of the disclosure provide a computer program product. The computer program product includes program instructions. The program instructions, when executed by a processor, cause the processor to perform the method as mentioned in the above first aspect and any of alternative implementations.

BRIEF DESCRIPTION OF THE DRAWINGS

[34] To illustrate technical solutions of implementations of the disclosure or the related art more clearly, the following will introduce accompanying drawings used in the implementations

of the disclosure or the related art.

[35] FIG. 1 is a schematic diagram illustrating an architecture of a distributed training cluster provided in implementations of the disclosure.

[36] FIG. 2 is a schematic diagram illustrating an architecture of another distributed training cluster provided in implementations of the disclosure.

[37] FIG. 3 is a schematic diagram illustrating an architecture of a distributed training platform system provided in implementations of the disclosure.

[38] FIG. 4 is a flow chart illustrating a method for network bandwidth adjustment provided in implementations of the disclosure.

[39] FIG. 5 is a flow chart illustrating another method for network bandwidth adjustment provided in implementations of the disclosure.

[40] FIG. 6 is a flow chart illustrating yet another method for network bandwidth adjustment provided in implementations of the disclosure.

[41] FIG. 7 is a schematic structural diagram illustrating a device for network bandwidth adjustment provided in implementations of the disclosure.

[42] FIG. 8 is a schematic structural diagram illustrating a server provided in implementations of the disclosure.

DETAILED DESCRIPTION

[43] Terms “first”, “second”, “third”, and the like used in implementations of the specification, the claims, and the accompany drawings of the disclosure are used to distinguish similar objects rather than describe a particular order or priority. In addition, terms “include”, “comprise”, and “have” as well as variants thereof are intended to cover non-exclusive inclusion, for example, including a series of steps or units. A method, system, product, or apparatus is not limited to clearly listed steps or units; on the contrary, it can optionally include other steps or units that are not clearly listed or inherent to the process, method, product, or apparatus. Multiple refers to two or more.

[44] A method for network bandwidth adjustment provided in implementations of the disclosure is applied to a distributed training cluster. The distributed training cluster includes a scheduler node, one or more work nodes, and one or more service nodes. The scheduler node is configured to run a startup script of a training task. The work node is configured to execute the

training task and push a gradient(s) obtained through training iteration to the server node. The server node, as a parameter server, is mainly configured to store a parameter(s) for the training task, receive the gradient pushed by the work node, and update the local parameters.

[45] In some implementations, the distributed training cluster for deep learning training may include a scheduler node, multiple work nodes, and multiple service nodes. The multiple work nodes obtain parameters from the service nodes and push gradients to the service nodes simultaneously, which may cause network congestion at the service nodes and eventually cause loss of data in transmission. When the work nodes obtain parameters from the service nodes or push gradients to the service nodes once again, timeout may occur, such that a subsequent training process may be affected. Therefore, ensuring a network bandwidth of the service node is the key to smooth completion of the deep learning task.

[46] The following illustrates architectures of two distributed training clusters.

[47] FIG. 1 is a schematic diagram illustrating an architecture of a distributed training cluster provided in implementations of the disclosure. As illustrated in FIG. 1, the distributed training cluster includes a scheduler node 101, one or more work nodes 102, and one or more service nodes (also known as server nodes) 103. The scheduler node 101, the work node 102, and the service node 103 are all physical nodes, such as servers. In FIG. 1, the work node 102 is configured to execute a training task and push a gradient obtained through training iteration to the service node 103. The service node 103, as a parameter server, is mainly configured to store a parameter(s) of the training task, receive the gradient pushed by the work node 102, and update the local parameters. The scheduler node 101 is configured to run a startup script of the training task (e.g., a training task startup script), monitor a duration of each training iteration performed by the work node 102, and update a bandwidth of the service node 103 through a first server in response to timeout of any training iteration performed by the work node 102. In some implementations, the training task startup script includes computer program codes for implementing the method for network bandwidth adjustment provided in implementations of the disclosure. For example, the script includes program codes for implementing one or more functions such as polling durations of one or more training iterations by each of multiple work nodes executing the training task, determining timeout of the training iteration, determining how to adjust the network bandwidth, and so on. In some implementations, the training task startup script is further used for starting the training task, or may be started in response to starting of the

training task.

[48] FIG. 2 is a schematic diagram illustrating an architecture of another distributed training cluster provided in implementations of the disclosure. In FIG. 2, a scheduler node 201, a work node(s) 202, and a service node(s) 203 are all virtual machines. Data interaction between the scheduler node 201, the work node 202, and the service node 203 can be conducted through a private network obtained by adopting the single root I/O virtualization (SR-IOV) technology, e.g., a SR-IOV network. For example, the scheduler node 201, the work node 202, and the service node 203 may run on a same server (corresponding to a second server) or a same server cluster. The scheduler node 201, the work node 202, and the service node 203 are all virtual machines managed by an OpenStack platform. FIG. 3 is a schematic diagram illustrating an architecture of a distributed training platform system provided in implementations of the disclosure. As illustrated in FIG. 3, the distributed training platform system includes a control node 301 and a computing node 302 (corresponding to the distributed training cluster in FIG. 2). The interaction between the control node 301 and the computing node 302 can be conducted through a public network. The scheduler node 201 of the computing node 302 interacts with the control node 301 through the public network (e.g., Internet). That is, the distributed training cluster in FIG. 2 includes multiple virtual machines managed by the OpenStack platform, e.g., the scheduler node 201, the work node 202, and the service node 203. Optionally, there are only SR-IOV network cards on the work node 202 and the service node 203, while there are SR-IOV network cards and Ethernet cards on the scheduler node 201. During creation of the nodes, corresponding network bandwidths are set for the SR-IOV network card. Optionally, a network system service, e.g., a neutron component, of the OpenStack cloud platform is responsible for providing two-layers or three-layers network for virtual machines. Services contained in the neutron component include a neutron-server service, a neutron-database service, and a neutron-sriov-agent service, etc. The control node (corresponding to the first server) provides the neutron-server service and the neutron-database service. The computing node (corresponding to the second server) provides the neutron-sriov-agent service. In FIG. 3, an agent service represents the neutron-sriov-agent service, a core service represents the neutron-server service, and a database service represents the neutron-database service. The following illustrates three servers respectively.

[49] Neutron-server service: the core service of the OpenStack cloud platform system. The

service may be configured to receive a bandwidth update request, further configured to synchronize an updated network bandwidth value (corresponding to a second bandwidth) to a neutron database, and further configured to transmit a remote procedure call (RPC) request to invoke a specific agent, e.g., neuron-sriov-agent, to update a bandwidth of the SR-IOV network card of the virtual machine (e.g., the service node).

[50] Neutron-database service: the database service of the OpenStack cloud platform system, which is configured to save the updated network bandwidth and ensure synchronization of all network-related data.

[51] Neutron-sriov-agent service: the agent service of the SR-IOV type network of the Openstack cloud platform system, which may be configured to modify the network bandwidth of the SR-IOV network card of the server node in the distributed training cluster.

[52] The following illustrates operations performed by each node when the method for network bandwidth adjustment provided in implementations of the disclosure is applied to the distributed training platform system in FIG. 3. FIG. 4 is a flow chart illustrating a method for network bandwidth adjustment provided in implementations of the disclosure. As illustrated in FIG. 4, the method includes the following.

[53] At 401, a scheduler node runs a startup script to start a training task.

[54] In one example, an instruction format of starting training is as follows: [run_task work_ip1 work_ip2 server_ip1 server_ip2 timeout mult_size]. The instruction format represents that there are two work nodes (e.g., work_ip1 and work_ip2) and two server nodes (e.g., server_ip1 and server_ip2). Timeout represents a maximum threshold (corresponding to a first time threshold) that a present iteration duration exceeds a previous average iteration duration, and mult_size represents times by which the bandwidth of each of all present server nodes is expanded. It can be understood that after the scheduler node runs the startup script to start the training task, a work node obtains a parameter from a service node to execute the training task. In one example, the distributed training cluster includes multiple work nodes, where each work node executes a part of the training task. Each work node obtains parameters from the server node and pushes a gradient obtained through training iteration to the service node. The training task in the disclosure may be a deep learning training task.

[55] At 402, the scheduler node monitors a first duration spent in completing a N^{th} training iteration during execution of the training task by the work node.

[56] In one example, after starting the script, the scheduler node can always poll a duration of each training iteration during execution of the training task by each work node, and accumulate the duration of each training iteration to calculate an average (corresponding to a second duration) of durations of multiple previous training iterations performed by each work node. That is, the scheduler node can monitor a time length spent on each training iteration by the work node. In some implementations, the scheduler may monitor and record a duration spent on each training iteration by each work node to obtain historical iteration duration records (also known as historical iteration duration information) of each work node. Assuming that the scheduler node monitors a first duration spent in completing a N^{th} training iteration during execution of the training task by a work node, the first duration can be recorded to the historical iteration duration records of the work node, and thus the historical iteration duration records include durations from a duration spent in completing a first training iteration to the duration spent in completing the N^{th} training iteration by the work node.

[57] At 403, the scheduler node transmits a bandwidth obtaining request to a control node on condition that the scheduler node determines that a time length spent in completing the N^{th} training iteration during execution of the training task by the work node is timed out.

[58] Optionally, the bandwidth obtaining request is used for obtaining a present bandwidth of each service node. Specifically, the scheduler node transmits the bandwidth obtaining request to a network core service (e.g., neutron-server) in an OpenStack cloud platform in the control node. That is, the network core service (e.g., neutron-server) in the OpenStack cloud platform can obtain the bandwidth obtaining request. For example, there are two service nodes in the distributed training platform system of FIG. 3, and the bandwidth obtaining request is used for querying for a network bandwidth value of each of the two service nodes. Various manners can be used to determine that the time length spent in completing the N^{th} training iteration during execution of the training task by the work node is timed out. In a first manner, the scheduler node may calculate an average of durations from a duration spent in completing a first training iteration to a duration spent in completing a N^{th} training iteration by the work node to obtain a second duration, e.g., an average iteration duration. On condition that a difference between the first duration and the second duration is not less than a first time threshold (corresponding to timeout), determine that the time length spent in completing the N^{th} training iteration by the work node is timed out. The first duration is longer than the second duration. The second duration is

the average iteration duration spent in completing at least one historical training iteration during execution of the training task by the work node. For example, N equals 5, and the second duration is an average of durations from a duration spent in completing a first training iteration to a duration spent in completing a fourth training iteration by the work node. In a second manner, the scheduler node obtains a maximum duration among durations from a duration spent in completing a first training iteration to a duration spent in completing a $(N-1)^{\text{th}}$ training iteration by the work node to obtain a third duration, where N is an integer greater than 1. On condition that a difference between the first duration and the third duration is not less than a second time threshold (corresponding to timeout), determine that the time length spent in completing the N^{th} training iteration by the work node is timed out. The first duration is longer than the third duration. A maximum timeout time threshold (e.g., timeout) may be preset by a user. In a second manner, the scheduler node may calculate a fourth duration spent in consecutively completing K training iterations by the work node, where K is an integer greater than 1. A fifth duration is obtained according to a duration of at least one training iteration. The fifth duration is an average duration spent in consecutively completing K historical training iterations by the work node. On condition that a difference between the fourth duration and the fifth duration is not less than a third time threshold (corresponding to timeout), determine that the time length spent in completing the at least one training iteration during execution of the training task by the work node is timed out. The fourth duration is longer than the fifth duration. The first time threshold, the second time threshold, and the third time threshold are all preset by the user.

[59] In some examples, for different training tasks, different time thresholds can be configured in advance. In some examples, the startup script further includes a parameter, where the parameter indicates whether timeout corresponds to the first time threshold, the second time threshold, or the third time threshold, so that the scheduler node can use a corresponding manner to determine whether the work node is timed out. In some examples, timeout corresponding to the first time threshold, timeout corresponding to the second time threshold, and timeout corresponding to the third time threshold are all notified to the scheduler node, and then which of the above manners are used to determine whether the work node is timed out can be configured in the script or determined by the scheduler node. Optionally, any of the above three manners can be used to determine whether the work node is timed out. Optionally, several manners among the above three manners can be separately used to determine whether the work node is timed out,

and as long as the work node is determined to be timed out in one of the several manners, it can be determined that the work node is timed out. Optionally, multiple manners among the above three manners can be separately used to determine whether the work node is timed out, and when the work node is determined to be timed out in at least two of the multiple manners, it can be determined that the work node is timed out. The disclosure is not limited thereto.

[60] Upon determining that the time length spent in completing the at least one training iteration during execution of the training task by the work node is timed out, the scheduler node can determine that a network bandwidth of the service node is insufficient. Once the startup script run by the scheduler node captures this situation, a bandwidth obtaining request may be transmitted to the network core service (e.g., neutron-server) in the OpenStack cloud platform in the control node to query for a network bandwidth value of each service node. Thereafter, the scheduler node transmits a request to the neutron-server to update the network bandwidth value of the SR-IOV network card of each service node. Here, an updated network bandwidth value is `mult_size` times the original bandwidth value, where `mult_size` is a real number greater than 1.

[61] At 404, the scheduler node obtains a present bandwidth of each service node.

[62] Optionally, the scheduler node receives the present bandwidth of each service node transmitted by the network core service (e.g., neutron-server).

[63] At 405, the scheduler node transmits a bandwidth update request to the control node.

[64] The bandwidth update request is used for requesting to update the bandwidth of each service node. As an example, the scheduler node transmits a bandwidth update request to the network core service (e.g., neutron-server) in the OpenStack cloud platform in the control node. In one example, a present bandwidth of a service node is a first bandwidth, and the bandwidth update request is used for requesting the control node to update the bandwidth of the service node to a second bandwidth. Alternatively, before executing operations at 405, the scheduler node can perform following operations. Calculate a product of the present bandwidth of each service node and `mult_size` to obtain an updated bandwidth of each service node. A bandwidth update request is generated according to the updated bandwidth of each service node. In other words, the bandwidth update request includes the updated bandwidth of each service node. In some implementations, the scheduler node obtains the first bandwidth of a service node, and determines the second bandwidth based on the first bandwidth and a preset bandwidth adjustment range included in the script.

[65] At 406, the network core service (e.g., neutron-server) provided by the OpenStack cloud platform updates a new network bandwidth value of each service node to a database.

[66] The new network bandwidth value refers to the updated bandwidth of each service node.

[67] At 407, the network core service (e.g., neutron-server) provided by the OpenStack cloud platform transmits a RPC request to a neutron-sriov-agent service on a computing node.

[68] Optionally, the RPC request (corresponding to an update bandwidth instruction) is used for requesting the neuron-sriov-agent to update the bandwidth of the SR-IOV network card of the virtual machine (e.g., the service node).

[69] At 408, the neutron-sriov-agent service on the computing node updates the bandwidth of each service node.

[70] As an example, after receiving the RPC request (corresponding to the update bandwidth instruction), the neutron-sriov-agent service on the computing node may immediately invoke an ip link set instruction to sequentially update the network bandwidth of the SR-IOV network card of each service node. It can be understood that the updated bandwidth of each service node is the same as that of each server indicated in the bandwidth update request transmitted by the scheduler node.

[71] At 409, the work node continues executing the training task until the training task is completed.

[72] In implementations of the disclosure, the network bandwidth of the parameter server in the distributed training cluster can be dynamically adjusted in real time without manual operation, which can avoid timeout of the iterative process of the distributed training task caused by the network bandwidth of the parameter server in the distributed training cluster, thereby ensuring smooth completion of the deep learning task.

[73] FIG. 5 is a flow chart illustrating another method for network bandwidth adjustment provided in implementations of the disclosure. As illustrated in FIG. 5, the method includes the following.

[74] At 501, a time length spent in completing at least one training iteration during execution of a training task by a work node is obtained.

[75] In some implementations, implementations of the disclosure are performed by a second server. The second server runs a first virtual machine, a second virtual machine, and a

third virtual machine. The second virtual machine is the above work node and the third virtual machine is the above service node. The second server may be a server or a server cluster. In implementations, determine that the time length spent in completing the at least one training iteration during execution of the training task by the work node is timed out as follows. The first virtual machine (corresponding to the scheduler node) determines that the time length spent in completing the at least one training iteration during execution of the training task by the work node (corresponding to the second virtual machine) is timed out.

[76] In some implementations, implementations of the disclosure are performed by a second server (corresponding to the scheduler node). The work node and the service node are both physical nodes. Alternatively, one of the work node and the service node is a virtual machine running on a third server, and the other of the work node and the service node is a physical node or a virtual machine running on a fourth server. The virtual machine is a simulator of a computer system. The virtual machine refers to a complete computer system which is simulated through software, has complete hardware system functions, and runs in a completely isolated environment, and can provide functionality of a physical computer. That is to say, compared to other devices, a virtual machine is a physical computer, e.g., a physical node. It can be understood that regardless of whether the work node, the service node, and the scheduler node are physical nodes or virtual machines, the scheduler node can perform the method in FIG. 5 to adjust a bandwidth of the service node.

[77] At 502, a bandwidth update request is transmitted to a first server if the time length spent on the at least one training iteration is timed out.

[78] The bandwidth update request is used for requesting the first server to update the bandwidth of the service node, and the service node stores data for the training task.

[79] In some implementations, after the second server transmits a bandwidth update request to the first server, the method further includes the following. The second server receives an update bandwidth instruction from the first server. The second server updates the bandwidth of the service node from a first bandwidth to a second bandwidth according to the update bandwidth instruction. In one example, after receiving the update bandwidth instruction transmitted by the neutron-server in the first server (corresponding to the control node), the neutron-sriov-agent in the second server invokes an ip link set instruction to sequentially update the network bandwidth of the SR-IOV network card of each service node, for example, expand

the network bandwidth of the SR-IOV network card of each service node by `mult_size` times.

[80] In implementations of the disclosure, on condition that the time length spent in completing the at least one training iteration during execution of the training task by the work node is timed out, the bandwidth update request is transmitted to the first server so as to update the bandwidth of the service node, which can effectively solve the problem of the insufficient network bandwidth of the parameter server and avoid the timeout of the training by the work node.

[81] The following illustrates in detail how to determine that a time length spent in completing a N^{th} training iteration during execution of the training task by the work node is timed out.

[82] In an alternative implementation, before executing operations at 501, the second server can obtain a first duration spent in completing the N^{th} training iteration by the work node. The second server determines that the time length spent in completing the N^{th} training iteration during execution of the training task by the work node is timed out as follows. The scheduler node determines that the time length spent in completing the N^{th} training iteration by the work node is timed out, e.g., a time length spent in completing N training iterations by the work node is timed out, according to the first duration and historical iteration duration records of the work node. The historical iteration duration records include a duration spent in completing at least one training iteration during execution of the training task by the work node. The scheduler node may be a second server or a first virtual machine run on the second server.

[83] As an example, the historical iteration duration records include durations from a duration spent in completing a first training iteration to a duration spent in completing the N^{th} training iteration during execution of the training task by the work node. The scheduler node calculates an average of the durations from the duration spent in completing the first training iteration to the duration spent in completing the N^{th} training iteration by the work node to obtain a second duration. The scheduler node determines that the time length spent in completing the N^{th} training iteration by the work node is timed out, on condition that a difference between the first duration and the second duration is not less than a first time threshold. The first duration is longer than the second duration.

[84] As another example, the historical iteration duration records include the durations from the duration spent in completing the first training iteration to the duration spent in

completing the N^{th} training iteration during execution of the training task by the work node. The scheduler node obtains a maximum duration among durations from the duration spent in completing the first training iteration to a duration spent in completing a $(N-1)^{\text{th}}$ training iteration by the work node, to obtain a third duration, where N is an integer greater than 1. The scheduler node determines that the time length spent in completing the N^{th} training iteration by the work node is timed out, on condition that a difference between the first duration and the third duration is not less than a second time threshold. The first duration is longer than the third duration.

[85] In implementations, based on the first duration and the historical iteration duration records, it can accurately and quickly determine whether the time length spent in completing the N^{th} training iteration by the work node is timed out.

[86] In an alternative implementation, before executing operations at 501, the second server can obtain: a fourth duration spent in consecutively completing K training iterations by the work node, where K is an integer greater than 1; obtain a fifth duration according to a duration of at least one training iteration, where the fifth duration is an average duration spent in consecutively completing K historical training iterations by the work node. Determine that the time length spent in completing the at least one training iteration during execution of the training task by the work node is timed out as follows. On condition that a difference between the fourth duration and the fifth duration is not less than a third time threshold, determine that the time length spent in completing the at least one training iteration during execution of the training task by the work node is timed out. The fourth duration is longer than the fifth duration. In some examples, assuming that $K=3$ and the work node has executed 12 training iterations for the training task, to determine whether a time length spent in consecutively completing the K training iterations by the work node is timed out, fourth duration D spent in consecutively completing training iterations from a 10th training iteration to a 12th training iteration by the work node is obtained, and an average duration spent in consecutively completing 3 historical training iterations by the work node is obtained according to duration A spent in consecutively completing training iterations from a first training iteration to a third training iteration by the work node, duration B spent in consecutively completing training iterations from a fourth training iteration to a sixth training iteration by the work node, duration C spent in consecutively completing training iterations from a seventh training iteration to a ninth training iteration by the work node, and fourth duration D . A fifth duration may be obtained by dividing a sum of A , B ,

C, and D by 4. If the difference between the fourth duration and the fifth duration is greater than or equal to the third time threshold, determine that the time length spent in consecutively completing the K training iterations by the work node is timed out.

[87] In implementations, it can accurately and quickly determine whether the time length spent on the consecutive K training iterations by the work node is timed out.

[88] FIG. 6 is a flow chart illustrating yet another method for network bandwidth adjustment provided in implementations of the disclosure. The method in FIG. 6 is a further refinement and improvement of the method in FIG. 5. The method in FIG. 6 is applied to the distributed training platform system in FIG. 3. As illustrated in FIG. 6, the method may include the following.

[89] At 601, a scheduler node runs a startup script. The script is used for obtaining a time length spent in completing at least one training iteration during execution of a training task by a work node.

[90] The scheduler node may be a first virtual machine run in a second server. Optionally, the scheduler node runs the startup script to start algorithm training, queries for a training iteration duration of each work node, and determines whether the iteration duration of each work node is timed out. The script includes at least one of information used for determining that the time length spent on the at least one training iteration is timed out or a preset bandwidth adjustment range.

[91] At 602, the scheduler node obtains a first duration spent in completing a N^{th} training iteration by a target work node.

[92] The target work node may be any of the work nodes in FIG. 2 or FIG. 3. In a practical process, the scheduler node can obtain a duration spent on each training iteration by each of one or more work nodes. In implementations of the disclosure, the target work node is taken as an example to illustrate processes of the method for adjusting a bandwidth of a service node.

[93] At 603, the scheduler node calculates an average of durations from a duration spent in completing a first training iteration to a duration in completing a N^{th} training iteration by the target work node to obtain a second duration.

[94] At 604, the scheduler node determines whether a difference between the first duration and the second duration is not less than a first time threshold (corresponding to timeout).

[95] If yes, proceed to operations at 605; otherwise, proceed to operations at 607.

Assuming that the first duration is 12ms, the second duration is 6ms, the first time threshold is 5ms, and thus the difference between the first duration and the second duration is 6ms. Therefore, the difference between the first duration and the second duration is not less than the first time threshold.

[96] At 605, the scheduler node obtains a present bandwidth of each service node.

[97] In one example, the service node may be a virtual machine running in a second server e.g., the service node in FIG. 3. In some implementations, optionally, the scheduler node transmits a bandwidth obtaining request to a first server, where the bandwidth obtaining request is used for obtaining a present bandwidth of each service node. The scheduler node receives the present bandwidth of each service node transmitted by the network core service (e.g., neutron-server).

[98] At 606, the second server updates the bandwidth of each service node through the neutron-sriov-agent service.

[99] For implementations in operations at 606, reference may be made to implementations in which the neutron-sriov-agent updates the bandwidth of each service node in FIG. 4, which are not repeated herein. The second server may be a computing node.

[100] At 607, the scheduler node determines whether the training is finished.

[101] If yes, proceed to operations at 608; otherwise, proceed to operations at 602.

[102] At 608, the training task is finished.

[103] The following illustrates a device for network bandwidth adjustment that can implement the method for network bandwidth adjustment provided in the foregoing implementations.

[104] FIG. 7 is a schematic structural diagram illustrating a device for network bandwidth adjustment provided in implementations of the disclosure. As illustrated in FIG. 7, the device for network bandwidth adjustment includes an obtaining unit 701, a determining unit 702, and a transmitting unit 703.

[105] The obtaining unit 701 is configured to obtain a time length spent in completing at least one training iteration during execution of a training task by a work node.

[106] The determining unit 702 is configured to determine that the time length spent on the at least one training iteration is timed out.

[107] The transmitting unit 703 is configured to transmit a bandwidth update request to a

first server in response to determining that the time length spent on the at least one training iteration is timed out. The bandwidth update request is used for requesting the first server to update a bandwidth of a service node, and the service node stores data for the training task.

[108] In an alternative implementation, the at least one training iteration is embodied as N training iterations, and the determining unit 702 is specifically configured to: determine that the time length spent on the at least one training iteration is timed out, according to a first duration of the time length spent on the at least one training iteration and historical iteration duration information of executing the training task by the work node. The first duration is a duration spent in completing a N^{th} training iteration in the N training iterations during execution of the training task by the work node.

[109] In an alternative implementation, the determining unit 702 is specifically configured to: obtain a second duration according to a duration spent in completing at least one historical training iteration during execution of the training task by the work node, where the second duration is an average duration spent in completing the at least one historical training iteration during execution of the training task by the work node; determine that the time length spent on the at least one training iteration is timed out, on condition that a difference between the first duration and the second duration is greater than or equal to a first time threshold.

[110] In an alternative implementation, the determining unit is specifically configured to: obtain a maximum duration among durations from a duration spent in completing a first training iteration in the N training iterations to a duration spent in completing a $(N-1)^{\text{th}}$ training iteration in the N training iterations by the work node, according to the historical iteration duration information of executing the training task by the work node; determine the maximum duration as a third duration; determine that the time length spent on the at least one training iteration is timed out, on condition that a difference between the first duration and the third duration is greater than or equal to a second time threshold.

[111] In an alternative implementation, the at least one training iteration is embodied as K training iterations that are consecutive, and the determining unit 702 is specifically configured to: obtain a fourth duration spent in consecutively completing the K training iterations by the work node; obtain an average duration spent in consecutively completing K historical training iterations of the training task by the work node, and determine the average duration as a fifth duration; determine that the time length spent on the at least one training iteration is timed out,

on condition that a difference between the fourth duration and the fifth duration is greater than or equal to a third time threshold.

[112] In an alternative implementation, the work node and the service node are both physical nodes. Alternatively, a method for network bandwidth adjustment is applied to a second server, where one of the work node and the service node is a virtual machine running on a third server, and the other of the work node and the service node is a physical node or a virtual machine running on a fourth server.

[113] In an alternative implementation, the method for network bandwidth adjustment is applied to a first virtual machine on a second server, and the second server further runs a second virtual machine and a third virtual machine. The second virtual machine is the work node, and the third virtual machine is the service node.

[114] In an alternative implementation, the device further includes a running unit 704 configured to run a training task startup script before the obtaining unit obtains the time length spent on the at least one training iteration. The training task startup script is used for obtaining the time length spent in completing the at least one training iteration during execution of the training task by the work node.

[115] In an alternative implementation, the training task startup script includes at least one of information used for determining that the time length spent on the at least one training iteration is timed out or a preset bandwidth adjustment range.

[116] In an alternative implementation, the obtaining unit 701 is further configured to obtain a present first bandwidth of the service node. The determining unit 702 is further configured to determine, according to the first bandwidth and a preset bandwidth adjustment range, to adjust the bandwidth of the service node to a second bandwidth. The bandwidth update request includes the second bandwidth, and the second bandwidth is greater than the first bandwidth.

[117] It can be understood that the above division of each unit of the device for network bandwidth adjustment is only a logical function division, which can be fully or partially integrated into a physical entity or physically separated in actual implementation. For example, the above units can be separate processing elements or integrated in the same chip. In addition, the above units can also be stored in the memory elements of the controller in the form of program codes, which can be invoked and executed by a processing element of the processor to

perform the functions of the above units. Furthermore, the units can be integrated or implemented separately. The processing element herein may be an integrated circuit chip with signal processing capability. In implementations, steps of the method or the above units may be implemented by integrated logic circuit of hardware in the processor element or by instructions in the form of software. The processing element may be a general-purpose processor, such as a central processing unit (CPU), or one or more integrated circuits configured to implement the above method, such as one or more application-specific integrated circuits (ASIC), one or more digital signal processors (DSP), or one or more field-programmable gate arrays (FPGA), etc.

[118] FIG. 8 is a schematic structural diagram illustrating a server provided in implementations of the disclosure. The server 800 may vary considerably depending on configuration or performance and may include one or more CPUs 822 (e.g. one or more processors) and memories 832, and one or more storage media 830 (e.g. one or more mass storage devices) storing applications 842 or data 844. The memory 832 and the storage medium 830 may be transient storage or persistent storage. The programs stored in the storage medium 830 may include one or more modules (not illustrated). Each module is configured to perform operations on a series of instructions in the server. Furthermore, the central processor 822 may be configured to communicate with the storage medium 830 to perform a series of instruction operations in the storage medium 830 on the server 800. The server 800 may be a device for network bandwidth adjustment provided in implementations of the disclosure.

[119] The server 800 may further include one or more power sources 826, one or more wired or wireless network interfaces 850, one or more input and output interfaces 858, and/or one or more operating systems 841, such as Windows Server™, Mac OS X™, Unix™, Linux™, FreeBSD™, and the like.

[120] The steps performed by the second server in the above implementations may be based on structure of the server illustrated in FIG. 8. Specifically, the CPU 8 can realize the functions of each unit in FIG. 7.

[121] Implementations of the disclosure provide a computer-readable storage medium. The computer-readable storage medium stores computer programs. The computer programs are executed by a processor to perform: transmitting a bandwidth update request to a first server in response to determining that a time length spent in completing at least one training iteration during execution of a training task by a work node is timed out, where the bandwidth update

request is used for requesting the first server to update a bandwidth of a service node, and the service node is a node storing data required for execution of the training iteration task by the work node. The computer-readable storage medium includes a non-transitory computer-readable storage medium.

[122] Implementations of the disclosure provide a computer program product including instructions which, when executed on a computer, cause the computer to perform the method for network bandwidth adjustment provided in foregoing implementations.

[123] The foregoing merely illustrates some implementations of the disclosure, but the scope of protection of the disclosure is not limited thereto. Any skilled person in the technical field can easily think of various equivalent modifications or substitutions within the technical scope disclosed in the disclosure, and these modifications or substitutions should be covered within the scope of protection of the disclosure. Therefore, the scope of protection of the disclosure shall be subject to the scope of protection of the claims.

What is claimed is:

1. A method for network bandwidth adjustment, comprising:
obtaining a time length spent in completing at least one training iteration during execution of a training task by a work node; and
transmitting a bandwidth update request to a first server in response to determining that the time length spent on the at least one training iteration is timed out, wherein the bandwidth update request is used for requesting the first server to update a bandwidth of a service node, and the service node stores data for the training task.
2. The method of claim 1, wherein the at least one training iteration is embodied as N training iterations, and determining that the time length spent on the at least one training iteration is timed out comprises:
determining that the time length spent on the at least one training iteration is timed out, according to a first duration of the time length spent on the at least one training iteration and historical iteration duration information of executing the training task by the work node, wherein the first duration is a duration spent in completing a N^{th} training iteration in the N training iterations during execution of the training task by the work node.
3. The method of claim 2, wherein determining that the time length spent on the at least one training iteration is timed out, according to the first duration of the time length spent on the at least one training iteration and the historical iteration duration information of executing the training task by the work node comprises:
obtaining a second duration according to a duration spent in completing at least one historical training iteration during execution of the training task by the work node, wherein the second duration is an average duration spent in completing the at least one historical training iteration during execution of the training task by the work node; and
determining that the time length spent on the at least one training iteration is timed out, on condition that a difference between the first duration and the second duration is greater than or equal to a first time threshold.
4. The method of claim 2, wherein determining that the time length spent on the at least one

training iteration is timed out, according to the first duration of the time length spent on the at least one training iteration and the historical iteration duration information of executing the training task by the work node comprises:

obtaining a maximum duration among durations from a duration spent in completing a first training iteration in the N training iterations to a duration spent in completing a $(N-1)^{\text{th}}$ training iteration in the N training iterations by the work node, according to the historical iteration duration information of executing the training task by the work node;

determining the maximum duration as a third duration; and

determining that the time length spent on the at least one training iteration is timed out, on condition that a difference between the first duration and the third duration is greater than or equal to a second time threshold.

5. The method of claim 1, wherein the at least one training iteration is embodied as K training iterations that are consecutive, and determining that the time length spent on the at least one training iteration is timed out comprises:

obtaining a fourth duration spent in consecutively completing the K training iterations by the work node;

obtaining an average duration spent in consecutively completing K historical training iterations of the training task by the work node, and determining the average duration as a fifth duration; and

determining that the time length spent on the at least one training iteration is timed out, on condition that a difference between the fourth duration and the fifth duration is greater than or equal to a third time threshold.

6. The method of any of claims 1 to 5, wherein

the work node and the service node are both physical nodes; or

the method for network bandwidth adjustment is applied to a second server, wherein one of the work node and the service node is a virtual machine running on a third server, and the other of the work node and the service node is a physical node or a virtual machine running on a fourth server.

7. The method of any of claims 1 to 5, wherein the method for network bandwidth adjustment is applied to a first virtual machine on a second server, and the second server further runs a second virtual machine and a third virtual machine, wherein the second virtual machine is the work node, and the third virtual machine is the service node.
8. The method of any of claims 1 to 7, further comprising:
prior to obtaining the time length spent in completing the at least one training iteration during execution of the training task by the work node,
running a training task startup script, wherein the training task startup script is used for obtaining the time length spent in completing the at least one training iteration during execution of the training task by the work node.
9. The method of claim 8, wherein the training task startup script comprises at least one of information used for determining that the time length spent on the at least one training iteration is timed out or a preset bandwidth adjustment range.
10. The method of any of claims 1 to 9, further comprising:
obtaining a present first bandwidth of the service node; and
determining, according to the first bandwidth and a preset bandwidth adjustment range, to adjust the bandwidth of the service node to a second bandwidth, wherein the bandwidth update request comprises the second bandwidth, and the second bandwidth is greater than the first bandwidth.
11. A device for network bandwidth adjustment, comprising:
an obtaining unit configured to obtain a time length spent in completing at least one training iteration during execution of a training task by a work node;
a determining unit configured to determine that the time length spent on the at least one training iteration is timed out; and
a transmitting unit configured to transmit a bandwidth update request to a first server in response to determining that the time length spent on the at least one training iteration is timed out, wherein the bandwidth update request is used for requesting the first server to update a

bandwidth of a service node, and the service node stores data for the training task.

12. The device of claim 11, wherein the at least one training iteration is embodied as N training iterations, and the determining unit is configured to:

determine that the time length spent on the at least one training iteration is timed out, according to a first duration of the time length spent on the at least one training iteration and historical iteration duration information of executing the training task by the work node, wherein the first duration is a duration spent in completing a N^{th} training iteration in the N training iterations during execution of the training task by the work node.

13. The device of claim 12, wherein the determining unit is configured to:

obtain a second duration according to a duration spent in completing at least one historical training iteration during execution of the training task by the work node, wherein the second duration is an average duration spent in completing the at least one historical training iteration during execution of the training task by the work node; and

determine that the time length spent on the at least one training iteration is timed out, on condition that a difference between the first duration and the second duration is greater than or equal to a first time threshold.

14. The device of claim 12, wherein the determining unit is configured to:

obtain a maximum duration among durations from a duration spent in completing a first training iteration in the N training iterations to a duration spent in completing a $(N-1)^{\text{th}}$ training iteration in the N training iterations by the work node, according to the historical iteration duration information of executing the training task by the work node;

determine the maximum duration as a third duration; and

determine that the time length spent on the at least one training iteration is timed out, on condition that a difference between the first duration and the third duration is greater than or equal to a second time threshold.

15. The device of claim 11, wherein the at least one training iteration is embodied as K training iterations that are consecutive, and the determining unit is configured to:

obtain a fourth duration spent in consecutively completing the K training iterations by the work node;

obtain an average duration spent in consecutively completing K historical training iterations of the training task by the work node, and determine the average duration as a fifth duration; and

determine that the time length spent on the at least one training iteration is timed out, on condition that a difference between the fourth duration and the fifth duration is greater than or equal to a third time threshold.

16. The device of any of claims 11 to 15, wherein

the work node and the service node are both physical nodes; or

a method for network bandwidth adjustment is applied to a second server, wherein one of the work node and the service node is a virtual machine running on a third server, and the other of the work node and the service node is a physical node or a virtual machine running on a fourth server.

17. The device of any of claims 11 to 15, wherein a method for network bandwidth adjustment is applied to a first virtual machine on a second server, and the second server further runs a second virtual machine and a third virtual machine, wherein the second virtual machine is the work node, and the third virtual machine is the service node.

18. The device of any of claims 11 to 17, further comprising:

a running unit configured to run a training task startup script before the obtaining unit obtains the time length spent on the at least one training iteration, wherein the training task startup script is used for obtaining the time length spent in completing the at least one training iteration during execution of the training task by the work node.

19. The device of claim 18, wherein the training task startup script comprises at least one of information used for determining that the time length spent on the at least one training iteration is timed out or a preset bandwidth adjustment range.

20. The device of any of claims 11 to 19, wherein
the obtaining unit is further configured to obtain a present first bandwidth of the service node; and
the determining unit is further configured to determine, according to the first bandwidth and a preset bandwidth adjustment range, to adjust the bandwidth of the service node to a second bandwidth, wherein the bandwidth update request comprises the second bandwidth, and the second bandwidth is greater than the first bandwidth.
21. A computer-readable storage medium, wherein the computer-readable storage medium stores computer programs and the computer programs comprise program instructions, wherein the program instructions, when executed by a processor, cause the processor to perform the method of any of claims 1 to 10.
22. An electronic device, comprising:
a memory configured to store programs;
a processor configured to execute the programs stored in the memory, wherein when the programs are executed, the processor is configured to perform the method of any of claims 1 to 10.