MODEL TRAINING METHOD AND ELECTRONIC DEVICE

TECHNICAL FIELD

[0001] This application relates to the field of computer technologies, and in particular, to a model training method and an electronic device.

BACKGROUND

[0002] The vast potential of large artificial intelligence (Artificial Intelligence, AI) models makes vendors research and use the large models. Training large AI models with hundreds of billions of parameters typically requires distributed cluster training using thousands of graphics processing unit (graphics processing unit, GPU) cards. In such scenarios, frequent GPU card failures may cause a loss of model data that is currently being trained. The loss of model data results in the interruption of model training and necessitates re-training, causing a waste of high-cost GPU card computational power. Therefore, a method is needed to restore a model training state upon the loss of model data, so as to avoid a waste of GPU card computational power caused by GPU card failures, thereby improving model training efficiency.

SUMMARY

[0003] This application provides a model training method and apparatus, and a compute device cluster, and this application further provides a computer program product and a computer-readable storage medium, to avoid problems that model training is interrupted and GPU computational power is wasted due to a GPU card failure.

[0004] The following technical solutions are used in embodiments of this application.

[0005] According to a first aspect, this application provides a model training method. The method is applied to a training cluster, the training cluster includes a plurality of model training groups, the plurality of model training groups include a first model training group and a second model training group, the first model training group and the second model training group are configured to train a neural network model in parallel in a data parallel manner, the first model training group and the second model training group include a same node structure, the first model training group includes a first training node, the second model training group includes a second training node, the first training node corresponds to the second training node based on the node structure, and the method includes:

generating checkpoint data after one round of training ends, where the checkpoint data is checkpoint data of the first training node and the second training node;

storing, by the first training node, the checkpoint data in a memory of the first training node, and storing, by the second training node, the checkpoint data in a memory of the second training node; and

if the first training node is faulty in a process of training a model, obtaining, by the first training node, the checkpoint data from the memory of the second training node, to perform data recovery on the first training node.

[0006] According to the model training method of the first aspect, memories of training nodes are used to back up checkpoint data of the training nodes. In a process of backing up the checkpoint data of the training nodes, the training nodes do not need to communicate with each other, to greatly reduce consumption of communication resources.

[0007] According to the model training method of the first aspect, the memories of the training nodes are used to back up the checkpoint data of the training nodes, and checkpoint data of a faulty training node is retransmitted and recovered by using a feature of redundancy backup of a multi-model group. A remote storage service does not need to participate in a checkpoint data backup and recovery process, and the remote storage service is removed from a critical path, to lower performance and bandwidth requirements for remote storage, thereby significantly reducing costs of an entire system.

[0008] In an implementation of the first aspect, the method further includes:

[0009] The second training node that is not faulty performs data recovery on the second training node by using checkpoint data stored in the memory of the second training node.

[0010] In an implementation of the first aspect, the generating the checkpoint data after one round of training ends includes:

when one round of training ends, performing average calculation on model parameters of all training nodes in the training cluster, and using a calculation result as checkpoint data of the round of training.

[0011] In an implementation of the first aspect, the method further includes:

[0012] The first training node downloads initial checkpoint data; and

the first training node sends the initial checkpoint data to the second training node.

[0013] According to the method in the foregoing implementation, the initial checkpoint data is downloaded to one model training group, and the initial checkpoint data is sent to another model training group through the model training group, to lower a bandwidth requirement for the initial checkpoint data, and improve efficiency of downloading the initial checkpoint data.

[0014] In an implementation of the first aspect, the method further includes:

[0015] The first model training group uploads checkpoint data stored in a memory of each training node in the first model training group to the remote storage service.

[0016] According to the foregoing implementation, in a solution in which remote storage is removed from a critical path, cluster memory data is asynchronously persisted to a remote end, to reduce dependence on performance of the remote end. In addition, according to the method in this embodiment of this application, temporary storage of a CKPT by the memory is distinguished from persistent storage of the CKPT by the remote storage service, and space consumption of the remote storage is not increased.

[0017] In an implementation of the first aspect, in a process of uploading checkpoint data by the first model training group, deleting the checkpoint data stored in the memory of each training node in the first model training group is prohibited, and overwriting the stored checkpoint data with new checkpoint data is prohibited.

[0018] According to the method in the foregoing implementation, generation of the new checkpoint data can be prevented from interfering with uploading of the checkpoint data, to ensure that the checkpoint data can be successfully uploaded to the remote storage service.

[0019] In an implementation of the first aspect, in a process of uploading checkpoint data by the first model training group, storing new checkpoint data in a memory of a training node in the first model training group is stopped.

[0020] In an implementation of the first aspect, after the checkpoint data is uploaded by the first model training group, the new checkpoint data stored in a memory of a training node in the second model training group is synchronized to the memory of the training node in the first model training group.

[0021] According to the method in the foregoing implementation, generation of the new checkpoint data can be prevented from interfering with uploading of the checkpoint data, to ensure that the checkpoint data can be successfully uploaded to the remote storage service; and it may be ensured that versions of checkpoint data of a training node that uploads checkpoint data and a training node that does not upload checkpoint data are consistent.

[0022] According to a second aspect, this application provides a model training apparatus. The apparatus is used in a training cluster, the training cluster includes a plurality of model training groups, the plurality of model training groups include a first model training group and a second model training group, the first model training group and the second model training group are configured to train a neural network model in parallel in a data parallel manner, the first model training group and the second model training group include a same node structure, the first model training group includes a first training node, the second model training group includes a second training node, and the first training node corresponds to the second training node based on the node

structure; and the training cluster generates checkpoint data after one round of training ends, where the checkpoint data is checkpoint data of the first training node and the second training node; and

the apparatus includes:

a storage control module, configured to: instruct the first training node to store the checkpoint data in a memory of the first training node, and instruct the second training node to store the checkpoint data in a memory of the second training node; and

a recovery control module, configured to: if the first training node is faulty in a process of training a model, instruct the first training node to obtain the checkpoint data from the memory of the second training node, to perform data recovery on the first training node.

[0023] In an implementation of the second aspect, the recovery control module is further configured to:

instruct the second training node to perform data recovery on the second training node by using checkpoint data stored in the memory of the second training node.

[0024] In an implementation of the second aspect, when one round of training ends, the training cluster calculates an average of model parameters of all training nodes in the training cluster, and uses a calculation result as checkpoint data of the round of training.

[0025] In an implementation of the second aspect, the apparatus further includes an initialization module, and the initialization module is configured to:

instruct the first training node to download initial checkpoint data; and

instruct the first training node to broadcast the initial checkpoint data to the second training node.

[0026] In an implementation of the second aspect, the apparatus further includes a data upload module, and the data upload module is configured to:

instruct the first model training group to upload checkpoint data stored in a memory of each training node in the first model training group to a remote storage service.

[0027] In an implementation of the second aspect, the storage control module is further configured to:

in a process of uploading checkpoint data by the first model training group, prohibit deleting the checkpoint data stored in the memory of each training node in the first model training group, and prohibit overwriting the stored checkpoint data with new checkpoint data.

[0028] In an implementation of the second aspect, the storage control module is further configured to:

in a process of uploading checkpoint data by the first model training group, stop storing new checkpoint data in a memory of a training node in the first model training group.

[0029] In an implementation of the second aspect, the storage control module is further

configured to:

after the checkpoint data is uploaded by the first model training group, synchronize the new checkpoint data stored in a memory of a training node in the second model training group to the memory of the training node in the first model training group.

[0030] According to a third aspect, this application provides a compute device cluster, including at least one compute device. Each compute device includes a processor and a memory.

[0031] The processor of the at least one compute device is configured to execute instructions stored in the memory of the at least one compute device, to enable the compute device cluster to perform the method according to the first aspect.

[0032] According to a fourth aspect, this application provides a computer program product including instructions. When the instructions are run by a compute device cluster, the compute device cluster is enabled to perform the method according to the first aspect.

[0033] According to a fifth aspect, this application provides a computer-readable storage medium, including computer program instructions. When the computer program instructions are executed by a compute device cluster, the compute device cluster performs the method according to the first aspect.

BRIEF DESCRIPTION OF DRAWINGS

[0034] FIG. 1 is a diagram of reading/writing a CKPT in a model training cluster according to an embodiment of this application;

[0035] FIG. 2 is a diagram of reading/writing data in a direct remote storage according to an embodiment of this application;

[0036] FIG. 3 shows a process of reading/writing data in a distributed cache according to an embodiment of this application;

[0037] FIG. 4 shows a process of reading/writing CKPT data by using a memory according to an embodiment of this application;

[0038] FIG. 5 is a schematic flowchart of a model training method according to an embodiment of this application;

[0039] FIG. 6 is a diagram of a structure of a training cluster according to an embodiment of this application;

[0040] FIG. 7 is a schematic flowchart of a model training method according to an embodiment of this application;

[0041] FIG. 8 is a diagram of a model training apparatus according to an embodiment of this application;

[0042] FIG. 9 is a diagram of a training node deployment scenario according to an embodiment of this application;

[0043] FIG. 10 is a diagram of a training node deployment scenario according to an embodiment of this application;

[0044] FIG. 11 is a diagram of a compute device according to an embodiment of this application;

[0045] FIG. 12 is a diagram of a compute device cluster according to an embodiment of this application; and

[0046] FIG. 13 is a diagram of a network connection of a compute device cluster according to an embodiment of this application.

DESCRIPTION OF EMBODIMENTS

[0047] To make objectives, technical solutions, and advantages of this application clearer, the following clearly and completely describes the technical solutions in this application with reference to specific embodiments and accompanying drawings in this application. It is clear that the described embodiments are merely some but not all of embodiments of this application. All other embodiments obtained by a person of ordinary skill in the art based on embodiments of this application without creative efforts shall fall within the protection scope of this application.

[0048] Terms used in embodiments of this application are only used to explain specific embodiments of this application, but are not intended to limit this application.

[0049] To resolve problems that model training is interrupted and GPU computational power is wasted due to a GPU card failure, in a feasible technical solution, a manner of periodically storing a checkpoint (Checkpoint, CKPT) is used to avoid a loss of training data in a training process.

[0050] For example, a persistent storage service is provided on a cloud server, and model data is stored on the cloud server every three hours. If the GPU card failure occurs in the fourth hour, a training cluster can load the model data stored on the cloud server last time, and does not need to start training from scratch.

[0051] However, the storage and recovery of CKPT in a large-scale distributed training cluster has low performance and high costs. Specifically, the manner of periodically storing the checkpoint has a high requirement on network bandwidth and performance of remote storage. In the large-scale distributed training cluster, network bandwidth and performance of remote storage become a bottleneck of an entire system, and a long time overhead is introduced to restore a training state, causing a large waste of GPU card resources.

[0052] The distributed training cluster splits a large model and places the split parts on a plurality of nodes for parallel training. All nodes that form a complete model are called a model group.

[0053] For example, 512 nodes are used to train a 70-billion-parameter model. To improve parallel training efficiency of the model, the 512 nodes may be divided into 32 model groups to train 32 models in parallel. 16 nodes form a model group to train a single model.

[0054] For example, FIG. 1 is a diagram of reading/writing a CKPT in a model training cluster according to an embodiment of this application.

[0055] Any two of the 32 model groups are shown in a model (model) group 101 and a model (model) group 102 shown in FIG. 1.

[0056] For the 70-billion-parameter model, a CKPT size of a model is 1 TB. A single card cannot store such a model, so the model is divided into 128 parts and placed on 128 cards for training. The model (model) group 101 includes nodes (Node) 0 to 15, and the model (model) group 102 includes nodes (Node) 16 to 31. Each node contains eight GPU cards (GPUs 0 to 7).

[0057] When each round of training starts, the 32 model groups jointly load a same CKPT to ensure that the 32 model groups are in a same initialization state. That is, when each round of training starts, parameters of the model groups are the same.

[0058] In a training process, the 32 model groups use the eight GPU cards for pipeline parallelism, optimizer parallelism, and data parallelism.

[0059] When one round of training ends, training results of all model groups are summarized to obtain an average value as a CKPT result of the current round of training, and the CKPT result is synchronized to all model groups before the next round of training is performed.

[0060] After each round of training ends, the CKPT result of the round of training is uploaded to a remote object storage service (Object Storage Service, OBS).

[0061] When each round of training starts, 32 model groups need to obtain data from an OBS. Therefore, the OBS needs to provide 32 TB of CKPT data download traffic to ensure data service for the 32 model groups. If the download is required to be completed within 30 minutes, the OBS needs to provide a bandwidth of at least 32 TB / 30 min = 145.64 Gbps. In some application scenarios, the OBS needs to provide 16 TB of dataset download traffic. If the download is required to be completed within 30 minutes, the OBS needs to provide a bandwidth of at least 48 TB/30 min = 218.453 Gbps. This imposes a very high requirement on network bandwidth of remote storage. As download duration is shortened and a quantity of model parameters increases, a higher bandwidth requirement is imposed on the remote storage, so that costs of the remote storage are significantly increased.

[0062] On the other hand, when all nodes store and download CKPTs, GPU training cards are

in an idle state, causing a large waste of computational power. In addition, GPU cards are charged by time. Tens of thousands of GPU cards cost tens of thousands of CNY per minute. It should be noted that calculation results from the time when the checkpoint is stored to the time when a cluster is faulty are all lost. The waste of GPU computational power during this period is also a source of costs. Therefore, a model storage frequency greatly affects performance and costs of the training cluster.

[0063] In a feasible solution, a direct remote storage is used to improve performance of storing and reading a CKPT and reduce costs thereof.

[0064] FIG. 2 is a diagram of reading/writing data in a direct remote storage according to an embodiment of this application.

[0065] A most direct way to improve storage and reading of the CKPT is to improve performance of a remote storage service. Therefore, the performance of the remote storage service can be improved by improving network bandwidth of the remote storage service, deploying the remote storage service on a network closer to a training cluster, developing an efficient network file system, or the like.

[0066] As shown in FIG. 2, the remote storage service is placed on a network closer to a node (Node) in the training cluster. In this manner, training nodes may use high bandwidth and short network paths to store and read CKPTs. Representative technical solutions include Amazon S3, Huawei Cloud OBS, Huawei Cloud scalable file service (Scalable File Service, SFS), and the like. However, this manner does not have great advantages in performance and costs. This is because a large amount of traffic still needs to be transmitted through an external network, a network path is much longer than that in the cluster, and higher network bandwidth indicates higher costs in the solution.

[0067] In another feasible solution, a distributed cache is used to improve performance of storing and reading a CKPT and reduce costs thereof.

[0068] FIG. 3 shows a process of reading/writing data in a distributed cache according to an embodiment of this application.

[0069] The distributed cache places a path for storing and reading the CKPT in a cluster for model training.

[0070] As shown in FIG. 3, local solid-state drive (solid-state drive, SSD) disks of each node are connected to form a cache pool to expand a storage capacity. A CKPT is stored in the cache pool, and the complete CKPT is evenly distributed in the cache pool to ensure network load balancing in the cluster. In this way, when downloading CKPT data, cluster nodes preferentially read the CKPT data from the cache pool, so that network dependence on a remote storage service can be reduced through a network between the nodes. Representative solutions include JuiceFs,

Alluxio, and MemArts. However, performance of these solutions depends on an optimal prefetch algorithm to improve a hit rate of the cache pool and an optimal local affinity scheduling algorithm to reduce network latency in the cluster. The performance of these solutions can still be greatly improved.

[0071] In another feasible solution, a memory storage is used to improve performance of storing and reading a CKPT and reduce costs thereof.

[0072] FIG. 4 shows a process of reading/writing CKPT data by using a memory according to an embodiment of this application.

[0073] As shown in FIG. 4, storing and reading a CKPT by using a memory (Memory) involves directly storing the CKPT in the memory. High-speed memory storage and access improve performance in storing and accessing the CKPT and reduce dependency of cluster nodes on a network, thereby mitigating impact of network latency and bandwidth and significantly improving performance of a training cluster in storing and reading the CKPT. However, a disadvantage of this solution is that data in the memory becomes unavailable when a system crash occurs. The training cluster can avoid accessing a remote storage network only when data stored in the memory is guaranteed to remain accessible during the system crash.

[0074] In a memory storage solution (for example, Gemini), a solution of grouping and backing up the cluster nodes is used, that is, nodes in a same group back up data of other nodes in the group to each other, to improve a probability that the cluster nodes remain available when the system crashes, to avoid unavailability of memory data. However, in the memory storage solution, at least half of memory needs to be consumed to back up data of other nodes, and memory consumption is large. In addition, to back up data of a node to another node, a parameter plane network is scheduled to transmit backup CKPT data in a training process, which may interfere with normal model training traffic and affect training efficiency.

[0075] The foregoing solution cannot have both high performance and low costs. High performance refers to minimizing impact of a GPU card failure in the training process, rapidly recovering a training state, and improving training efficiency. Low costs refers to less storage space, lower small network bandwidth requirements, and reduced waste of GPU card computational power. Detailed analysis is shown in Table 1.

Table 1

Representative solutions	S3, OBS, SFS	JuiceFs, Alluxio, Memarts	AWS Gemini
Main ideas	Direct remote storage	Distributed cache	Memory storage
Disadvantages	High network latency; High bandwidth	Depends on an optimal prefetch algorithm and a local affinity scheduling	High memory consumption; traffic interference

	requirement	algorithm	
Fault recovery performance	Low	Medium	High, but unstable
Storage space consumption	Large	Large	Small
Network bandwidth requirement	High	Low	Low
Waste of card computational power	Large	Medium	Low, but unstable

[0076] The direct remote storage relies too heavily on remote storage services. Therefore, in a large-scale model training cluster, the network latency is high and the bandwidth requirement is high. When a system experiences a fault, a long time is required to read remote data, and the fault recovery performance is poor. If a storage frequency of the CKPT is increased, the storage space consumption is large. When a fault occurs, a large number of cards are in an idle state waiting for fault recovery, resulting in a large waste of computational power.

[0077] The distributed cache distributes data on different nodes in the cluster. Therefore, performance of the distributed cache depends on the optimal prefetch algorithm to ensure cache hit during fault recovery, and the local affinity scheduling algorithm to ensure data hit on a local node. If the storage frequency of the CKPT is increased, the storage space consumption is large, and the bandwidth requirement for a remote network is lowered. Since it is difficult to increase the storage frequency of the CKPT, the training state cannot be restored to a recent state, resulting in a large waste of card computational power.

[0078] Gemini uses the memory storage solution. However, the solution depends on the memory of each node to back up data of other nodes. Therefore, the memory consumption is high and traffic interference exists. During fault recovery, data on the memory node needs to be ensured to remain available, and data on the backup node also is available. Gemini has high performance but is unstable. Gemini stores temporary data in the memory, which can significantly improve the storage frequency of the CKPT. This solution lowers the requirement for remote network bandwidth. When a fault occurs, this solution may restore the training state to a recent state, resulting in a small waste of card computational power. However, because the solution depends on the recovered data being available in the node memory, the performance of the solution is unstable.

[0079] For low performance and high costs of checkpoint storage in a large model training cluster, an embodiment of this application provides a model training method. The model training method includes a training cluster checkpoint storage process.

[0080] In the large model training cluster, the memory of each node is typically at least three

times the memory of all graphics cards on the node, and has sufficient space to store a plurality of copies of CKPT data. Therefore, in the model training method according to an embodiment of this application, the memory of a node is directly used to store CKPT data of each round of training.

[0081] In addition, the distributed training cluster usually uses a plurality of model groups to perform training in a parallel manner. When each round of training ends, the plurality of model groups perform data aggregation and synchronization. During each round of data synchronization, data of the plurality of model groups is the same. Therefore, in the model training method according to an embodiment of this application, a data redundancy characteristic of the plurality of model groups is used to perform backup redundancy for memory data.

[0082] To implement mutual backup of CKPT data between the plurality of model groups, in the model training method according to an embodiment of this application, a CKPT version of each node is tracked to ensure that backup data among the plurality of model groups is in a same state, to ensure data consistency.

[0083] Specifically, a network in which GPU cards between nodes directly communicate with each other and that is used for exchanging training parameters is referred to as a parameter plane network. In the model training method according to an embodiment of this application, distributed cluster memory collaborative management is used, and the GPU cards between nodes are connected to the parameter plane network. A cluster memory collaborative management control senses and controls the storage and recovery of a model CKPT. The cluster memory collaborative management control collaborates with cluster nodes to directly store a CKPT of each node to a local memory, implementing storage of the CKPT. By tracking and coordinating CKPT versions of all nodes, a cluster training state can be restored by using the local memory and parameter plane network to transmit the CKPT required for fault recovery. The cluster memory collaborative management control senses a CKPT storage state in a training process, including information such as generation time, storage location, size, and version number. The CKPT is directly stored in the local memory or a storage device, to achieve fast storage of the CKPT.

[0084] FIG. 5 is a schematic flowchart of a model training method according to an embodiment of this application.

[0085] S500: Start a model training process.

[0086] Specifically, in S500, model training is performed through a training cluster.

[0087] The model training method according to this embodiment of this application is applied to a training cluster for model training.

[0088] The training cluster includes a plurality of model training groups, and the plurality of model training groups are configured to train a neural network model in parallel in a data parallel manner. Each model training group is for a complete model structure. Each model training group

includes a plurality of training nodes, and different model training groups include a same node structure, that is, the different model training groups include a same quantity of training nodes, and training nodes included in the different model training groups are in one-to-one correspondence.

[0089] For example, the training cluster includes at least two model training groups (a first model training group and a second model training group), and the first model training group and the second model training group each may correspond to a complete model structure. The first model training group includes at least a first training node, and the second model training group includes at least a second training node. The first training node corresponds to the second training node based on a node structure.

[0090] It may be understood that the first model training group may further include a third training node, and the second model training group may further include a fourth training node. The third training node corresponds to the fourth training node based on a node structure.

[0091] S501: After one round of training ends, generate a model parameter training result of the current round of training, that is, generate a CKPT of the current round of training.

[0092] The CKPT of the current round of training corresponds to a complete model, that is, the CKPT of the current round of training includes CKPTs corresponding to all training nodes in one model training group.

[0093] Specifically, the CKPT of the current round of training includes at least a CKPT A, and the CKPT A is a CKPT of the first training node and the second training node. It may be understood that the CKPT of the current round of training may further include a CKPT B, and the CKPT B is a CKPT of the third training node and the fourth training node.

[0094] Specifically, in S501, data is exchanged with different training nodes to obtain trained parameters, and model parameters of all nodes are aggregated to a central node for average calculation when one round of training ends. A calculation result is used as a model parameter training result of the round of training, that is, the CKPT of the current round of training. The central node may be any training node or any combination of a plurality of training nodes in the training cluster. The central node may also be any compute node in the training cluster other than all training nodes.

[0095] S510: Synchronize the CKPT of the current round of training to all model training groups (all training nodes).

[0096] Specifically, in S510, a same CKPT (the CKPT of the current round of training) is loaded into all the model training groups, and training nodes in a same model training group respectively load CKPTs corresponding to the training nodes. That is, based on a node structure, a part of the CKPT of the current round of training corresponding to one training node is loaded into

corresponding training nodes in the model training group.

[0097] Specifically, the CKPT A is synchronized to the first training node and the second training node, and the CKPT B is synchronized to the third training node and the fourth training node.

[0098] S520: Store a CKPT of each training node in a memory of the training node.

[0099] Specifically, in S520, the first training node stores the CKPT A in a memory of the first training node, and the second training node stores the CKPT A in a memory of the second training node. It may be understood that the third training node stores the CKPT B in a memory of the third training node, and the fourth training node stores the CKPT B in a memory of the fourth training node.

[00100] FIG. 6 is a diagram of a structure of a training cluster according to an embodiment of this application.

[00101] As shown in FIG. 6, in an embodiment, the training cluster includes at least a model training group 61 and a model training group 62.

[00102] The model training group 61 includes training nodes 600 to 615, and the model training group 62 includes training nodes 616 to 631.

[00103] In S500, training nodes in the model training group 61 perform data exchange for model training, and training nodes in the model training group 62 perform data exchange for model training. After one round of training is completed, model parameters trained by the model training group 61 (the training nodes 600 to 615) and the model training group 62 (the training nodes 616 to 631) are summarized and calculated by using an average value, to obtain a model parameter training result of the current round of training, that is, the CKPT of the current round of training. The CKPT of the current round of training includes CKPT 00 to 15.

[00104] The CKPT of the current round of training is synchronized to the model training group 61 and the model training group 62. The training nodes 600 to 615 of the model training group 61 respectively load corresponding CKPTs in the CKPT of the current round of training, and the training nodes 616 to 631 of the model training group 62 respectively load corresponding CKPTs in the CKPT of the current round of training. That is, the CKPTs 00 to 15 are respectively synchronized to the training nodes 600 to 615, and the CKPTs 00 to 15 are respectively synchronized to the training nodes 616 to 631.

[00105] After CKPT synchronization, CKPTs loaded by the training nodes 600 to 615 are consistent with CKPTs loaded by the training nodes 616 to 631. For example, a CKPT loaded by a training node 600 is consistent with a CKPT loaded by a training node 616.

[00106] Further, the CKPTs 00 to 15 are respectively stored in the memory of the training nodes 600 to 615, and the CKPTs 00 to 15 are respectively stored in the memory of the training nodes

616 to 631.

[00107] According to the model training method in this embodiment of this application, memories of training nodes are used to back up CKPTs of the training nodes. In a backup process, the training nodes do not need to communicate with each other, to greatly reduce consumption of communication resources.

[00108] Optionally, in an embodiment, a CKPT of a latest round of training is stored in the memory of the training node.

[00109] Optionally, in another embodiment, a CKPT of multi-round training including the latest round is stored in the memory of the training node. Specifically, a key value (Key Value, KV) is maintained on the training node, where Key is a timestamp-based version number of the CKPT, and Value is metadata related to the CKPT, including a size, a storage location, and the like of the CKPT.

[00110] Optionally, in an embodiment, to avoid memory occupancy overflow, a size of a CKPT in the memory of the training node is monitored, and when the size of the CKPT in the memory of the training node exceeds a preset value, an earliest version of a CKPT in the memory is deleted.

[00111] Optionally, in an embodiment, to avoid memory occupancy overflow, a quantity of versions of CKPTs stored in the memory of the training node is limited. When the quantity of versions of CKPTs in the memory of the training node reaches the preset value, the earliest version of the CKPT in the memory is deleted.

[00112] Considering that a capacity of the memory of the training node is always limited, to expand the capacity of the memory, optionally, in an embodiment, some or all of the CKPT stored in the memory of the training node is stored in another storage medium (for example, an SSD) other than the memory.

[00113] When there are a large quantity of model training groups, large model parallelism may be implemented. However, because CKPT data stored in the memory of training nodes in different model training groups is consistent, backup redundancy of the CKPT data is also large. To reduce the backup redundancy, optionally, in an embodiment, the model training groups are grouped, and a plurality of model training groups in one group jointly back up one piece of CKPT data. For example, four model groups back up one piece of CKPT data, so that a memory of a cluster stores more model versions.

[00114] According to the model training method in this embodiment of this application, the memory of the training node is used to store the CKPT data, and the CKPT data in the memory is soon replaced by a new version, and is data that is temporarily stored. However, in an actual use process, a version of a CKPT needs to be persisted at intervals for further analysis and use in the future.

[00115] For the foregoing case, in an embodiment, CKPT data in the memory of all training nodes in one model training group of the plurality of model training groups is uploaded to a remote storage service based on a CKPT persistence requirement.

[00116] According to the method in an embodiment of this application, a solution in which remote storage is removed from a critical path is used, and cluster memory data is asynchronously persisted to a remote end, to reduce dependence on performance of the remote end. In addition, according to the method in an embodiment of this application, temporary storage of the CKPT by the memory is distinguished from persistent storage of the CKPT by the remote storage service, and space consumption of the remote storage is not increased.

[00117] In an embodiment, to ensure that a memory does not overflow, new checkpoint data overwrites old checkpoint data when the new checkpoint data is stored in the memory. Alternatively, memory space occupancy is monitored, and the old checkpoint data is deleted when the space is insufficient. However, in a data upload phase, a situation may occur in which a version of checkpoint data is deleted or overwritten by data of a new version when the checkpoint data is being uploaded halfway. To avoid this situation, in an embodiment, during uploading of CKPT data to the remote storage service, memory data of the model training group is prevented from being deleted or replaced by a new version.

[00118] According to the method in an embodiment of this application, generation of the new checkpoint data can be prevented from interfering with uploading of the checkpoint data, to ensure that the checkpoint data can be successfully uploaded to the remote storage service.

[00119] Specifically, in an embodiment, for a model training group that is uploading CKPT data to the remote storage service, deleting CKPT data stored in the memory is prohibited, and overwriting the stored CKPT data with new CKPT data is prohibited. That is, the CKPT data stored in the memory is frozen in an entire data upload process. However, in a CKPT data upload process, new CKPT data may be stored in unused space in the memory.

[00120] Further, to avoid overflow of memory storage space, in another embodiment, S520 is not performed for the model training group that is uploading CKPT data to the remote storage service. That is, in the entire data upload process, the new CKPT data is not stored in the memory. In this way, the CKPT data stored in the memory is not overwritten by the new CKPT data, and the stored CKPT data is not deleted due to insufficient storage space. Ensure that the CKPT data stored in the memory remains unchanged in the entire data upload process. After the CKPT data is uploaded to the remote storage service, a latest version of CKPT data that is not stored to the memory is synchronized from another model training group (for example, an adjacent model training group) in the CKPT data upload process.

[00121] According to the method in an embodiment of this application, generation of the new

checkpoint data can be prevented from interfering with uploading of the checkpoint data, to ensure that the checkpoint data can be successfully uploaded to the remote storage service; and it may be ensured that versions of checkpoint data of a training node that uploads checkpoint data and a training node that does not upload checkpoint data are consistent.

[00122] Based on a training cluster checkpoint storage process provided in an embodiment of this application, the model training method in an embodiment of this application further includes a training cluster checkpoint recovery process.

[00123] In the model training method according to an embodiment of this application, fast data recovery is performed by using a CKPT stored in a memory of a training node and a parameter plane network.

[00124] Specifically, when a training node is faulty, the faulty training node is recovered, and states of all the training nodes are restored to a latest CKPT, to ensure that all the nodes are in a consistent state.

[00125] For a training node that is not faulty, latest CKPT data is preferentially read from a local memory. Because a speed of reading the memory is high, CKPT recovery efficiency is high in this manner.

[00126] A memory of the training node whose fault is recovered does not have a previously stored CKPT. Therefore, CKPT data in a memory of a corresponding training node in another model training group is read through a parameter plane network, to recover a CKPT state.

[00127] FIG. 7 is a schematic flowchart of a model training method according to an embodiment of this application.

[00128] S700: Start a model training process. Refer to S500.

[00129] S710: Determine whether a training node is faulty.

[00130] Return to S700 if no training node is faulty.

[00131] Perform S711 if a training node is faulty.

[00132] S711: Interrupt training operations of all training nodes.

[00133] S720: Recover a faulty training node.

[00134] S730: For a training node other than the faulty training node, data recovery is performed on the faulty training node data by using a latest version of CKPT data stored in a memory of the training node.

[00135] S731: Data recovery is performed on the faulty training node by using a latest version of CKPT data stored in a memory of a training node that is in another model group and that corresponds to the faulty training node based on a node structure.

[00136] For example, a first training node in a first model training group is faulty. A second training node in a second model training group performs data recovery on the training node by

using a latest version of CKPT data stored in a memory of the second training node. The first training node in the first model training group obtains checkpoint data from the memory of the second training node, to perform data recovery on the first training node.

[00137] As shown in FIG. 6, in an embodiment, if the training node 615 is faulty, training of the training nodes 600 to 615 and the training nodes 616 to 631 is interrupted. The fault of the training node 615 is recovered. The training nodes 600 to 614 are recovered by using a latest version of CKPT data stored in memories of the training nodes 600 to 614, respectively. The training nodes 616 to 631 are recovered by using a latest version of CKPT data stored in memories of the training nodes 616 to 631, respectively. The training node 615 is recovered by using a latest version of CKPT data stored in a memory of the training node 631.

[00138] According to the method in this embodiment of this application, in a large model training cluster, a problem of low performance and high costs of storing and reading a CKPT may be resolved, a time of storing and reading the CKPT may be reduced, a storage frequency of storing the CKPT may be increased, storage space consumption may be not increased, a bandwidth requirement for a remote storage service may be lowered, and a waste of GPU card computational power may be reduced.

[00139] According to the method in an embodiment of this application, in a scenario in which a fault occurs, data is retransmitted and recovered by using a feature of redundancy backup of a multi-model group. In this way, the remote storage service is removed from a critical path, to lower performance and bandwidth requirements for remote storage, thereby significantly reducing costs of an entire system.

[00140] Further, before model training, when a training node in a training cluster loads a CKPT in an initial state, local memories of all training nodes do not store CKPT data. If all training nodes are enabled to pull the CKPT data from the remote storage service, an instantaneous bandwidth requirement of the remote storage is extremely high, and performance is severely degraded.

[00141] For the foregoing case, in an embodiment, when the training node in the training cluster loads the CKPT in the initial state, one model training group among the plurality of model training groups is caused to pull the CKPT from the remote storage service. The training node in the model training group that downloads the CKPT from the remote storage service sends the CKPT (for example, in a broadcast manner) to training nodes in another model training group through a parameter plane network.

[00142] For example, the first training node in the first model training group downloads initial checkpoint data corresponding to the training node; and the first training node in the first model training group broadcasts, to the second training node in the second model training group, the initial checkpoint data downloaded by the first training node.

[00143] Optionally, in a broadcast process, a model group that has received the broadcast data may broadcast the data to more model groups.

[00144] For example, the training node in the first model training group is used to download the initial checkpoint data; the training node in the first model training group broadcasts the downloaded initial checkpoint data to the training node in the second model training group; and the training node in the second model training group broadcasts the downloaded initial checkpoint data to the training node in a third model training group.

[00145] According to the method in an embodiment of this application, initial checkpoint data is downloaded to one model training group, and the initial checkpoint data is sent to another model training group through the model training group, to lower a bandwidth requirement for the initial checkpoint data, and improve efficiency of downloading the initial checkpoint data.

[00146] Further, according to the model training method provided in this embodiment of this application, an embodiment of this application further provides a model training apparatus, and the model training apparatus is used in a training cluster. For method steps performed by the model training apparatus, refer to the method steps in the foregoing method embodiments.

[00147] FIG. 8 is a diagram of a model training apparatus according to an embodiment of this application.

[00148] As shown in FIG. 8, in an embodiment, a model training apparatus 800 is used in a training cluster 810.

[00149] The training cluster 810 includes at least a first model training group 811 and a second model training group 812. The first model training group 811 and the second model training group 812 are respectively for a complete model structure. The first model training group 811 and the second model training group 812 include a same node structure.

[00150] As shown in FIG. 8, the first model training group 811 includes at least a first training node 813, and the second model training group 812 includes at least a second training node 814. The first training node 813 corresponds to the second training node 814 based on a node structure.

[00151] The first model training group 811 further includes a third training node 815, and the second model training group 812 further includes a fourth training node 816. The third training node 815 corresponds to the fourth training node 816 based on a node structure.

[00152] After one round of training ends, the training cluster 810 generates a model parameter training result of the current round of training, that is, generates a CKPT of the current round of training. Specifically, when one round of training ends, the training cluster 810 calculates an average of model parameters of all training nodes, and uses a calculation result as checkpoint data of the round of training.

[00153] Specifically, the training cluster 810 generates at least a CKPT A after one round of

training ends, where the CKPT A is a CKPT of the first training node 813 and the second training node 814. It may be understood that the training cluster 810 further generates a CKPT B after one round of training ends, and the CKPT B is a CKPT of the third training node 815 and the fourth training node 816.

[00154] The training cluster 810 synchronizes the CKPT of the current round of training to each model training group in the training cluster 810. Specifically, the training cluster 810 synchronizes, based on a node structure, a CKPT of a single training node in the CKPT of the current round of training to a corresponding training node of each model training group. That is, the CKPT A is synchronized to the first training node 813 and the second training node 814, and the CKPT B is synchronized to the third training node 815 and the fourth training node 816.

[00155] In an embodiment, the model training apparatus 800 includes a storage control module 801. The storage control module 801 instructs each training node in the training cluster 810 to store, in a memory of the training node, a CKPT obtained by the training node.

[00156] That is, the storage control module 801 instructs the first training node 813 to store, in a memory of the first training node 813, the CKPT A obtained by the first training node 813, and the storage control module 801 instructs the second training node 814 to store, in a memory of the second training node 814, the CKPT A obtained by the second training node 814.

[00157] The storage control module 801 further instructs the third training node 815 to store, in a memory of the third training node 815, the CKPT B obtained by the third training node 815, and the storage control module 801 further instructs the fourth training node 816 to store, in a memory of the fourth training node 816, the CKPT B obtained by the fourth training node 816.

[00158] In a model training process, if a training node in the training cluster 810 is faulty, the training cluster 810 interrupts training operations of all training nodes, and recovers the faulty training node.

[00159] In an embodiment, the model training apparatus 800 further includes a recovery control module 802. The recovery control module 802 is configured to: when a training node in the training cluster 810 is faulty, instruct the faulty training node to perform data recovery on the training node by using a latest version of CKPT data stored in a memory of a training node that is in another model group and that corresponds to the faulty training node based on a node structure, and instruct a training node other than the faulty training node to perform data recovery on the training node by using a latest version of CKPT data stored in a memory of the training node.

[00160] For example, if the first training node 813 is faulty in a process of training the model, the recovery control module 802 instructs the first training node 813 to obtain checkpoint data from the memory of the second training node 814, to perform data recovery on the first training node 813. The recovery control module 802 instructs the second training node 814 to obtain

checkpoint data from the memory of the second training node 814, to perform data recovery on the second training node 814; instructs the third training node 815 to obtain checkpoint data from the memory of the third training node 815, to perform data recovery on the third training node 815; and instructs the fourth training node 816 to obtain checkpoint data from the memory of the fourth training node 816, to perform data recovery on the fourth training node 816.

[00161] Optionally, in an embodiment, the model training apparatus 800 further includes an initialization module 803. The initialization module 803 is configured to: before the training cluster 810 starts to train the model, instruct a model training group in the training cluster 810 to download initial checkpoint data corresponding to a complete model structure, that is, each training node in the model training group downloads initial checkpoint data corresponding to the training node; and instruct a model training group that completes downloading the initial checkpoint data to send the downloaded initial checkpoint data to another model training group, that is, a training node that completes downloading the initial checkpoint data sends the downloaded initial checkpoint data to a corresponding training node in the another model training group.

[00162] Specifically, the initialization module 803 instructs the first training node 813 to download initial checkpoint data corresponding to the first training node 813, and instructs the first training node 813 to broadcast the downloaded initial checkpoint data to the second training node 814. It may be understood that the initialization module 803 further instructs the third training node 815 to download initial checkpoint data corresponding to the third training node 815, and instructs the third training node 815 to broadcast the downloaded initial checkpoint data to the fourth training node 816.

[00163] In an embodiment, the model training apparatus 800 further includes a data upload module 804. The data upload module 804 is configured to instruct a model training group in the training cluster 810 to upload checkpoint data stored in a memory of each training node in the model training group to the remote storage service. In a process in which the model training group uploads the checkpoint data stored in the memory of each training node in the model training group to the remote storage service, another model training group does not upload checkpoint data to the remote storage service.

[00164] For example, the data upload module 804 instructs the first model training group 811 to upload checkpoint data stored in a memory of each training node in the first model training group 811 to the remote storage service. Meanwhile, the second model training group 812 does not upload the checkpoint data to the remote storage service.

[00165] Optionally, in an embodiment, the storage control module 801 is further configured to: in a process in which a model training group uploads checkpoint data to the remote storage service, prohibit deleting checkpoint data stored in a memory of each training node in the model training

group, and prohibit using new checkpoint data to overwrite the checkpoint data stored in the memory of each training node in the model training group.

[00166] For example, in a process in which the first model training group 811 uploads the checkpoint data, the storage control module 801 prohibits deleting the checkpoint data stored in the memory of each training node in the first model training group 811, and prohibits using new checkpoint data to overwrite the checkpoint data stored in the memory of each training node in the first model training group 811.

[00167] Optionally, in another embodiment, the storage control module 801 is further configured to: in a process in which a model training group uploads checkpoint data to the remote storage service, stop storing new checkpoint data in a memory of a training node in the model training group.

[00168] For example, in a process in which the first model training group 811 uploads the checkpoint data, the storage control module 801 stops storing new checkpoint data in the memory of the training node in the first model training group 811.

[00169] Further, in an embodiment, the storage control module 801 is further configured to: after a model training group uploads checkpoint data, synchronize new checkpoint data stored in a memory of a training node in another model training group (the new checkpoint data is checkpoint data that is not stored in a memory of a training node in the model training group in a process of uploading checkpoint data by the model training group) to a memory of a training node in the model training group that uploads the checkpoint data.

[00170] For example, after the first model training group 811 uploads the checkpoint data, new checkpoint data stored in the memory of the training node in the second model training group 812 is synchronized to the memory of the training node in the first model training group 811.

[00171] In descriptions of embodiments of this application, for ease of description, various modules are described separately by function. Division into the modules is merely logical function division. During implementation of embodiments of this application, the functions of the modules may be implemented in a same piece of or a plurality of pieces of software and/or hardware.

[00172] Specifically, in actual implementation, all or some of the apparatus provided in embodiments of this application may be integrated into one physical entity (for example, a GPU or a processor of another type), or may be physically separated. In addition, the modules may all be implemented in a form of software invoked by a processing element; or may all be implemented in a form of hardware; or some modules may be implemented in a form of software invoked by a processing element, and some modules may be implemented in a form of hardware. For example, a detection module may be an independently disposed processing element, or may be integrated in one chip of an electronic device for implementation. Implementations of other modules are

similar. In addition, all or some of the modules may be integrated, or may be independently implemented. During implementation, steps in the foregoing methods or the foregoing modules can be implemented by using a hardware integrated logical circuit in the processing element, or by using instructions in a form of software.

[00173] For example, in the foregoing embodiment, the storage control module 801, the recovery control module 802, the initialization module 803, and the data upload module 804 may all be implemented by using software, or may be implemented by using hardware. For example, the following uses the storage control module 801 as an example to describe an implementation of the storage control module 801. Similarly, for implementations of the recovery control module 802, the initialization module 803, and the data upload module 804, refer to the implementation of the storage control module 801.

[00174] The module is used as an example of a software functional unit, and the storage control module 801 may include code running on a computing instance. The computing instance may include at least one of a physical host (compute device), a virtual machine, and a container. Further, there may be one or more computing instances. For example, the storage control module 801 may include code running on a plurality of hosts/virtual machines/containers. It should be noted that the plurality of hosts/virtual machines/containers configured to run the code may be distributed in a same region (region), or may be distributed in different regions. Further, the plurality of hosts/virtual machines/containers configured to run the code may be distributed in a same availability zone (availability zone, AZ), or may be distributed in different AZs. Each AZ includes one data center or a plurality of data centers that are geographically close to each other. Usually, one region may include a plurality of AZs.

[00175] Similarly, the plurality of hosts/virtual machines/containers configured to run the code may be distributed in a same virtual private cloud (virtual private cloud, VPC), or may be distributed in a plurality of VPCs. Usually, one VPC is disposed in one region. A communication gateway needs to be disposed in each VPC for communication between two VPCs in a same region and cross-region communication between VPCs in different regions. The VPCs are interconnected through the communication gateway.

[00176] The module is used as an example of a hardware functional unit, and the storage control module 801 may include at least one compute device, for example, a server. Alternatively, the storage control module 801 may be a device implemented by using an application-specific integrated circuit (application-specific integrated circuit, ASIC), a programmable logic device (programmable logic device, PLD), or the like. The PLD may be implemented by a complex programmable logic device (complex programmable logic device, CPLD), a field-programmable gate array (field-programmable gate array, FPGA), generic array logic (generic array logic, GAL),

or any combination thereof.

[00177] The plurality of compute devices included in the storage control module 801 may be distributed in a same region, or may be distributed in different regions. The plurality of compute devices included in the storage control module 801 may be distributed in a same AZ, or may be distributed in different AZs. Similarly, a plurality of compute devices included in the storage control module 801 may be distributed in a same VPC, or may be distributed in a plurality of VPCs. The plurality of compute devices may be any combination of compute devices such as a server, an ASIC, a PLD, a CPLD, an FPGA, and GAL.

[00178] It should be noted that, in another embodiment, the storage control module 801 may be configured to perform any step in the XXX method, the recovery control module 802 may be configured to perform any step in the XXX method, the initialization module 803 may be configured to perform any step in the XXX method. Steps to be implemented by the storage control module 801, the recovery control module 802, and the initialization module 803 may be specified as required, and different steps in the XXX method are respectively implemented by the storage control module 801, the recovery control module 802, and the initialization module 803, to implement all functions of the YY apparatus.

[00179] Specifically, in an embodiment, one or more of the storage control module 801, the recovery control module 802, the initialization module 803, and the data upload module 804 may be implemented based on a model training framework.

[00180] In this embodiment of this application, a code library used by a developer to develop a deep learning model is referred to as a model training framework, and may perform a complete process such as model definition, training, and inference. The model training framework provides memory management for training nodes to back up CKPTs to the memory and recover training nodes from the CKPTs in the memory.

[00181] FIG. 9 is a diagram of a training node deployment scenario according to an embodiment of this application.

[00182] As shown in FIG. 9, each training node is equipped with a plurality of graphics cards used for model training and a large-capacity memory. In addition to a conventional transmission control protocol (Transmission Control Protocol, TCP) network between training nodes, a high-bandwidth parameter plane network is further included between the graphics cards of the training nodes, and is used for parameter transmission of the model.

[00183] A training framework used for model training and a software package of the method according to an embodiment of this application are installed in an operating system of each training node. The software package of the method according to the embodiment of this application provides a software development kit (Software Development Kit, SDK) that can be invoked by

the training framework. The SDK is used as an interface for interaction between the training framework and the software package of the method according to the embodiment of this application, and is configured to invoke a function of storing and restoring a CKPT of the method according to an embodiment of this application.

[00184] A remote storage service that provides data persistence services is included outside the training cluster. The remote storage service provides a portable operating system interface POSIX (Portable Operating System Interface of UNIX, POSIX).

[00185] According to the method according to an embodiment of this application, cluster memory collaborative management is used, and a node memory stores an entire CKPT, to improve frequency of storing the CKPT. In addition, a cluster memory control manages a multi-node CKPT, and supports group redundancy backup. In addition, the CKPT is directly stored in the memory, to greatly reduce latency of storing the CKPT.

[00186] According to the method according to an embodiment of this application, data in a local memory is directly read by using a local memory and a parameter plane network for collaborative recovery, to greatly shorten recovery latency of the CKPT. In addition, a faulty node reads data through the parameter plane network, to greatly shorten recovery latency.

[00187] According to the method and the apparatus provided in embodiments of this application, a product related to the method and the apparatus includes a CKPT storage and download service provided for a large model distributed training cluster, and may be bound to a large model training cluster for sale in a form of a service that improves CKPT storage and access efficiency.

[00188] Specifically, in the product related to the method and the apparatus provided in the embodiments of this application,

a memory of a training node stores complete CKPT data of the node. In a computer system structure, there are many designs and products that effectively use a memory to store or cache data. According to the method in an embodiment of this application, a major feature of using a memory is that a memory of a node is used to store complete CKPT data of the node instead of caching some data. In this way, when the system needs to recover the CKPT data, corresponding data may be completely read from the memory of the local node as long as the data of a corresponding version is stored in the node, and the data does not need to be read from other places.

[00189] Collaborative fast recovery using the parameter plane network and the local memory: Different from the solution of reading the CKPT from the remote storage, according to the method in an embodiment of this application, data is preferentially read from the local memory during fault recovery. Because each node stores complete CKPT data of the node, the solution can achieve second-level fault recovery. If the local node is replaced due to a fault, data is read from a redundant backup node in the cluster through the parameter plane network. For an AI training cluster with 32

model groups, each node can have a maximum of 32 redundant backups. Such redundant backups greatly improve availability of node memory backup data.

[00190] The CKPT data is distinguished into temporary storage and persistent storage. According to the method in an embodiment of this application, in a training process, CKPT data of each round is stored in a memory as temporary storage, and the memory data is asynchronously persisted to a remote storage based on a persistence requirement, and the remote storage is removed from a critical path, balancing costs of storage and network bandwidth with performance of storage and recovery.

[00191] FIG. 10 is a diagram of a training node deployment scenario according to an embodiment of this application.

[00192] A training cluster includes at least a model training group 901 and a model training group 902. The model training group 901 includes training nodes 911 to 914, and the model training group 902 includes training nodes 915 to 918.

[00193] A training framework and related components are installed in the training cluster, and run in an operating system running environment of the training cluster, to provide a training framework service 920.

[00194] The training nodes 911 to 914 and training nodes 915 to 918 are installed with SDKs with related functions to interface with the training framework.

[00195] In a model training process, the training framework service 920 invokes a storage function interface of the SDK to perform CKPT storage.

[00196] A remote storage service 930 is deployed outside the training cluster. In the model training process, the training framework service 920 periodically uploads CKPTs stored in memories of the training nodes 911 to 914 or the training nodes 915 to 918 to the remote storage service 930.

[00197] Specifically, before model training starts, the training framework service 920 invokes a loading interface provided by an SDK of the training node. When it is detected that the training node does not store CKPT data, one model training group (the model training group 901 or 902) is selected from all model training groups, and the model training group 901 or 902 downloads the CKPT from the remote storage service 930.

[00198] The model training group 901 or 902 that downloads the CKPT data from the remote storage service 930 broadcasts data to another training node (a training node in the model training group 902 or 901) through a parameter plane network.

[00199] The training framework service 920 starts a training process, obtains trained parameters through data exchange with different training nodes, and aggregates model parameters of all nodes to a central node for average calculation when one round of training ends, to serve as a model

parameter training result of the round of training, that is, CKPT.

[00200] The central node may be any training node in the training cluster, or the central node may be an independent node that is configured in the training cluster and that is other than the training nodes.

[00201] After one round of training ends and the CKPT is obtained, the central node synchronizes the final CKPT of the current round to all nodes to prepare for a next round of training. In addition, the training framework service 920 invokes a CKPT storage function of an SDK of each training node to store the CKPT in a memory of each training node.

[00202] When the SDK of the training node is invoked to store a temporary CKPT, the SDK notifies the training node to check whether the local memory has sufficient space for storing the current CKPT. If the local memory is insufficient, the SDK checks whether a user has configured an option for overflowing to a disk. If the user has configured the option for overflowing to the disk, a subsequent CKPT storage process is directly stored on a local disk. Otherwise, an earliest CKPT in the local memory is deleted first.

[00203] The training node stores the CKPT of the training node in a pre-allocated local memory or disk, and maintains a KV on the local node, where Key is a timestamp-based version number of the CKPT, and Value is metadata related to the CKPT, including a size, a storage location, and the like of the CKPT.

[00204] In a process of performing a plurality of rounds of model training, for each preset time interval (for example, three hours) or training round (for example, 30 rounds), the training framework service 920 stores the current CKPT in the remote storage service 930 for further analysis and use in the future.

[00205] Specifically, when the training framework service 920 starts, a time (for example, three hours) for persisting to the remote storage service 930 is configured, and when a persistent storage condition is met, the training framework service 920 invokes a persistent storage interface through an SDK of the training node, to perform persistent storage.

[00206] The training framework service 920 selects one of all model training groups as a model training group for persistent storage, and enables the model training group to upload the CKPT stored in the local memory to the remote storage service. During uploading, a latest version of the CKPT in the model group is locked and cannot be deleted. After the selected model training group makes the CKPT persistent to the remote storage, the CKPT persistence operation is complete and a locked state of the CKPT is released.

[00207] During fault recovery, the training framework service 920 invokes the SDK of the training node to load a model CKPT to prepare for resuming training. Specifically, the training framework service 920 determines a stored latest CKPT version, and the training node searches a

local memory of the training node to check whether a CKPT corresponding to the version is stored. If the local memory of the training node stores the CKPT corresponding to the version, the training node directly reads latest CKPT data from the memory, loads the CKPT data to a graphics card to perform a next round of model training, and ends a recovery operation. If the local memory of the training node does not store the CKPT corresponding to the version, the training node reads, through the parameter plane network, CKPT data of the version from a corresponding training node in another model training group, loads the CKPT data to a graphics card to perform a next round of model training, and ends a recovery operation.

[00208] This application further provides a compute device.

[00209] FIG. 11 is a diagram of a compute device according to an embodiment of this application.

[00210] As shown in FIG. 11, a compute device 1100 includes a bus 1102, a processor 1104, a memory 1106, and a communication interface 1108. The processor 1104, the memory 1106, and the communication interface 1108 communicate with each other through the bus 1102. The compute device 1100 may be a server or a terminal device. It should be understood that quantities of processors and memories in the compute device 1100 are not limited in this application.

[00211] The bus 1102 may be a peripheral component interconnect (peripheral component interconnect, PCI) bus, an extended industry standard architecture (extended industry standard architecture, EISA) bus, or the like. Buses may be classified into an address bus, a data bus, a control bus, and the like. For ease of indication, the bus is indicated by using only one line in FIG. 11. However, it does not indicate that there is only one bus or only one type of bus. The bus 1102 may include a path for information transmission between components (for example, the memory 1106, the processor 1104, and the communication interface 1108) of the compute device 1100.

[00212] The processor 1104 may include any one or more of processors such as a central processing unit (central processing unit, CPU), a graphics processing unit (graphics processing unit, GPU), a microprocessor (microprocessor, MP), or a digital signal processor (digital signal processor, DSP).

[00213] The memory 1106 may include a volatile memory (volatile memory), for example, a random access memory (random access memory, RAM). The processor 1104 may further include a non-volatile memory (non-volatile memory), for example, a read-only memory (read-only memory, ROM), a flash memory, a hard disk drive (hard disk drive, HDD), or a solid-state drive (solid-state drive, SSD).

[00214] The memory 1106 stores executable program code, and the processor 1104 executes the executable program code to separately implement functions of the storage control module 801, the recovery control module 802, the initialization module 803, and the data upload module 804,

to implement the model training method provided in this embodiment of this application. In other words, the memory 1106 stores instructions used to perform the model training method provided in this embodiment of this application.

[00215] The communication interface 1108 uses a transceiver module, for example, but not limited to, a network interface card or a transceiver, to implement communication between the compute device 1100 and another device or a communication network.

[00216] An embodiment of this application further provides a compute device cluster. The compute device cluster includes at least one compute device. The compute device may be a server, for example, a central server, an edge server, or a local server in a local data center. In some embodiments, the compute device may alternatively be a terminal device, for example, a desktop computer, a notebook computer, or a smartphone.

[00217] FIG. 12 is a diagram of a compute device cluster according to an embodiment of this application.

[00218] As shown in FIG. 12, the compute device cluster includes at least one compute device 1100. A memory 1106 in one or more compute devices 1100 in the compute device cluster may store same instructions for performing the model training method provided in embodiments of this application.

[00219] In some possible implementations, the memory 1106 in the one or more compute devices 1100 in the compute device cluster may separately store some instructions for performing the model training method provided in embodiments of this application. In other words, a combination of the one or more compute devices 1100 may jointly execute the instructions for performing the model training method provided in embodiments of this application.

[00220] It should be noted that memories 1106 in different compute devices 1100 in the compute device cluster may store different instructions, and are respectively used for performing some functions of the model training apparatus provided in embodiments of this application. In other words, instructions stored in memories 1106 in different compute devices 1100 may implement functions of one or more of the storage control module 801, the recovery control module 802, the initialization module 803, and the data upload module 804.

[00221] In some possible implementations, one or more compute devices in the compute device cluster may be connected through a network. The network may be a wide area network, a local area network, or the like.

[00222] FIG. 13 is a diagram of a network connection of a compute device cluster according to an embodiment of this application.

[00223] As shown in FIG. 13, two compute devices 1100A and 1100B are connected through a network. Specifically, each compute device is connected to the network through a communication

interface in the compute device.

[00224] A connection manner between the compute device clusters shown in FIG. 13 may be based on a consideration that the model method provided in this application requires interaction between a plurality of training nodes, data aggregation between a plurality of training nodes, and data synchronization between a plurality of training nodes.

[00225] It should be understood that functions of the compute device 1100A shown in FIG. 13 may alternatively be implemented by a plurality of compute devices 1100. Similarly, functions of the compute device 1100B may alternatively be implemented by a plurality of compute devices 1100.

[00226] An embodiment of this application further provides another compute device cluster. A connection relationship between compute devices in the compute device cluster may similarly refer to the connection manner in the compute device cluster in FIG. 12 and FIG. 13. A difference is that a memory 1106 in one or more compute devices 1100 in the compute device cluster may store same instructions for performing the model training method.

[00227] In some possible implementations, the memory 1106 in the one or more compute devices 1100 in the compute device cluster may separately store some instructions for performing the model training method. In other words, a combination of the one or more compute devices 1100 may jointly execute instructions for performing the model training method.

[00228] A person skilled in the art should understand that embodiments of this application may be provided as a method, an apparatus, or a computer program product. Therefore, the present invention may use a form of hardware only embodiments, software only embodiments, or embodiments with a combination of software and hardware. In addition, the present invention may use a form of a computer program product that is implemented on one or more computer-usable storage media that include computer-usable program code.

[00229] In several embodiments provided in this application, when any function is implemented in a form of software functional unit and sold or used as an independent product, the function may be stored in a computer-readable storage medium. Based on such an understanding, the technical solutions of this application essentially, the part contributing to the conventional technology, or some of the technical solutions may be implemented in a form of a software product. The computer software product is stored in a storage medium, and includes several instructions for enabling a compute device (which may be a personal computer, a server, a network device, or the like) to perform all or some of the steps of the methods described in embodiments of this application.

[00230] In addition, an embodiment of this application provides a computer program product including instructions. The computer program product may be software or a program product that includes instructions and that can run on a compute device or can be stored in any usable medium.

When the computer program product runs on at least one compute device, the at least one compute device is caused to perform the model training method in embodiments of this application.

[00231] An embodiment of this application further provides a computer-readable storage medium. The computer-readable storage medium may be any usable medium that can be stored by a compute device, or a data storage device, such as a data center, including one or more usable media. The usable medium may be a magnetic medium (for example, a floppy disk, a hard disk, or a magnetic tape), an optical medium (for example, a DVD), a semiconductor medium (for example, a solid-state drive), or the like. The computer-readable storage medium includes instructions. The instructions instruct the compute device to perform the model training method provided in embodiments of this application.

[00232] Embodiments of this application are described with reference to the flowcharts and/or block diagrams of the method, the device (apparatus), and the computer program product according to embodiments of this application. It should be understood that computer program instructions may be used to implement each process and/or each block in the flowcharts and/or the block diagrams and a combination of a process and/or a block in the flowcharts and/or the block diagrams. These computer program instructions may be provided for a general-purpose computer, a dedicated computer, an embedded processor, or a processor of any other programmable data processing device to generate a machine, so that the instructions executed by a computer or a processor of any other programmable data processing device generate an apparatus for implementing a specific function in one or more processes in the flowcharts and/or in one or more blocks in the block diagrams.

[00233] These computer program instructions may be stored in a computer-readable memory that can instruct the computer or any other programmable data processing device to work in a specific manner, so that the instructions stored in the computer-readable memory generate an artifact that includes an instruction apparatus. The instruction apparatus implements a specific function in one or more processes in the flowcharts and/or in one or more blocks in the block diagrams.

[00234] The computer program instructions may alternatively be loaded onto a computer or another programmable data processing device, so that a series of operations and steps are performed on the computer or the another programmable device, so that computer-implemented processing is generated. Therefore, the instructions executed on the computer or the another programmable device provide steps for implementing a specific function in one or more procedures in the flowcharts and/or in one or more blocks in the block diagrams.

[00235] It should be noted that in embodiments of this application, the term "at least one" means one or more, the term "a plurality of" means two or more. "And/or" describes an association

relationship between associated objects and indicates that three relationships may exist. For example, A and/or B may indicate the following cases: Only A exists, both A and B exist, and only B exists. A and B may be singular or plural. The character "/" generally indicates an "or" relationship between the associated objects. "At least one of the following" and similar expressions refer to any combination of these terms, including any combination of single or plural terms. For example, at least one of a, b, and c may indicate a, b, c, a and b, a and c, b and c, or a, b, and c, where a, b, and c may be singular or plural.

[00236] In embodiments of this application, the term "include" or any other variant thereof is intended to cover a non-exclusive inclusion, so that a process, a method, a commodity, or a device that includes a series of elements not only includes those elements but also includes other elements that are not expressly listed, or further includes elements inherent to such a process, method, commodity, or device. Without further limitations, an element limited by "includes a/an" does not exclude other same elements existing in the process, the method, the commodity, or the device that includes the element.

[00237] This application can be described in the general context of executable computer instructions executed by a computer, for example, a program module. Generally, the program module includes a routine, program, object, component, data structure, and the like for executing a particular task or implementing a particular abstract data type. This application may be practiced in distributed computing environments in which tasks are performed by remote processing devices that are connected through a communication network. In a distributed computing environment, program modules may be located in both local and remote computer storage media including storage devices.

[00238] Embodiments in this application are described in a progressive manner, the same and similar parts between various embodiments can be referred to each other, and each embodiment focuses on differences from other embodiments. Especially, an apparatus embodiment is basically similar to a method embodiment, and therefore is described briefly; for related parts, refer to partial descriptions in the method embodiment.

[00239] A person of ordinary skill in the art may be aware that, units and algorithm steps described in embodiments of this application may be implemented by electronic hardware, or a combination of computer software and electronic hardware. Whether the functions are performed by hardware or software depends on particular applications and design constraint conditions of the technical solutions. A person skilled in the art may use different methods to implement the described functions for each particular application, but it should not be considered that the implementation goes beyond the scope of this application.

[00240] It may be clearly understood by a person skilled in the art that, for the purpose of

convenient and brief description, for a detailed working process of the foregoing apparatus and unit, refer to a corresponding process in the foregoing method embodiments. Details are not described herein again.

[00241] The foregoing descriptions are merely specific implementations of this application. Any variation or replacement readily figured out by a person skilled in the art within the technical scope disclosed in this application shall fall within the protection scope of this application. Therefore, the protection scope of this application shall be subject to the protection scope of the claims.